38

Schriften aus der Fakultät Wirtschaftsinformatik und
Angewandte Informatik der Otto-Friedrich-Universität Bamberg

Effective and Efficient Summarization of Two-Dimensional Point Data

Approaches for Resource Description and Selection in Spatial Application Scenarios

Stefan Kufer

University of Bamberg Press

38 Schriften aus der Fakultät Wirtschaftsinformatik und Angewandte Informatik der Otto-Friedrich-Universität Bamberg

Contributions of the Faculty Information Systems and Applied Computer Sciences of the Otto-Friedrich-University Bamberg

Schriften aus der Fakultät Wirtschaftsinformatik und Angewandte Informatik der Otto-Friedrich-Universität Bamberg

Contributions of the Faculty Information Systems and Applied Computer Sciences of the Otto-Friedrich-University Bamberg

Band 38

University of Bamberg Press
2019

Effective and Efficient Summarization of Two-Dimensional Point Data

Approaches for Resource Description and Selection in Spatial Application Scenarios

Stefan Kufer

University of Bamberg Press
2019

Bibliographische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliographie; detaillierte bibliographische Informationen sind im Internet über http://dnb.d-nb.de/ abrufbar.

Diese Arbeit hat der Fakultät Wirtschaftsinformatik und Angewandte Informatik der Otto-Friedrich-Universität Bamberg unter dem Titel „**Effective and Efficient Summarization of Two-Dimensional Point Data for Resource Description and Selection in Spatial Application Scenarios**“ als Dissertation vorgelegen.
1. Gutachter: Prof. Dr. Andreas Henrich
2. Gutachter: Prof. Dr. Christoph Schlieder
Tag der mündlichen Prüfung: 20.05.2019

Herstellung und Druck: docupoint, Magdeburg
Umschlaggestaltung: University of Bamberg Press

ISSN: 1867-7401
ISBN: 978-3-86309-672-4 (Druckausgabe)
eISBN: 978-3-86309-673-1 (Online-Ausgabe)
URN: urn:nbn:de:bvb:473-opus4-551376
DOI: http://dx.doi.org/10.20378/irbo-55137

Acknowledgements

First and foremost, I particularly thank my supervisor Prof. Dr. Andreas Henrich for giving me the chance to pursue my doctoral adventure. His support, inspiration, and the always open door have contributed a great deal to the success of this undertaking as well as helped me to find the way whenever I needed guidance. I additionally want to thank Prof. Dr. Daniela Nicklas and the second assessor of my thesis, Prof. Dr. Christoph Schlieder, for being part of the thesis committee. The cooperation has always been very pleasant.

Further thanks go to all my colleagues I have worked with at Media Informatics Group—Sebastian Boosz, Martin Bullin, Tobias Fries, Andrea Garzarella, Tobias Gradl, Adrian Hub, Jasmin Mikolay, Hans-Christian Sperker, and Michael Sünkel—as well as the administrative staff Siegfried Hofmann and Daniela Pielenhofer: Thank you so much for creating the very enjoyable atmosphere at the chair and the countless hours of laughter and mischief.

Another big thank you to all the student tutors who greatly supported me in my teaching activities, some of them for several years: Hannah Deininger, Matthias Delfs, Florian Fehrenbacher, Tobias Heckel, Michael Herold, Christos Ioannidis, Leon Martin, Johannes Rabold, Sascha Riechel, Madeleine Rosenhagen, Stefan Schedel, and Oliver Siegl.

I owe special gratitude to Peter Garzarella and Dr. Stefan Kolb for proofreading the thesis and their multitude of helpful comments and suggestions, and in particular to my third proofreader Dr. Daniel Blank who in addition provided support and great advice on so many occasions during our time together at the chair. Many thanks also to Felix Engl for his implementation of the R-tree and the excellent cooperation during our mutual research on that topic. Furthermore, I am very grateful to Florence Rat, Dr. Martin Sticht, and Petr Vasilyev for their time and feedback on the trial sessions for the disputation presentation.

Last but not least, I thank my friends and my family for the never-ending support during this long journey.

Kurzfassung

Raum ist überall, und das gilt auch für räumliche Daten. Die digitale Revolution hat zu einem enormen Bestand an räumlichen Daten geführt, und die Menge der neu erzeugten räumlichen Daten nimmt von Tag zu Tag zu. Häufig handelt es sich bei diesen räumlichen Daten um zweidimensionale Punktdaten, die zusätzlich mit Datenobjekten wie z.B. Medienobjekten (etwa Bildern oder Texten) verknüpft sind. Aufgrund der großen Menge an Datenobjekten, die erzeugt werden und anschließend zu verwalten sind, besteht ein Bedarf an effektiven und effizienten Suchsystemen, die in der Lage sind, die räumlichen Eigenschaften dieser Datenobjekte zu adressieren. In diesem Zusammenhang sind verschiedene Suchszenarien vorstellbar: Die Datenobjekte können in einem verteilten System (d.h. auf einer Reihe von verschiedenen Maschinen) oder in einem zentralisierten System (d.h. auf einer einzelnen Maschine) verwaltet werden, was in beiden Fällen zu unterschiedlichen Anforderungen an die entsprechenden Suchsysteme führt. Jedoch ist in beiden Szenarien das Konzept der Ressourcenbeschreibung und -auswahl ein anwendbares Paradigma für Ähnlichkeitssuchen, die in Bezug auf die räumlichen Eigenschaften der Datenobjekte durchgeführt werden. Wenn man sich auf den räumlichen Aspekt konzentriert, ist eine Ressource dabei eine abstrakte Entität, die eine Menge räumlicher Punktdaten verwaltet. Zur Beschreibung des räumlichen Fußabdrucks einer Ressource können die von ihr verwalteten Punktdaten mittels geometrisch abgegrenzter Flächen, die die Punktmenge der Ressource räumlich überdecken, „zusammengefasst“ werden. Diese Ressourcenbeschreibungen können dann für eine gezielte Auswahl der Ressourcen, welche die in Hinblick auf räumliche Eigenschaften relevanten Datenobjekte verwalten, verwendet werden.

Die vorliegende Arbeit beschäftigt sich mit der effektiven und effizienten Zusammenfassung von Mengen zweidimensionaler, räumlicher Punktdaten zum Zwecke der Ressourcenbeschreibung und -auswahl in verschiedenen „räumlichen“ Anwendungsszenarien. Der Begriff „effektiv“ bezieht sich auf eine räumlich sehr genaue geometrische Abgrenzung der zu beschreibenden Datenpunktmengen, wohingegen der Begriff „effizient“ sich auf ihre speicherplatzsparende Repräsentation bezieht. In der vorliegenden Arbeit werden zwei suchbasierte „räumliche“ Anwendungsszenarien untersucht, und in beiden kann die Suchaufgabe über das Konzept der Ressourcenbeschreibung und -auswahl modelliert werden.

Das erste Szenario ist ein verteiltes Anwendungsszenario, bei dem Mengen räumlicher Punktdaten von einer Reihe unabhängiger Ressourcen wie Peers in einem Peer-to-Peer-Netzwerk verwaltet werden. Die Ressourcenbeschreibungen sollen eine gezielte Auswahl der Peers, die die für eine konkrete Anfrage relevanten Datenpunkte verwalten, ermöglichen. Gleichzeitig sollen „irrelevante“ Ressourcen bei der Anfrage-

bearbeitung möglichst nicht kontaktiert werden. Das verteilte Anwendungsszenario ist der Hauptteil dieser Arbeit, und unsere Zusammenfassungsansätze sind speziell für diesen Zweck entwickelt worden. Für diese Ansätze wird eine Vielzahl von Konzepten betrachtet, die die Kompression der Datenpunkte, die Verwendung beliebig komplexer Hüllkörper, die Repräsentation komplexer Objekte (z.B. komplexer Hüllkörper) über eine Menge einfacherer Objekte sowie eine Aufteilung der zu beschreibenden Datenpunktmenge in Gruppen und die anschließende präzise Abgrenzung jeder einzelnen Gruppe beinhalten. Insgesamt werden in dieser Arbeit 14 Zusammenfassungsansätze vorgestellt, die in die Kategorien Datenpartitionierungsansätze (data partitioning approaches), Raumpartitionierungsansätze (space partitioning approaches) und hybride Ansätze (hybrid approaches) unterteilt werden können. Im Anschluss an die Spezifikation geeigneter Ressourcenauswahltechniken, die auf den verschiedenen Zusammenfassungen aufbauen, wird eine umfassende Evaluation der Zusammenfassungsansätze auf Basis von k-Nächste-Nachbarn-Anfragen (kNN-Anfragen) durchgeführt. Die Evaluation berücksichtigt dabei verschiedene Datenkollektionen und unterschiedliche Rahmenbedingungen, um die Robustheit der verschiedenen Ansätze zu untersuchen.
Im zweiten Anwendungsszenario wird eine Nutzung unserer Zusammenfassungsansätze in einer multidimensionalen Datenstruktur untersucht, was ein zentralisiertes Anwendungsszenario darstellt. Konkret werden die beiden für diesen Zweck am besten geeigneten Zusammenfassungsansätze aus dem verteilten Anwendungsszenario in einen R-Baum integriert, welcher Mengen zweidimensionaler Punktdaten verwaltet. Der R-Baum ist eine baumbasierte, zentralisierte multidimensionale Datenstruktur, die von 1984 an über viele Jahre intensiv erforscht wurde. Die „Ressourcen“ innerhalb eines R-Baums sind die Knoten der hierarchischen Baumstruktur. Die Ressourcenbeschreibungen sollen das gezielte Traversieren von Pfaden der Baumstruktur bei der Suche nach den für eine gegebene Anfrage relevanten Datenpunkten ermöglichen. Traditionell werden diese Knoten durch Minimum Bounding Rectangles (MBRs) beschrieben, die sehr speicherplatzeffizient sind, aber eher grobe Beschreibungen des räumlichen Fußabdrucks darstellen. Nach einer Präsentation des klassischen R-Baums werden in der Arbeit die notwendigen Modifikationen am R-Baum, welche zur Integration unserer Zusammenfassungsansätze notwendig sind, diskutiert. Außerdem werden geeignete Algorithmen zur Berechnung von Zusammenfassungen für eine gegebene Menge von Rechtecken entwickelt, da dies eine Voraussetzung für den effizienten Einsatz unserer komplexen Zusammenfassungsansätze in R-Bäumen ist. Im Anschluss an die Spezifikation geeigneter Algorithmen für Bereichs- und kNN-Anfragen wird eine umfangreiche Evaluation durchgeführt, die sowohl das Verbesserungspotenzial gegenüber dem „traditionellen“ R-Baum (der MBR-Zusammenfassungen verwendet) als auch den Zielerreichungsgrad unter Verwendung des geradlinigen Integra-

tionsansatzes, den wir verfolgen, bewertet. Auch in dieser Evaluation werden verschiedene Datenkollektionen und unterschiedliche Rahmenbedingungen berücksichtigt.
Insgesamt präsentiert die Arbeit eine Vielzahl von Zusammenfassungsansätzen für die Beschreibung von Mengen zweidimensionaler Punktdaten. Diese Zusammenfassungsansätze eignen sich hervorragend für den Einsatz innerhalb des untersuchten verteilten Anwendungsszenarios. Darüber hinaus untersuchen wir einen Einsatz der beiden am besten dafür geeigneten Zusammenfassungsansätze in einem sehr intensiv erforschten zentralisierten Anwendungsszenario. Hierbei legen wir Verbesserungspotenziale dar und identifizieren wichtige Hindernisse, die es für unsere Zusammenfassungsansätze innerhalb eines solchen Umfeldes zu überwinden gilt.

Abstract

Space is everywhere, and so is spatial data. The digital revolution has led to an enormous pool of available spatial data, and the amount of newly generated spatial data is increasing day by day. Often, this spatial data is two-dimensional point data that in addition is associated with data objects such as media objects (e.g. pictures or texts). As a consequence of the huge amount of data objects which is generated and then has to be maintained, there is a need for effective and efficient search systems which are capable of addressing the spatial properties of the data objects. In this context, different search scenarios exist: The data objects might be maintained in a distributed system (i.e. on a set of different machines) or in a centralized system (i.e. on a single machine) which, in each case, leads to varying requirements for appropriate search systems. However, in both scenarios, the concept of resource description and selection is an applicable paradigm for conducting similarity searches with regard to the spatial properties of the data objects. Hereby, when focusing on the spatial aspects, a resource is an abstract entity that administers spatial point data. For describing the spatial footprint of a resource, its spatial data point set can be 'summarized' by means of geometrically delineated areas which cover this data point set. These resource descriptions are then usable for a targeted selection of the resources which administer the relevant data objects based on spatial properties.

This thesis is concerned with the effective and efficient summarization of two-dimensional spatial point data as a means for resource description and selection in spatial application scenarios. The term 'effective' refers to a spatially very accurate geometric delineation of the data point set to describe whereas the term 'efficient' relates to its storage-space-efficient representation. Two search-based spatial application scenarios are assessed in this thesis, and in both, the search task can be modelled by adhering to the concept of resource description and selection.

The first scenario is a distributed application scenario in which spatial point data is maintained by a set of independent resources such as peers in a peer-to-peer network. Given a concrete query, the resource descriptions shall enable the targeted selection of the peers administering the relevant data points while ignoring 'irrelevant' resources. The distributed application scenario is the main part of this thesis, and our summarization approaches are specifically developed for this purpose. Hereby, a variety of concepts is considered which include the compression of the data points, the use of arbitrarily complex bounding volumes to delineate them, the representation of complex objects (such as complex bounding volumes) with a set of simple objects, and a division of the data point set to describe into groups and the subsequent concise delineation of each group. Overall, 14 summarization approaches are presented in this thesis which can

be categorized into data partitioning approaches, space partitioning approaches, and hybrid approaches. Following the specification of suitable resource selection schemes which are based on the various summaries, an extensive evaluation is conducted by means of assessing the approaches' performances for k nearest neighbor (kNN) queries. The evaluation considers different data collections and varying environmental conditions in order to assess the robustness of the approaches.
In the second application scenario, a utilization of our summarization approaches in a multidimensional data structure is assessed—which constitutes a centralized application scenario. More specifically, the two summarization approaches of the distributed application scenario which are most suitable for this purpose are integrated into an R-tree which administers sets of two-dimensional point data. The R-tree is a tree-based, centralized multidimensional data structure which has been the subject of intensive research over many years, starting in 1984. Within an R-tree, the 'resources' are the nodes of the hierarchical tree structure. The resource descriptions shall enable the targeted traversal of paths in the tree structure when searching for the relevant data points given a specific query. Traditionally, these nodes are described by Minimum Bounding Rectangles (MBRs) which are very storage-space-efficient but rather coarse descriptions of a spatial footprint. After presenting the classical R-tree, the modifications to the R-tree that are necessary to integrate our summarization approaches are discussed. Also, appropriate algorithms for calculating summaries from a set of rectangles are outlined as this is a prerequisite for the efficient use of our sophisticated summarization approaches in R-trees. Following the specification of appropriate range query and kNN query algorithms, an extensive evaluation is conducted which assesses both the improvement potential over the traditional R-tree using MBR summaries as well as the degree of achievement by means of the straightforward approach to the integration that we pursue. Also in this evaluation, different data collections and varying environmental conditions are considered.
Overall, the thesis presents a wide variety of summarization approaches for describing sets of two-dimensional point data. These summarization approaches are excellently suited for usage in the investigated distributed application scenario. Furthermore, we assess the utilization of the two summarization approaches which are most appropriate for this purpose in an already very intensively researched centralized application scenario. Here, we examine further improvement potentials and identify important obstacles which are to overcome for our summarization approaches in such an environment.

Contents

Glossary

In this work, specific terms are used to describe certain facts conveniently and concisely. This glossary serves as a reference to quickly look up these terms. Of course, all terms are also explained at the appropriate place during their introduction within the work.

accuracy-determining parameter: A parameter of a resource description approach which has a direct influence on the spatial accuracy of the approach's resource summaries. An approach can have one to several accuracy-determining parameters (the only exception is the parameter-free MBR approach). In general, all approach parameters except the cc parameter of $\text{UFS}_{n,cc}$ respectively $\text{DFS}^{b}_{n,cc}$ are accuracy-determining.

approach: See resource description approach.

Approach$^{x,y}_{a,b}$: For a hybrid approach, the subscript parameters are the parameters of the description base while the superscript parameters are the parameters of the refinement. For example, for KDMBR^{b}_{n}, parameter n is for the basic k-d space partition while parameter b is for the quantization accuracy of the refining MBRs.

approximate kNN query: A kNN query for which the returned set of k data points is not guaranteed to be comprised of the true k nearest neighbors but only of data points which are not too far away from the true k nearest neighbors. In comparison to exact kNN queries, they can often be processed much faster.

area: A geometrically delineated region of the data space which usually contains data points. For example, a Minimum Bounding Rectangle (MBR) of a data point set delineates an area.

area-related key figures: A triplet of quantitative key figures relating to the indexed areas of a set of summary-represented resources. In particular, the three key figures are the 'overlap between summaries', the 'data space coverage', and the 'surface area per summary'. These key figures are always calculated technique-specifically for an entire data collection.

baseline: Theoretical optimum of the resource fraction contacted (rfc) to solve a query. It is the rfc value that results if only relevant resources would be contacted while processing a query whereas all irrelevant resources are pruned. In the evaluation, the baseline is averaged over the conducted set of queries.

basic R-tree: The respective R-trees constructed with the use of MBR summaries which serve as a base for the evaluations of the `MBR-like R-trees` in the diverse scenarios (`T_5 scenario`, `T_25 scenario`, `R_5 scenario`, and `R_25 scenario`). Since within an assessment, all `MBR-like R-trees` originate from the same `basic R-tree`, all their structures (i.e. the assignment of child nodes to parent nodes and of data points to leaf nodes) are identical.

bend of a Skyline: Sharp and sudden flattening of the `Skyline` (which has the shape of a curve).

big resource: A resource which administers many data points.

bit vector clipping: Generally, all `resource descriptions` are bit vectors. These bit vectors are physically clipped in a byte-aligned fashion after the rightmost bit of the logical bit vector which is set to '1'. For example, a bit vector of length 16 for which the highest index of a bit set to '1' is 5 (i.e. it is the rightmost bit is set to '1', assuming the bit vector goes from left to right) is clipped after the first byte such that the bit vector has a physical length of only 8.

classical R-tree: An R-tree as specified in the original paper of Antonin Guttman ([Guttman 1984]).

complete quadtree: A `quadtree` which is structurally concluded and sound, i.e. the internal nodes and the leaf nodes of the quadtree structure are arranged in a proper way, and all the required nodes exist.

data-point-size: Amount of storage space required for (the coordinates of) a data point. In this thesis, it always accounts for 8 B of payload data.

data space unit (*dsu*): Base unit of the length of this work's spatial data space which is embedded into a two-dimensional Cartesian coordinate system. The data space ranges from -180 to 180 in the x-dimension and from -90 to 90 in the y-dimension. For example, the distance between the data points p_1 (0,0) and p_2 (1,0) is 1 data space unit (*dsu*) by use of the Euclidean distance.

degenerated MBR: An MBR which has a very large extent in one dimension and very small extents in all the other dimensions. Therefore, it is a very thin, elongated rectangle.

descriptive statistic: "A `descriptive statistic` (in the count noun sense) is a summary statistic that quantitatively describes or summarizes features of a collection of information. In contrast, descriptive

[1] See https://en.wikipedia.org/wiki/Descriptive_statistics, last visit: 05.08.2018.

statistics in the mass noun sense is the process of using and analyzing those statistics."[1] In this work, we use `descriptive statistic` values to summarize certain results like the `rfc` values for a `technique` over a set of queries.

direct representation: See `directly represented resource`. The `direct representation` is the instance depiciting the information on the `directly represented resource`'s spatial footprint.

directly represented resource: The spatial footprint of a resource is 'directly' represented by the coordinates of the data point(s) it administers. The `direct representation` is one of the two subtypes of a `resource description`.

disk access: See `page access`.

duplicates: Two or more data points which feature the exactly same x/y-coordinates respectively lat/long-coordinates.

effectivity: The quality of how well it can be distinguished between relevant and irrelevant resources on basis of a `resource description approach` (respectively one of its parameterizations, i.e. a `technique`). It is measured by the average resource fraction contacted (`rfc`) for processing a query. The less resources are contacted, the better the `effectivity` of the approach.

efficiency: The quality of how storage-space-efficient the `resource descriptions` are. It is measured in the average resource description size (`rds`) required to describe the set of resources by use of a certain `technique`. The lower the average size (in byte), the better the `efficiency` of the approach.

exact kNN query: A kNN query where the true k nearest neighbors with respect to a given query point are returned as the query result.

fanout: In a hierarchical tree structure, the `fanout` of a node is the node's number of child nodes. The `fanout` of a tree is its internal nodes' average number of child nodes.

fixed grid: Grid that spans the entire data space. It is used to decompose the data space into grid cells. The grid cells can be of equal or non-equal size.

full quadtree: A `quadtree` for which at each level up to the leaf level, each former `quadtree cell` has been further partitioned, i.e. each internal node has four child nodes. Consequently, the leaf nodes of the quadtree

structure are all on the same level and the corresponding space partition is equivalent to a regular grid (assuming a `trie-based quadtree`).

full-precision MBR: The MBR of a set of data points whose bounds are captured in a 32-bit single precision floating-point number format. A quantization to depict (approximated) bounds in a storage-space-saving way is *not* applied.

global point density: The density of data points in a specific region of the data space with respect to a data collection's entire data point set. In regions with a high (low) `global point density`, there are a lot (only few) of the data collection's data points.

global space partitioning: The process of decomposing space into subspaces based on the spatial distribution of the entire data collection's data points. Therefore, the resulting space partition is the same for all resources. With regard to space partitioning, it is the counterpart to `local space partitioning`.

high parameterization: Parameterization of an `approach` which results in spatially (very) accurate `resource summaries` (if the `approach`'s other parameterizations serve as a basis for comparison). Also see `highest parameterization`.

highest parameterization of an approach: *The one parameterization* of the tested set of parameterizations of an `approach` which results in the spatially most accurate resource summaries (and therefore *most likely* offers the best `rfc` values for this `approach`).

indexed area: An `area` which is described by the `summary` of a resource. It contains at least one data point of this resource. It is neither known how many data points are contained within the `indexed area` nor where their exact locations within this `area` are. For example, when using the MBR approach for a resource's `summary`, the MBR of a resource's data point set is the `indexed area` of this resource.

initial quadtree: When $\text{MBRQT}^{c,a}$ and $\text{QTMBR}^{b}_{c,a}$ summaries are not calculated from a set of data points but *from a set of input rectangles*, the respective `summary` calculation procedures are adapted: For both `approaches`, an `initial quadtree` is imposed onto the data space to partition. This `initial quadtree` is a `full quadtree` of a target depth *td*. This means that each internal node of the `quadtree` structure has four children with the leaf nodes being at level *td* of the structure. In principle, the `initial quadtree` corresponds to a uniform grid of 4^{td} cells for which each dimension has the same number of cells.

inter-approach comparison: A comparison involving several different approaches. This means the concrete techniques which are compared are based on different approaches. For example, when comparing $\text{QTMBR}^{3}_{32,1.0}$ and $\text{MBRQT}^{16,1.0}$, it is an inter-approach comparison involving two techniques which are based on two different approaches (the $\text{QTMBR}^{b}_{c,a}$ approach and the $\text{MBRQT}^{c,a}$ approach).

intra-approach comparison: A comparison involving several different techniques which are based on the same approach. For example, when comparing $\text{QTMBR}^{3}_{32,1.0}$ and $\text{QTMBR}^{6}_{64,0.1}$, it is an intra-approach comparison involving two techniques of the $\text{QTMBR}^{b}_{c,a}$ approach.

irrelevant resource: Resource which does not administer any data point of a specific query's result.

k-d cell: Subspace of a k-d space partition. Logically, it represents a subregion of the data space. A k-d cell corresponds to a leaf node in the corresponding k-d-tree structure.

k-d-based approaches: Approaches which rely on a k-d-tree as description base. They can be approaches which rely solely on k-d-trees (KD_n) or hybrid approaches which utilize k-d-trees as their description base (KDMBR^{b}_{n}, $\text{KDMAR}^{b,k}_{n}$, $\text{KDQT}^{c,a}_{n}$).

k-d-tree: A means to decompose space into subspaces based on recursion. In each step, the (sub)space to decompose is split into two parts. It consists of the k-d space partition which is the logical decomposition of the space into subspaces (the k-d cells), and the k-d-tree structure which is a corresponding tree structure representing the split hierarchy.

local space partitioning: The process of decomposing space into subspaces based only on the data points administered by the resource to summarize. Therefore, it is a resource-individual space partition. With regard to space partitioning, it is the counterpart to global space partitioning.

low parameterization: Parameterization of an approach which results in spatially (very) coarse resource summaries (if the approach's other parameterizations serve as a basis for comparison). Also see lowest parameterization.

lower-level node: R-tree node which is at a lower level of the R-tree's logical, hierarchical structure (i.e. closer to the leaf node level). The level indices are reverse to the logical structure, i.e. the leaf nodes (which are at the lowest level of the R-tree) have the greatest level index while the

root node (which is at the uppermost level of the R-tree) has the level index 0.

lowest parameterization: *The one parameterization* of the tested set of parameterizations of an `approach` which results in the spatially least accurate `resource summaries` (and therefore *most likely* offers the worst `rfc` values for this `approach`).

MBR-based approaches: Approaches which rely on full-precision (i.e. non-quantized) rectangles as description base. They can be approaches which rely solely on full-precision rectangles (MBR approach, RecMAR$_{k,sl}$) or hybrid approaches which utilize full-precision rectangles as their description base (MBRQT^{c,a}, MARQT$^{c,a}_{k,sl}$). Also see `full-precison MBR`.

MBR-like R-tree: R-tree that has been built by utilizing MBR summaries during the construction phase and in which the MBR summaries have eventually been replaced with more sophisticated `summaries` (either MBRQT^{c,a} or QTMBR$^{b}_{c,a}$ summaries), afterwards. Hereby, storage space limitations are not considered, i.e. the R-tree structure stays the same (i.e. the assignment of child nodes to parent nodes and of data points to leaf nodes).

MBR-like MBR/MBRQT^{c,a}/QTMBR$^{b}_{c,a}$ R-tree: `MBR-like R-tree` in which MBR/MBRQT^{c,a}/QTMBR$^{b}_{c,a}$ summaries describe the spatial footprints of nodes.

MBRselectivity: The `Skyline parameterization` of an approach whose `rfc` value is closest to the `rfc` value of the MBR approach is entitled to be the `Skyline parameterization` at '`MBRselectivity`' of this approach. Hence, '`MBRselectivity`' means as close to the selectivity of the MBR approach as the recorded data enables it.

MBRQT^{8,a} parameterizations: The MBRQT^{c,a} parameterizations for which parameter $c = 8$. In the evaluation of the centralized application scenario, these are MBRQT_1, MBRQT_2, and MBRQT_3.

MBRQT^{16,a} parameterizations: The MBRQT^{c,a} parameterizations for which parameter $c = 16$. In the evaluation of the centralized application scenario, these are MBRQT_4, MBRQT_5, and MBRQT_6.

MBRsize: The `Skyline parameterization` of an `approach` whose `rds` value is closest to the `rds` value resulting for the MBR approach is entitled to be the `Skyline parameterization` at '`MBRsize`' of this `approach`. Hence, '`MBRsize`' means as close to the storage space consumption of the MBR approach as the recorded data enables it.

MBRsize-: It is defined analogously to 'MBRsize' but aims for significantly lower target-rds-values (i.e. rds values significantly below the rds value resulting for the MBR approach). In this work, we refrain from setting 'MBRsize-' to a predefined target-rds-value such as '8 B payload data' since that way, possibly no anchor points at nearby rds values are present on *all* the approaches' respective Skylines. Instead, suitable target-rds-values are selected in the given situations.

MBRsize+: It is defined analogously to 'MBRsize' but aims for significantly greater target-rds-values (i.e. rds values significantly greater than the rds value resulting for the MBR approach). In this work, we refrain from setting 'MBRsize+' to a predefined target-rds-value such as '24 B payload data' since that way, possibly no anchor points at nearby rds values are present on *all* the approaches' respective Skylines. Instead, suitable target-rds-values are selected in the given situations.

non-occupied cell: Cell of a space partition which does *not* contain any data point.

non-occupied subspace: See non-occupied cell.

non-quadtree-utilizing approach: Approach which does not make use of quadtrees (neither as description base nor as refinement): MBR, RecMAR$_{k,sl}$, UFS$_{n,cc}$, KD$_n$, KDMBR$_n^b$, KDMAR$_n^{b,k}$, and DFS$_{n,cc}^b$.

occupied cell: Cell of a space partition which contains at least one data point.

occupied subspace: See occupied cell.

outward appearance: A summary's appearance towards the query processor. In the query processing of the centralized application scenario, only the MINDIST between a summary and the query point is calculated. Hence, the only thing a query processor 'knows' of a summary is the summary's closest point to the query point. This closest point corresponds to the summary's outward appearance towards the query processor.

page access: An R-tree takes secondary memory into account, i.e. it stores its data on a hard drive disk. Therefore, its nodes are mapped to pages on this hard drive disk. Whenever a node is accessed, the corresponding page on the hard drive disk has to be accessed, i.e. a page access has to be conducted.

precise kNN query: See exact kNN query.

pruning: `Pruning` refers to excluding the entire data point set of a resource from further consideration on the basis of the lower bound distance between the query point and the resource's spatial footprint.

quadtree: A means to decompose space into subspaces based on recursion. In each step, the (sub)space is split into four parts. It consists of the quadtree space partition which is the logical decomposition of the space into subspaces (the `quadtree cells`), and the quadtree structure which is a corresponding tree structure representing the split hierarchy.

quadtree cell: Subspace of a quadtree space decomposition. Logically, it represents a subregion of the data space. A `quadtree cell` corresponds to a leaf node in the corresponding quadtree structure.

quadtree condensation: Four equal-colored sibling leaf nodes of the quadtree structure are condensed into their father node which then becomes a leaf node of the same color itself.

quadtree region: See `quadtree cell`.

quadtree subspace: See `quadtree cell`.

quadtree-based approaches: Approaches which rely on a `quadtree` as *description base*. They can be approaches which rely solely on `quadtrees` ($\text{QT}_{c,a}$), or hybrid approaches which utilize `quadtrees` as their description base ($\text{QTMBR}^{b}_{c,a}$, $\text{QTMAR}^{b,k}_{c,a}$).

quadtree-utilizing approaches: Approaches which utilize `quadtrees` *in any form* (be it as description base or as refinement): $\text{QT}_{c,a}$, $\text{GridQT}^{c,a}_{r}$, $\text{KDQT}^{c,a}_{n}$, $\text{QTMBR}^{b}_{c,a}$, $\text{QTMAR}^{b,k}_{c,a}$, $\text{MBRQT}^{c,a}$, and $\text{MARQT}^{c,a}_{k,sl}$.

query cell: The cell of a space partition in which the query point is located.

R_5 scenario: Evaluation scenario in which the `R` collection's data points are inserted into R-trees which have a leaf node capacity of 5 data points.

R_25 scenario: Evaluation scenario in which the `R` collection's data points are inserted into R-trees which have a leaf node capacity of 25 data points.

rds: Resource description size. Usually, it refers to the average resource description size required to describe the given set of resources by a specific `technique`. In corresponding contexts, it can also be used to denote the `descriptive statistic` values (i.e. mean `rds`, min `rds`, max `rds`, median `rds`, and so on) required for describing the given set of resources

by a specific technique, or the resource description size for a single resource and a specific technique.

relevant resource: Resource administering at least one data point that is part of the query result.

relative error: The relative error is assessed with respect to kNN queries. It is the resource fraction contacted (rfc) for which m of the final k closest data points have been found (with $m \leq k$).

resource description: Description of a resource's spatial footprint. It can be either a summary or a direct representation.

resource description approach: Specific approach to describe the spatial footprint of a resource. First, the resource's summary is calculated by the specific summarization approach. If the corresponding summary requires more storage space than the coordinates of the resource's data points, the resource is directly represented instead of summary-represented. The term resource description approach includes the calculation of a summary using a summarization approach and the possible replacement of the summary by the direct representation of the resource. In the centralized application scenario, there is *no* direct representation option.

resource description transmission method shares: Share information about how the resource descriptions are transmitted, i.e. how many percent of the total amount of resource descriptions are of which resource description type. Mostly, the resource description transmission method shares refer to a specific technique. For example, for KD_{32}, it could be w% sum-z, x% sum-nz, y% dr-z, and z% dr-nz ($w + x + y + z = 100$).

resource description type: In the very end, the resource description instances are all bit vectors. To save storage space, the bit vectors can be of different resource description types. For non-quadtree-utilizing approaches, the possible types are a zipped summary (*sum-z*), a non-zipped summary (*sum-nz*), a zipped direct representation (*dr-z*), or a non-zipped direct representation (*dr-nz*). For quadtree-utilizing approaches, the options are a zipped and LQ-encoded summary (*lq-z*), a non-zipped and LQ-encoded summary (*lq-nz*), a zipped and CLBQ-encoded summary (*cblq-z*), a non-zipped and CBLQ-encoded summary (*cblq-nz*), *dr-z*, or *dr-nz*.

resource size: The number of data points a resource administers. The more (less) data points, the bigger (smaller) the resource. Also see big resource and small resource.

rfc: Resource fraction contacted to process a query completely. Usually, it refers to the average resource fraction contacted over a set of queries. In corresponding contexts, it can also be used to denote the descriptive statistic values (i.e. mean rfc, min rfc, max rfc, median rfc, and so on) over a set of queries, or the resource fraction contacted for a single query and a specific technique.

selectivity: See effectivity.

Skyline: The Skyline of an approach is forged of an approach's set of non-dominated parameterizations. The dominance criterion is assessed with respect to the two 'dimensions' rds value and rfc value which result for the specific parameterizations of an approach. Thus, a point p_x representing a parameterization x dominates a point p_y representing a parameterization y if ($p_x.rfc \leq p_y.rfc$ and $p_x.rds < p_y.rds$), or ($p_x.rfc < p_y.rfc$ and $p_x.rds \leq p_y.rds$).

Skyline parameterization: A specific parameterization of an approach which is part of the approach's Skyline (i.e. it is not dominated by any other parameterization of this approach).

small resource: A resource which administers only few data points.

smallest indexable spatial unit: The smallest spatial area a summarization approach can describe. For some approaches, it is an 'infinitely' small area (corresponding to a point, such as for the approaches utilizing full-precision rectangles) while other approaches are only able to index areas with a surface area greater than 0 (such as quantization-based approaches or pure space partitioning approaches).

spatial scattering of a leaf node: Maximum distance between any pair of data points stored in the leaf node of an R-tree. The greater the maximum distance, the greater the spatial scattering.

spatially narrow resource: Resource whose data points are all located in a very closely confined region of the data space.

spatially spread resource: Resource whose data points are scattered far across the data space.

summarization approach: Specific approach to describe a set of spatial data points by one or several indexing areas.

summary: See summary-represented resource. The summary is the instance depicting the information on the summary-represented resource's spatial footprint.

summary-like R-tree: R-tree that has been built by utilizing the respective technique's summaries. Storage space limitations are considered in summary-like R-trees, i.e. the internal nodes' storage space capacity is restricted to 4,096 B. Consequently, the R-tree structures of different summary-like R-trees (i.e. the assignment of child nodes to parent nodes and of data points to leaf nodes) vary.

summary-like MBR/MBRQT^{c,a}/QTMBR$^{b}_{c,a}$ R-tree: Summary-like R-tree in which MBR/MBRQT^{c,a}/QTMBR$^{b}_{c,a}$ summaries describe the spatial footprints of nodes.

summary-represented resource: The spatial content of a resource is represented in an 'aggregated' form by one or several geometrically delineated areas containing all of a resource's data points. It is not known where the data points are exactly located but only that they are somewhere within these indexed areas. This allows for the calculation of upper and lower bound distances with respect to the distance between the spatial footprint of a resource and specific data points. The summary representation is one of the two subtypes of a resource description.

summary description type: The set of resource description types without the direct representation options. Hence, for quadtree-utilizing approaches, the available summary description types are *lq-z*, *lq-nz*, *cblq-z*, and *cblq-nz*. For non-quadtree-utilizing approaches, it is *sum-z* and *sum-nz* (also see resource description type).

summary size: Amount of storage space required for a concrete instance of a summary.

surface area: The amount of area covered by the surface of a geometrically delineated form. For example, an MBR of 10 dsu width and 10 dsu height has a surface area of $10\ \text{dsu} \cdot 10\ \text{dsu} = 100\ \text{dsu}^2$.

T_5 scenario: Evaluation scenario in which the T collection's data points are inserted into R-trees which have a leaf node capacity of 5 data points.

T_25 scenario: Evaluation scenario in which the T collection's data points are inserted into R-trees which have a leaf node capacity of 25 data points.

technique: A specific approach/parameter-combination. For example, KDMBR$^{b}_{n}$ is the general approach while KDMBR$^{3}_{32}$ is a technique with its parameters n set to 32 and b set to 3.

traditional R-tree: An R-tree which uses MBR summaries to describe the spatial footprints of its nodes. It does not necessarily have to use the algorithms of Guttman's `classical R-tree` but might use algorithms introduced with diverse R-tree variants such as the R*-tree.

tree-based quadtree/k-d-tree: Space partition based on a tree structure (either a quadtree structure or a k-d-tree structure) for which the splitting hyperplanes are obtained from the data points located in the subspace to split. Consequently, the resulting cells of the quadtree/k-d space decomposition might be arbitrarily-sized. It is the counterpart of the `trie-based quadtree/k-d-tree`.

trie-based quadtree/k-d-tree: Space decomposition based on a tree structure (either a quadtree structure or a k-d-tree structure) for which the subspace to split is always divided into regular parts (four for the `quadtree`, two for the `k-d-tree`). Consequently, the resulting cells of the quadtree/k-d space decomposition are equal-sized. It is the counterpart of the `tree-based quadtree/k-d-tree`.

upper-level node: R-tree node which is at an upper level of the R-tree's logical, hierarchical structure (i.e. closer towards the root node level). The level indices are reverse to the logical structure, i.e. the root node (which is at the uppermost level of the R-tree) has the level index 0 while the leaf nodes (which are at the lowest level of the R-tree) have the greatest level index.

Voronoi-based approaches: Approaches which rely on a Voronoi diagram as description base. They can be approaches which rely solely on Voronoi diagrams ($\mathrm{UFS}_{n,cc}$), or hybrid approaches which utilize Voronoi diagrams as their description base ($\mathrm{DFS}^{b}_{n,cc}$).

zero volume node: A node in an R-tree whose `summary` indexes (an) `area`(s) with a total `surface area` of 0 dsu^2.

Part I:

Background and Problem Description

1. INTRODUCTION

This thesis is concerned with the effective and efficient summarization of sets of two-dimensional spatial point data as a means of describing 'resources', and its utilization for the targeted selection of resources in different application scenarios. The term *efficient* refers to the storage-space-efficient summarization of these data point sets while the term *effective* relates to the spatial accuracy of the summarizations. Obviously, both aspects are conflicting, and the diverse application scenarios also require additional tasks complementing the summarization. In this section, the fundamentals of the thesis are outlined. Therefore, section 1.1 starts with a general motivation of this work. In section 1.2, the two application scenarios for which our resource description approaches are examined in this thesis are presented. Afterwards, the general problem description is showcased in section 1.3. In section 1.4, the thesis objectives are depicted. Finally, section 1.5 presents the thesis outline.

1.1. Motivation

"Space is everywhere". This well-worn phrase is a truism. As a subject of the real world, every human being naturally moves in space. Space is his or her environment, and this environment is populated with objects. Each of these objects has properties that include spatial characteristics: Position, dimension, spatial orientation, and so on [van der Zee and Scholten 2014, p. 6]. These spatial characteristics are integral aspects of every object. Space is everywhere, indeed, and so are spatial problems [Singh 2018, p. xvii].
Spatial problems are most often associated with spatial data, obviously. Both are not new. In second-century Egypt, Claudius Ptolemy experimented with spatial data when he was attempting to depict the world on his map (see Figure 1). Similarly, early astronomers used spatial data when they tried to create a celestial map [Fotheringham et al. 2000, p. 15]. The computational facilities at their disposal were rather rudimentary, though. Today, the digital revolution enables to process data of several orders of magnitude more. At the same time, technical progress leads to an ever faster generation of ever larger quantities of spatial data which are used in a wide variety of applications.

Oftentimes, these applications are visualizations of spatial data as they naturally lend themselves for this purpose. For example, the "Sunrise around the World" visualization[3] is a time-lapsed dot density map which shows geotagged Twitter tweets containing the word 'sunrise' (in different languages) around the world. The tweets have been captured on April 6,

[2]Source: https://upload.wikimedia.org/wikipedia/commons/2/23/PtolemyWorldMap.jpg, last visit: 13.07.2018.

[3]See http://cartodb.s3.amazonaws.com/static_vizz/sunrise.html, last visit: 12.07.2018.

Fig. 1: Ptolemy's world map.[2]

2014. The application shows the 'tidal wave' of tweets moving across the world map in synchronization with the sunrise (see Figure 2).
There are also more serious use cases. For example, there have been two massive accidents on the Dubai Highway (E10) in Abu Dhabi involving over 200 (March 2008) respectively 130 vehicles (April 2011). These accidents were caused by dense fog banks which severely restricted vision. Abu Dhabi is a smart city with a well integrated transport infrastructure in which a large number of sensors generate huge amounts of spatial data. To solve the collision problem, a system has been developed that collects and analyzes the data, and sends real-time warnings (via various channels) to drivers which are about to enter areas with poor visibility [GW 2016].
The omnipresence of spatial data also applies to areas from which one does not suspect it at first. In 2008, the Nobel Prize in Economics was awarded to Paul Krugman for his work which explains why some countries dominate the international trade. Now, what does this have to do with spatial data? The prize was given to Krugman on the basis of three papers. The first two dealt with international trade, in particular intra-industrial trade. The last paper extends the analysis to the spatial allocation of economic activities, making it the central model of the new economic geography literature [Brakman and Garretsen 2009, p. 2]. Without being able to judge whether this heavily discussed decision of the Nobel Prize Committee was appro-

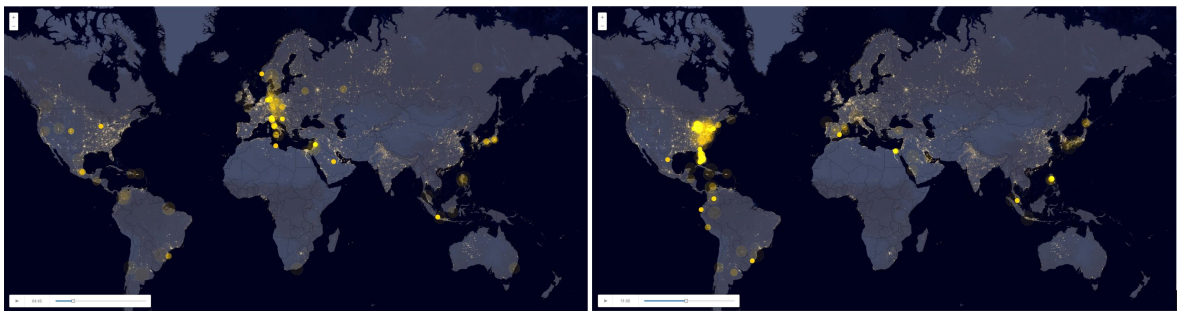


Fig. 2: 'Sunrise around the World' visualization of geotagged Twitter tweets containing the word 'sunrise'. The visualization is time-lapsed and depicts the tweets' locations at different points in time.

priate or not, once more, this shows the enormous importance of spatially related phenomena and processes.
Hence, not only space but also spatial data is (almost) everywhere. But what exactly is spatial data, and how do the terms geographic data and geospatial data relate? In this thesis, we follow the terminology introduced by Bhatta (see [Bhatta 2011]):[4] *Geographic* data is data for which the frame is the surface of the Earth. *Spatial* data has a broader meaning as 'spatial' is 'pertaining to space' (any space, not only the Earth's surface) and thus encompasses the term geographic. Accordingly, geographic data is a subset of spatial data. The often-used term *geospatial* typically refers to spatial data which is related to Earth and therefore conforms to the term geographic, although spatial and geospatial are oftentimes used interchangeably.
A relative of "space is everywhere" is the phrase "spatial is special". The justifications for this claim often refer to processing-relevant properties (e.g. the possibly complex structure of spatial data, or the more expensive spatial operations in comparison to standard relational operations [Gaede and Günther 1998, ch. 2.1]) or analysis-relevant properties (e.g. spatial dependence or spatial error [Anselin 1989]) of the spatial data. In any case, special properties of spatial data are the diversity of data sources and the enormous amounts of data. A *small selection* of spatial data sources includes the following [Wilson and Fotheringham 2008, p. 3]:

— censuses of population (recording information on each individual in each household and on the household itself),
— customer databases of retail-related companies,
— traffic flow monitoring (along streets, intersections, and so on),
— LiDAR (Light Detection And Ranging, low pass fly-overs by plane),
— digital elevation models captured via satellites, and
— data generated from consumer devices equipped with GPS sensors such as smartphones or cameras.

The amounts of spatial data generated both professionally as well as privately are massively increasing. As early as 2009, the McKinsey Global Institute reported that the pool of personal location data is in the range of 1 petabyte (PB) and growing at a rate of 20% per year. In this estimation, the data from RFID sensors and data stored in private archives was not even included [Dasgupta 2013]. In 2012, Google generated about 25 PB of data every day of which a significant portion falls into the spatiotemporal domain [Vatsavai et al. 2012, p. 2]—which includes the spatial property of the data, obviously. For the year 2015, Eldawy and Mokbel reported that for example space telescopes generated up to 150 gigabyte (GB) of weekly spatial data or 10 million geotagged tweets were issued from Twitter everyday (by assuming that about 2% of the daily tweets are geotagged) [Eldawy

[4] Also see http://basudebbhatta.blogspot.de/2010/02/spatial-and-geospatial.html, last visit: 13.07.2018.

and Mokbel 2016, p. 1]. Lesczynski and Crampton note it is estimated that up to 80% of 'big data' is "spatial" insofar as it is characterized by some locational component—be it spatial coordinates, geographical metadata, an associated street address, or the content of the data itself making a reference to a place in the physical space [Leszczynski and Crampton 2016, p. 1]. Regardless of the concrete amounts of spatial data produced nowadays, one thing is certain: The pool of available spatial data is very large, and the amount of spatial data generated per day is constantly increasing.
Generally, there are two broad categories of spatial data: raster data and vector data. For raster data, the space is partitioned at a fixed resolution, i.e. a fixed cell size is defined. In contrast, vector data is basically comprised of coordinates. Examples for vector data are polygons, lines, or points. In this work, we are specifically interested in spatial vector point data.
The typical processing chain of spatial data is extensive. First, the data has to be collected, for example from the sources listed above. Then, the data must be stored and transferred. This can be challenging given the potentially enormous amounts of data. Usually, the data must then be converted into useful information for which a data visualization and/or a data analysis have to be conducted. A depiction of the usual spatial data processing chain can be found in [Wilson and Fotheringham 2008, p. 2].
When processing sets of point data, searching for data points that fulfill certain properties is a recurring process. Despite the computing power available today, an *efficient* search in a large point data collection cannot take place if each point has to be considered individually in a sequential scan. The data has to be organized in an appropriate way so that only (optimally small) subsets of the total point data set must be considered for suchlike searches. In this context, the concept of *resource description and selection* is a useful paradigm to cope with the arising challenges. Similar to spatial data problems, this concept reaches back to days long before the invention of computers. As an example, consider the Library of Alexandria: It was founded by Ptolemy I Soter (not to be confused with Claudius Ptolemy) in the 3rd century B.C. For the number of papyrus scrolls maintained in the library, the estimates range from 40,000 to 400,000.[5] Irrespective of the concrete number, at the time, it was a huge amount of scrolls to be maintained. The scrolls were grouped together by subject and stored in bins. Each bin was bearing a label with painted tablets hanging over the stored papyri. These *Pinakes* (Greek for tables) gave bibliographical information for each roll. A typical entry provided—among other things—information on the author (name, birthplace, educational background, and so on), the first line of the work, and a summary of its contents [Phillips 2010].
The organization of spatial point data into 'resources' (similar to the bins in the Alexandria Library) and the concise, yet meaningful description of these resources' contents (similar to the *Pinakes*) is a generic, abstract ap-

[5] See https://en.wikipedia.org/wiki/Library_of_Alexandria, last visit: 13.07.2018.

proach for efficient and effective search procedures that can be used in various application scenarios. The two scenarios considered in this work are described in the following.

1.2. Investigated Application Scenarios

This section presents the application scenarios for which our spatial resource descriptions are developed. In general, the investigated application scenarios involve searching a set of spatial point data for points that fulfill a specific search criterion. Our understanding of a 'resource' in this work is that a resource is an abstract entity that administers spatial data point sets. There are many different concrete forms a resource can take. For example, it could be a peer in a peer-to-peer (P2P) system (within the context of a *distributed application scenario*), or a node in the tree structure of a multidimensional data structure (within the context of a *centralized application scenario*). In any case, the resource descriptions depict the spatial footprint of a resource in such way that while searching, it can be decided whether the resource has to be considered further or not.
At this point we want to make an important definition and differentiation of two terms: *summarization approach* and *resource description approach*. The term summarization refers to the aggregated description of a spatial data point set by geometrically delineated regions. For these aggregations, different approaches are developed in this thesis, the summarization approaches. It can happen that the summary of a resource requires more storage space than the coordinates of its data points themselves. Consequently, in such a case, a summary would be an inadequate description for the resource as in addition, it is only a spatial approximation (whereas the data points provide the exact spatial information). Under such circumstances, a resource might be represented by the coordinates of the data points which we call direct representation. Thus, a specific resource description approach includes a specific summarization approach and the possible replacement of the resulting summary by the direct representation. This will be explained explicitly in the course of the work at the appropriate places but we want to clarify these terms at the beginning of the work in order to avoid confusion regarding their distinction.[6]
The two application scenarios mentioned are also those that are considered in this thesis. Our focus is on the distributed application scenario, i.e. the summarization approaches for resource description purposes are primarily developed for this use case. It is outlined in section 1.2.1. As a second proof-of-concept, selected approaches are utilized to describe the spatial footprints of the nodes in one of the most predominant tree-based multidi-

[6] From now, the term 'resource description' is used as an umbrella term including both the description of a resource's spatial content by a summary or the direct representation by the coordinates of its data points (in case this direct representation does not require more storage space than the summary representation).

mensional data structures—the R-tree [Guttman 1984]. This application scenario is depicted in section 1.2.2.

1.2.1. DISTRIBUTED APPLICATION SCENARIO. The distributed application scenario assumes a search environment in which users administer geotagged media items in personal media archives [Henrich and Blank 2010, p. 22f]. The media items are stored locally on a user's personal device (such as smartphones, desktop computers, or the like). The personal media archives are organized in a peer-to-peer (P2P) system, and the users can search the P2P system for media items. The users' personal devices represent the peers (or *resources*) of the P2P network. Idle computing power in times of a user's inactivity can be utilized to maintain, analyze, and enrich the corresponding resource's media items. For the purpose of sharing and collaboration, users can issue queries to retrieve media items from the P2P network.

In order to facilitate the retrieval, a media item is described by four criteria: 1) its textual content, 2) low-level, media-type-specific features, 3) timestamps, and 4) its spatial footprint. Assuming the media items to be for example images, these criteria could be 1) an image's tags, 2) a histogram of the color distribution in the image, 3) the time and date when the image has been taken, and 4) the location the image has been taken specified by latitude/longitude-coordinates (lat/long-coordinates). To allow for the efficient and effective selection of the media archives administering the relevant media items when processing a query, a media archive's content is summarized by four corresponding resource descriptions (see Figure 3). For these, the features of the media items are aggregated criterion-wise to represent the resource with regard to the specific criterion. For example, the spatial footprints of an archive's media items are aggregated into a spatial description for the entire archive. Then, the resources can be queried by means of a separate query for each criterion. The combined, final query result is composed of the four partial results. An example for the verbal formulation of such an overall query would be "retrieve all images from the *Pico de las Nieves* on Gran Canaria showing an azure sky which have been taken in September 2016".

In this thesis, we focus on the spatial aspects of suchlike searches: How can the spatial footprints of resources be described efficiently such that the resources administering the relevant media items with regard to the spatial criterion of those queries can be selected effectively when processing the query? Within our assumed distributed application scenario, the distribution of data points to the resources is determined by origin, i.e. it is not dirigible by the system. Consequently, the requirements on the spa-

[7] Of course, also in a distributed environment, the resources can in principle be organized in a way (e.g. there might be a set of 'super resources') that from a conceptual point of view, hierarchies of resources are built. However, for our work, we assume that there are no such hierarchies and that all resources are 'equitable'.

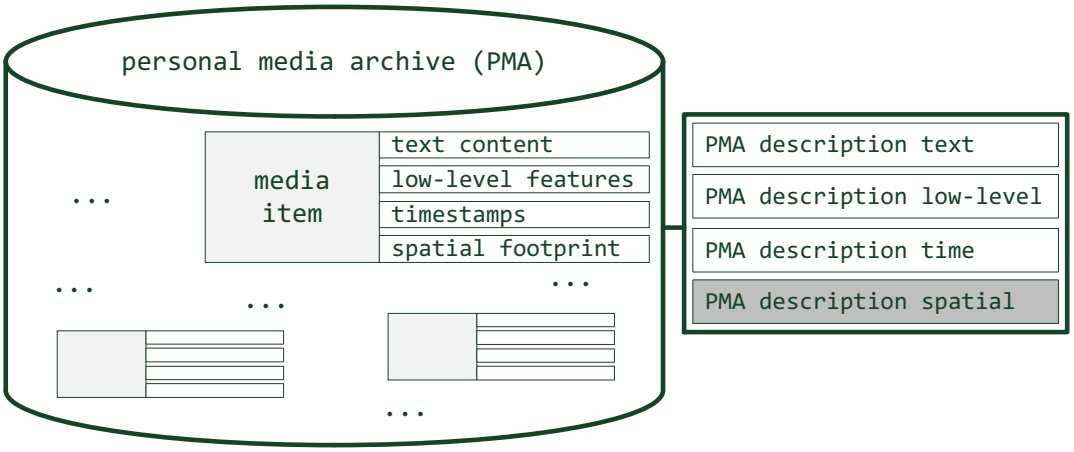


Fig. 3: Criteria for the resource selection in the distributed search scenario. Our focus is on the spatial aspects. The depiction is based on Figure 1 of [Henrich and Blank 2010, p. 22].

tial accuracy of the resource descriptions are very high because the data points cannot be arranged in the way it is most convenient for a straightforward description. From an organizational view, all resources are at the same hierarchical level, i.e. there is no hierarchy of resources (or the like)[7] that can be used by a query processor to exclude entire groups of resources from the search at once (see Figure 4): the relevance of *each* resource has to be assessed on basis of the resource description. A further assumption for the distributed application scenario is that the P2P system constituting the distributed database management system is implemented as a Java application.

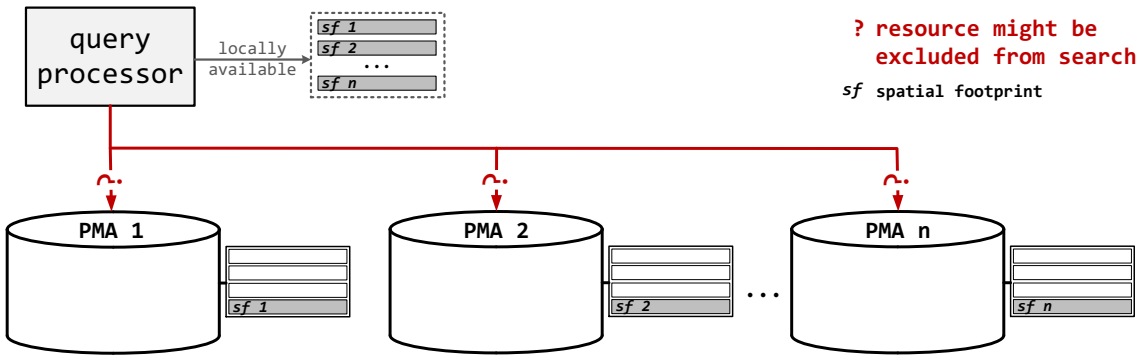


Fig. 4: Depiction of the conceptual search in the distributed application scenario.

1.2.2. CENTRALIZED APPLICATION SCENARIO. In the centralized application scenario, it is assumed that the media items are administered on a single machine and are organized by means of a hierarchical, balanced tree structure such as an R-tree[8] which administers (references to)

[8]The R-tree is discussed in more detail in section 2.4.3 and section 10.1.

the media items in its leaf nodes. The 'single machine' might be a peer of the distributed application scenario's P2P network[9] or a large, centralized multimedia database management system which allows for the retrieval of media items by their spatial properties. The organization of the hierarchical tree structure is based on the spatial attributes of the media items. It can act both as a primary index as well as a secondary index. In a primary index, the media items are directly administered in the leaf nodes as payload data and therefore, the tree structure decides on the storage location of the media items. In a secondary index, the leaf nodes only contain the spatial properties (i.e. the point coordinates) of the media items alongside a reference to the actual storage locations of the media items on disk (which are decided by the structure which is applied as primary index).
Regardless of whether it is a primary or secondary index, the organization into the tree structure serves to determine the relevant media objects with respect to a spatial query while considering as few nodes as possible. In such a tree structure, the single nodes can be regarded as resources. Internal nodes serve as 'signposts' to guide the search to the leaf nodes with the relevant media items. The spatial footprint of an internal node has to cover the spatial footprints of all the media items administered in its subtree. Paths that lead to leaf nodes with only non-relevant media items should be excluded from the search as early as possible. To do this as effectively as possible, for each node, an accurate description of the aggregated spatial footprints of the media items managed in the corresponding subtree or leaf node is required: The more accurate these resource descriptions, the better the 'signposts', and the more efficient the search.
In contrast to the distributed application scenario, the centralized scenario allows for a 'proactive' assignment of media items (respectively their data points) to nodes. Therefore, there is control over which nodes or subtrees administer which data points. Consequently, the requirements on the spatial accuracy of the resource descriptions are not quite as high as for the distributed application scenario because the assignment of data points to resources can be arranged such that it is fitting for the utilized resource description approach. Nevertheless, better signposts will still lead to better results in such an environment. The organization of the resources into nodes of a hierarchical tree structure results in that not necessarily all re-

[9]Note that in this case, a resource which is capable of issuing a query would maintain two types of indexes (also see [Blank 2015, p. 61]): A *local index* for the local query processing (such as an R-tree for point data), and a *resource index* to accelerate the resource selection process (i.e. the selection of the potentially relevant peers). However, this thesis does *not* explicitly address the layout of the *resource index*. The resource index could for example be designed as follows: Each spatial footprint which is indexed by a resource description (i.e. each area indexed by a summary or each data point of a direct representation) is inserted into an R-tree which maintains *region data*. Alongside each spatial footprint, the R-tree additionally stores the ID of the resource the spatial footprint belongs to. In combination with an adequate ranking algorithm, this resource index can then be used for an accelerated ranking of the resources.

sources have to be assessed directly for their relevance (as it is the case in the distributed application scenario, see Figure 4 again). Groups of resources can be collectively excluded from search at once—in case (the subtree of) a higher-level node is excluded from search (see Figure 5).
Multidimensional data structures such as the R-tree are usually implemented in comparatively lower-level programming languages such as C++ which are closer to the machine level than Java. Hence, it is assumed that the R-tree is implemented in C++. Consequently, our Java implementation of the R-tree simulates a C++ environment.

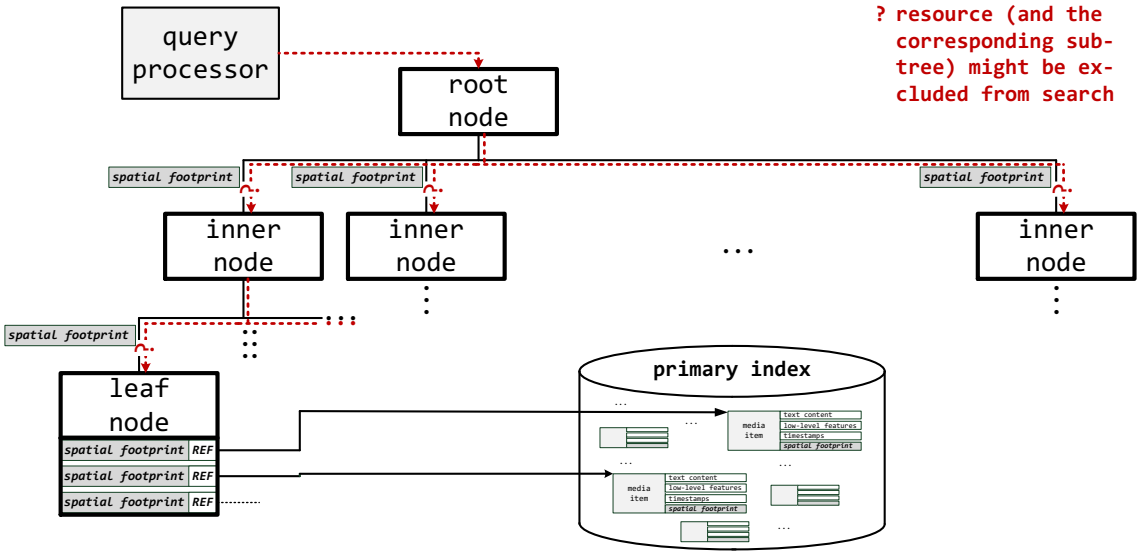


Fig. 5: Depiction of the conceptual search in the centralized application scenario. It showcases a situation in which the hierarchical tree structure is used as a secondary index. In a primary index, the media items would be administered directly in the leaf nodes. Note that in this abstracted depiction, a *logical* assignment of the spatial footprints to their corresponding nodes is made.

1.3. Problem Description

In this section, the concept of resource description and selection—which is part of the thesis title—is outlined alongside some background information. In general, it provides a suitable framework for the definition of the tasks to solve in our application scenarios. In an abstract view, both of them assume a set of resources to provide geotagged media items or generally documents accessible in a search system. As just discussed in section 1.2, the search system can be of two types:

— A distributed system in which the documents are spread over the resources (e.g. peers of a P2P network).
— A centralized system in which (references to) the documents are organized in a hierarchical tree structure consisting of resources (nodes).

As mentioned in section 1.2, the distributed application scenario is the main focus of our work. Consequently, the following explanations are such that they fit the distributed application scenario. Nevertheless, from an abstract point of view, they are also (partly) transferable to the centralized scenario.
In the distributed application scenario, for the documents provided by the resources, it is assumed that the users are interested in documents (such as images or short text messages) which are at or near a spatial location specified by the user. Thus, not the *contentual* but the *spatial* properties define the 'relevance' of a document with regard to a query. The single resources are independent of each other and the documents administered by a resource cannot be copied to other resources. Thus, each resource corresponds to a database of its own. We further assume that there is no central instance organizing the distribution of the documents to the resources—the origin of the document determines its assignment to a resource.[10]
Search engines for the Web—or large networks in general—are usually based on the *single database* model. This model is inappropriate for our distributed scenario since it relies on the documents of a network to be copied and stored in a centralized database for indexing and searching. Our foundation is the *multi-database* model which is suited for proprietary or carefully controlled content because there is no requirement of copying the documents of each single database to a large centralized database [Callan 2000, p. 128]. Instead, it explicitly models the existence of multiple databases and potentially is also scalable to a large number of databases. Brief descriptions of each database are created and a database selection service uses these *resource descriptions* to identify the database(s) that contain the relevant documents. Due to its complexity, the multi-database model introduces some additional problems which have been divided into three basic tasks in the literature for distributed information retrieval (distributed IR).[11] See [Callan 2000] or [Shokouhi and Si 2011] for further reading. The three basic tasks are the following:

— **Resource Description Problem**: A brief description of the contents of a database [Callan 2000, p. 128]. The information provided[12] should facilitate an efficient selection of the resources maintaining the relevant documents [Shokouhi and Si 2011, p. 1].
— **Resource Selection Problem**: Based on the resource descriptions, a subset of resources that (most likely) contain the relevant documents is selected to be contacted while processing the query [Shokouhi and Si 2011, p. 1]. Contacting resources without relevant documents should be avoided. The resource selection task is strongly coherent with the

[10] Obviously, the last few sentences do not apply to the centralized scenario.
[11] Distributed IR is also often termed federated search.
[12] We assume a cooperative environment, i.e. the resources provide their descriptions themselves. In an uncooperative environment, the description of a resource could be computed on basis of the queries the resource has issued [Blank 2015, p. 6].

resource description task such that the term ‘resource selection’ often refers to both tasks [Blank 2015, p. 7].

— **Result Merging Problem**: After the query has been submitted to the selected subset of resources, the returning results have to be integrated and merged into an overall result [Shokouhi and Si 2011, p. 1].

In the literature for distributed IR, these tasks are often described with regard to the concept of a user’s specific information need [Callan 2000; Shokouhi and Si 2011]. The rather abstract concept of a user’s information need[13] is not really existent in our scenarios since the relevance of a document with regard to a query—both represented as a spatial data point—is simply defined by the spatial proximity of the respective data points. Nevertheless, the distributed IR tasks are a useful framework to adhere to the challenges arising in *both* of our application scenarios.
Furthermore, Lu ([Lu 2007]) showed that generally, the search task in a P2P network—such as it is assumed in our distributed scenario—is closely related to the topic of federated search. In a P2P network, problems similar to the federated search challenges have to be addressed: Contents of resources (or more specifically peer nodes) have to be represented in a useful manner (resource description), the routing of queries to relevant resources is important (resource selection), and the results of the different resources have to be combined (result merging) [Shokouhi and Si 2011, p. 10f].

1.4. Thesis Objectives

In this section, the thesis objectives are outlined. In general, in basically all domains, the predominant means of summarizing a set of low-dimensional spatial data points is the Minimum Bounding Rectangle (MBR). Despite many advantages, the MBR suffers from serious drawbacks with regard to its spatial accuracy. To conquer these drawbacks by applying more suitable summarization approaches is the subject of this work. It is therefore also a form of basic research. The applicability of the newly developed approaches is to be checked for different areas of application. Therefore, the overall objectives of the thesis are the following:

❶ Research from Blank, Henrich, and Kufer ([Henrich and Blank 2010; Blank and Henrich 2012; Kufer et al. 2012]) up to the year 2012 has addressed the development of appropriate summarization approaches for two-dimensional spatial point data for resource description and selection purposes which significantly improve the MBR in the context of the distributed application scenario. *The major objective of this thesis is to develop novel summarization approaches which significantly improve these existing approaches. There are several properties that can be im-*

[13] Wissbrock defines that the information need refers to the amount of absence information which is necessary for a user to reach his or her goals in a particular situation while several assumptions such as ‘the user may not know what exactly his information need is’ hold [Wissbrock 2004, p. 80].

proved. Most importantly, improvements shall be achieved in terms of the resource selection performance or the resource description efficiency, ideally for both properties simultaneously. The resource selection performance is measured by the resource fraction contacted in order to answer a query. The resource description efficiency is measured by the average amount of bytes spent to describe a resource. The overall aim of such a resource description and selection approach is to minimize the communication costs in the network—which is a cost measure frequently applied in the research on distributed query processing [Blank 2015, p. 10].

❷ Several summarization approaches developed for the distributed application scenario are also suited for application in the centralized application scenario. The field of multidimensional data structures for indexing spatial data—which is concerned with the development of appropriate access methods—has been the subject of intensive research over a period of 30 years. This means that the 'low-hanging fruits' have already been harvested a long time ago. Nevertheless, the descriptions of the spatial contents maintained in a node (respectively its subtree) might still offer room for improvement. A suitable target for the integration of our summarization approaches is the predominant multidimensional data structure—the R-tree. It originally utilizes MBRs to describe the spatial footprint of a node. *Therefore, a second thesis objective is to integrate suitable summarization approaches into an R-tree, and to evaluate the potential for improvement by doing so.*

1.5. Thesis Outline

Generally, the thesis is organized into four parts. The first part introduces the topic of this work and provides comprehensive information on related work. The second part is concerned with the distributed application scenario while the third part deals with the centralized application scenario. In the fourth part, we present a brief conclusion of the thesis at hand. In detail, this work is structured into 14 sections. The thesis outline completes section 1. The remainder of this thesis is structured as follows:

— **Section 2** introduces the related work for the design of resource descriptions in the given application scenarios. Furthermore, some additional topics related to the overall context of this thesis are outlined. Concretely, section 2.1 to section 2.4 are dedicated to conceivable approaches for the design of resource descriptions. Generally, very simple strategies are discussed at first which then increasingly become more complex. The utilization of compression techniques to reduce the storage space footprint of a resource description is discussed in section 2.1. Afterwards, the usage of diverse, arbitrarily complex bounding geometric forms for describing a set of data points is showcased in section 2.2. The potentially high storage space and computational requirements introduced by complex forms can be mitigated by decomposing these complex forms into a set

of simpler forms. This is the topic of section 2.3. In order to keep the indexed surface areas as small as possible, a division of the data point set to describe into groups and then adequately describing each group is an appropriate strategy. Related research areas are clustering, computational geometry, and multidimensional data structures. The division into groups is discussed in section 2.4. Afterwards, appropriate and common query types for spatial data are outlined in section 2.5. Finally, section 2.6 is concerned with topics which are directly relevant for or at least related to our work but do not fit into one of the previous subsections. This includes linear quadtree encodings, P2P multidimensional indexing methods and P2P overlay networks, metric indexing, and Ptolemaic indexing. With section 2, also the **first part of the thesis** is completed.

— **Section 3** marks the beginning of the **second part of the thesis**. In this part, the distributed application scenario is considered in depth. In section 3, first, some general preliminaries for the further consideration of this scenario are conducted (section 3.1). Since we make heavy use of quadtrees in the summarization approaches we develop, we also need to consider some quadtree-specific preliminaries (section 3.2). An excursus on the inherent imperfections in spatial data (section 3.3) closes section 3.

— **Section 4** outlines the summarization approaches for resource description purposes in detail. This includes the description of their calculation as well as the definition of their storage format. In total, 14 approaches are presented. For a structured presentation, the approaches are classified into three categories: data partitioning approaches (section 4.1), space partitioning approaches (section 4.2), and hybrid approaches (section 4.3). Hence, section 4 is dedicated to the resource description task.

— **Section 5** discusses the subsequent resource selection task. The query type we investigate for the distributed application scenario is the kNN query. The efficient processing of this query type requires a ranking of the resources. Adequate ranking algorithms are outlined in section 5.1. On the basis of the ranking, the actual kNN algorithm can be conducted. A conceptual kNN algorithm for retrieving the *true* k nearest neighbors from a resource network is showcased in section 5.2.

— **Section 6** outlines the general environment of the subsequent evaluation. This includes discussions on how the results are generally assessed (section 6.1), the data collections (section 6.2), the experimental setup (section 6.3), and several adjustments we conduct to simulate a 'real-world-system' as realistically as possible (section 6.4). Due to the great number of approaches, the evaluation is split into two large sections.

— **Section 7** is the first of the evaluation sections of the distributed application scenario. All of the 14 resource description approaches are evaluated for a specific data collection. It starts with a general overview of the results (section 7.1) before the approaches are divided into groups based on similar inherent properties: the MBR-based approaches (section 7.2),

the kd-based approaches (section 7.3), the quadtree-based approaches (section 7.4), and the Voronoi-based approaches (section 7.5). From each of these groups, the one or two most suitable approaches are selected for further consideration in the in-depth evaluation in section 8. Finally, section 7 is concluded with a separate comparison of the pure space partitioning approaches and their simplest hybrid extensions (section 7.6).

— **Section 8** is the second part of the distributed application scenario's evaluation and compares six selected approaches in great depth. The comparison is conducted on the basis of additional data collections, several additionally collected key figures, qualitative analyses, and diverse changes with regard to the assumed 'environment' in which the assessments take place. First, the selected approaches are once again evaluated for the specific data collection already utilized in section 7, but in much more detail (section 8.1). The further evaluation includes the use of additional data collections (section 8.2 and section 8.6) and the variation of the underlying conditions for the diverse data collections (section 8.3, section 8.4, and section 8.5). All these assessments are for exact kNN queries (i.e. the true k nearest neighbors within the resource network are to be determined—which corresponds to an exact similarity search). In section 8.7, the suitability of the selected approaches for an approximate similarity search is evaluated. In section 8.8, two additional properties of the selected approaches are assessed: the achievable query radius reduction in the forefront of the kNN algorithm, and the respective runtimes of the initial resource rankings. Finally, section 8.9 closes with a short summarization of the evaluation's key results and an assessment of compliance with thesis objective ❶. It also represents the completion of the thesis' second part.

— **Section 9** constitutes the beginning of the **thesis' third part**. Here, the motivation for the integration of selected summarization approaches into an R-tree and also the general preliminaries with regard to the assumed centralized application scenario are discussed in brevity.

— **Section 10** is dedicated to the concrete integration of $\text{MBRQT}^{c,a}$ and $\text{QTMBR}^{b}_{c,a}$ summaries into an R-tree. At first, the classical R-tree as outlined by Guttman (which in particular uses MBRs to summarize the spatial contents of nodes) is presented (section 10.1). Afterwards, we describe some necessary modifications we apply to the classical R-tree in order to integrate the $\text{MBRQT}^{c,a}$ respectively $\text{QTMBR}^{b}_{c,a}$ summaries (section 10.2). In the aftermath of an explicit discussion of the general requirements for summaries which are utilized in R-trees (section 10.3), we then proceed to a specification of how to calculate $\text{MBRQT}^{c,a}$ and $\text{QTMBR}^{b}_{c,a}$ summaries when the respective input are other summaries (section 10.4). Suchlike summary calculations are an essential requirement for being able to make appropriate use of our summarization approaches in an R-tree. In this context, also some modifications to the storage formats of the $\text{MBRQT}^{c,a}$ and $\text{QTMBR}^{b}_{c,a}$ summaries are depicted.

Finally, section 10 is concluded with a discussion of the effects and the expectable trade-offs of the integration (section 10.5).

— **Section 11** depicts the algorithms of the query types for which we assess the R-trees in the evaluation. In section 11.1, the range query algorithm is outlined while in section 11.2, the kNN query algorithm is presented.
— **Section 12** consists of a discussion of the evaluation environment. First, it is described how the results are assessed in general (section 12.1), followed by a depiction of the data collections which are used in the evaluation (section 12.2). Finally, a presentation of the experimental setup (section 12.3) concludes the review of the evaluation environment.
— **Section 13** features the extensive evaluation. This evaluation includes different data collections and different environmental prerequisites. At first, the differences that arise for the R-tree construction times when using the diverse summarization approaches are assessed (section 13.1). Then, the structural divergences which result for the different R-trees are evaluated in the structural analysis (section 13.2). Afterwards, the kNN and range query results are showcased and investigated (section 13.3). Finally, section 13.4 concludes the extensive evaluation with a comprehensive summarization of its key findings, an assessment of the degree of achievement of thesis objective ❷, and an overview of starting points for future work.
— **Section 14** constitutes the **fourth and final part of the thesis**. It summarizes the thesis—the concept behind the summarization approaches' utilization, the respective methodology in the two assessed application scenarios, and the respective results—once again in all brevity.

2. RELATED WORK

In this section, the related work for this thesis is gathered. As stated before, the thesis is focused on designing accurate and efficient summaries of spatial data point sets. These summaries are then to be utilized for resource description and selection purposes aiming at the effective and efficient retrieval of spatial point data respectively associated data objects in search systems. Since in both of our application scenarios, we assume that the data objects are geotagged media items, the dimensionality d of the spatial data points associated with the media items is 2. For the following explanations, we only assume that a set of two-dimensional data points is assigned to a resource.

At the beginning of this section, the focus is on how *effective* (allowing for a targeted selection of the relevant resources while safely pruning[14] the other ones) yet *efficient* (low storage space requirements) resource descriptions can be designed. Generally, the spatial footprint of a resource can be described in two fundamentally different ways. The first one is to directly represent the resource, i.e. the spatial 'description' of the resource are just the coordinates of the resource's data points. The second way is to summarize the spatial contents of a resource—which is an aggregated representation: One or several geometrically delineated areas are described which contain all of the resource's data points. Of course, this is only an approximated representation of the resource's spatial footprint—but it allows for treating its data points as a group. Depending on the circumstances, both representational forms of a resource description—the direct representation and the summary representation—are viable.

The most naive approach of a resource description would be to use compression techniques in order to reduce the storage space requirements of the data points' coordinates (i.e. a direct representation). Of course, such an approach would not allow for speeding up the pruning of irrelevant resources since still *every single* data point would have to be tested for its relevance. For this purpose, 'meaningful' (i.e. spatially accurate) summaries of the data point sets have to be employed which—as far as possible—allow for computationally cheap distance and intersection tests. Nevertheless, compression techniques can also be applied to possibly reduce the storage space requirements of these summaries. Relevant compression schemes for both purposes—reducing the storage footprint of a set of data points as well as of a resource summary—are reviewed in section 2.1.

A practicable approach to design meaningful summaries is to cover and bound the set of data points by utilizing appropriate geometric forms which approximate the extents of the 'data point cloud' to describe. These forms can then be used to prune the *entire* set of enclosed data points at once. For choosing suitable bounding volumes, the tightness of the fit and the com-

[14] Pruning refers to excluding the data points of a resource from further consideration on the basis of the lower bound distance calculated for the resource's spatial footprint.

plexity of the forms' geometric shape have to be considered. An overview of conceivable bounding volumes is given in section 2.2.
The more complex the shape of a bounding volume, the more expensive the distance and intersection tests—which can result in considerable computational overhead. One way of mitigating these high computational costs is representing a complex form (like a polygon) by a set of simpler forms (like triangles or rectangles) whose *union* yields the complex form again. Relevant work on this conceptual approach is presented in section 2.3.
As another aspect, despite possibly arbitrary geometric complexity (leading to potentially huge storage space requirements), single bounding volumes might still be a poor fit for a set of data points by eventually containing a lot of dead space[15] in their interior. While the division of a single complex form into several simple forms might mitigate the computational complexity, the dead space issues are not even slightly resolved. A more suitable approach in all respects might be to divide the set of data points into spatially coherent groups and to represent each group by a geometrically delineated area covering the group's data points. This area can be a (simple) enveloping geometric form or a subspace obtained from a partition of the entire data space. Appropriate approaches of this category are available from various fields such as clustering or multidimensional data structures. Suchlike approaches are discussed in section 2.4—which concludes the review of the most relevant work for designing resource summaries for spatial data point sets.
Inarguably, there are still more research areas where ideas and inspirations could be drawn from—like shape fitting (see for example [Yu et al. 2008]) or certain fields of computational approximation algorithms such as random partitioning via shifting (see for example [Har-Peled 2011, ch. 11]). Nevertheless, it is impossible to list and review all conceivable research areas in this work. Furthermore, the diverse fields oftentimes are closely related to or at least connected to the surveyed research areas (with hence very similar solutions), or the focus is on a dimensionality (much) higher than $d = 2$. Therefore, we settle for the research areas surveyed in section 2.2 to section 2.4 which include all the (origins of the) ideas and techniques implemented in the summarization approaches presented in this work.
Following the focus on the design of resource descriptions, in section 2.5, it is discussed which types of queries can generally be supported in a search system for spatial data and which distance metrics are applicable for the implementation of especially (k) nearest neighbor (NN) queries.
Finally, section 2.6 presents an overview of other research fields that are connected to the overall context of this work—from which *some* are extremely relevant for the design of *efficient* resource summaries. Examples

[15] Dead space is space not containing any data points at all [Chakrabarti and Mehrotra 1999, p. 5].

for discussed research areas are linear quadtree encodings or metric space indexing.

2.1. Compressing Data

In the context of our work, there are two possible scopes for the application of compression techniques: the (coordinates of the) data points themselves (i.e. the direct representation of resources) and the resource summaries.
The coordinates of the data points in our application scenario are 32-bit IEEE single precision floating-point numbers. Their data type can either be taken into account by applying specialized floating-point compression schemes; or, it can be ignored by simply using general purpose compression schemes, treating each coordinate as a vector of 32 bits. For the resource summaries, the case is similar. A resource summary can be a set of floating-point numbers—for example describing the extents of a bounding volume—which could be treated either specifically as floating-point data or more generally as a bit vector (also called bitmap). Or, the resource summary is actually already a bit vector which for example captures binary information on the occupation of subspaces (drawn from the data space) with data points (see section 4.2 for examples). Hence, regardless of whether it is the direct or the summarized representation, the data to be considered for compression is either floating-point data or bit vector data.[16] In addition, the subsequent processing of the data must also be kept in mind. Lossy compression does not make sense under the given circumstances, especially for bit vector data. Therefore, we focus on lossless compression. In general, compression schemes are usually byte-based. In the following, compression schemes for bit vectors are discussed first, succeeded by floating-point number compression schemes.

2.1.1. BIT VECTOR COMPRESSION SCHEMES. For compressing bit vectors, the options are to use general purpose compression schemes or specialized schemes (suited for fast further processing) [Wu et al. 2006, p. 3].

General Purpose Compression Schemes. General purpose compression schemes utilize the entropy of the data and usually only aim for the compact storage of data. An example of such a scheme is the run length encoding (RLE) which encodes repeated symbols (i.e. the single bytes) as a pair: the symbol itself and the number of repetitions. Aside from the RLE, it can be distinguished between two general methods: statistical and dictionary coding.
The former are based on a statistical model, namely an alphabet and the probability distribution of a source. An example is the Huffman coding [Huffman 1952] where fixed-length codes are replaced by variable-length codes, assigning shorter code words to more frequently occurring symbols.

[16] Of course, it is also conceivable to develop resource descriptions which for example capture integer values. However, the resource descriptions in this work are all either based on floating-point numbers or bit vectors.

Improved, adaptive Huffman codes exist (see [Shanmugasundaram and Lourdusamy 2011, ch. 2.4]). For the arithmetic coding, the basic idea of Elias is sketched in [Abramson 1963] and later was further improved by for example Rissanen [Rissanen 1976]. Generally, it does not assign a code word to each symbol but replaces a stream of input symbols with a single floating-point number (and therefore encodes several symbols together).
In contrast, the dictionary coding techniques rely on recurring patterns in the data. The basic idea is to replace these repetitions by shorter references to a 'dictionary' containing the original [Shanmugasundaram and Lourdusamy 2011, p. 70]. Concrete dictionary-based compression techniques are usually based on the Lempel-Ziv scheme, originating from two algorithms proposed by Ziv and Lempel in 1977 [Ziv and Lempel 1977] and 1978 [Ziv and Lempel 1978]. Based on these, the concrete compression algorithms can be divided into two families: those derived from LZ77 (LZ77, LZSS, LZH, LZB) and those derived from LZ78 (LZ78, LZW, LZFG) [Shanmugasundaram and Lourdusamy 2011, p. 68].
In general, any practical dictionary coding scheme can be outperformed by a related statistical one [Bell et al. 1989, p. 4]. Therefore, statistical coding is the area to look for better compression. Dictionary-based schemes on the other hand are usually faster [Burrows and Wheeler 1994, p. 1]. In [Burrows and Wheeler 1994], a technique is presented for which the authors claim it achieves compression ratios within ~1% of statistical coding techniques but at speeds comparable to dictionary-coding techniques. For further reading on general purpose compression schemes, we refer to [Shanmugasundaram and Lourdusamy 2011] and [Bell et al. 1989].

Specialized Compression Schemes. The general purpose schemes are efficient in reducing file sizes but performing logical operations is usually much slower than on uncompressed bit maps (since compressed bit maps have to be explicitly unpacked before any operation). Hence, specialized schemes have been proposed for improving the performance of bitwise logical operations while offering good compression ratios at the same time. The operations can be applied directly on the compressed bitmaps, i.e. there is no need for unpacking. Examples are the byte-aligned bitmap code (BBC) [Antoshenkov 1995] which is byte-based, or the word-aligned hybrid run-length code (WAH) [Wu et al. 2001] which takes into account that modern computers read one word of for example 32 bit at a time and can perform operations on whole words.
Subsequent efforts can be classified into a) enhanced versions of the WAH and b) approaches using arbitrary unit segment lengths for compression (instead of steadily $(w-1)$ bit segments like in the WAH where w is the word length of for example 32 bit) [Guzun et al. 2014, p. 484f]. Guzun et al. present a variable length WAH along with a general framework for bitmap indices which is designed to negotiate the trade-offs—compression ratio and faster operations [Wu et al. 2006, p. 4]—of these two classes. All specialized schemes are based on RLE. Generally, smaller encoding lengths

provide better compression but require more decoding during execution [Guzun et al. 2014, p. 484f].
The direct compression of individual bitmaps is only one way to reduce the storage space requirements of bit vector data. Another strategy is to first reduce the number of bitmaps. For example, bit-sliced indices create a bitmap for each binary digit (i.e. for a set of 32-bit single precision floating-point format numbers, 32 bitmaps would be created). They are also referred to as binary encoding schemes [Wu et al. 2006, p. 4]. Examples are given in [O'Neil and Quass 1997], [Chan and Ioannidis 1999], or [Koudas 2000]. The bitmaps of the bit-sliced indices can then be compressed with regular compression schemes.

2.1.2. FLOATING-POINT DATA COMPRESSION SCHEMES. There are also specialized compression algorithms specifically targeted at floating-point data, distinguishable in lossy and lossless compression schemes.[17] In [Lindstrom 2014], a very high throughput, high compression ratio (and therefore lossy but optionally error-bounded) algorithm is presented. An open-source implementation termed ZFP which is loosely based on this algorithm is available. It is claimed that ZFP is often orders of magnitude more accurate and many times faster than other lossy compressors.[18] We do not go into further detail for lossy compressors at this point and refer to [Lindstrom 2014, ch. 2.2] for an overview and further references.
Regarding the lossless schemes, there are some designed for 32-bit IEEE single precision floating-point numbers while others are suited for 64-bit IEEE double precision floating-point numbers or even variable-precision floating-point data. Isenburg et al. describe a lossless compression scheme for 32-bit floating-point numbers which uses predictive coding [Isenburg et al. 2004]. The predicted and actual floating-point values are broken into sign, exponent, and mantissa, and their corrections are separately compressed with context-based arithmetic coding. The exponent is used to switch between different arithmetic contexts since the quality of predictions varies with the exponent. Generally, for using this compression scheme, the points have to be put in order and a suitable prediction scheme has to be applied. The authors claim their algorithm works with any prediction scheme. An overview of prediction schemes is given in [Isenburg et al. 2004, p. 3f]. Lindstrom and Isenburg present an evolution of the algorithm described in [Isenburg et al. 2004]. It is termed FPZIP for which the authors claim they have achieved a well-balanced trade-off between computational speed and data reduction [Lindstrom and Isenburg 2006, p. 1245]. In contrast to the approach presented in [Isenburg et al. 2004], the new scheme is more general as it compresses floating-point and integer

[17] In contrast to bit vector compression, for floating-point compression, lossy compression could *potentially* be applicable in our scenarios if used *very* carefully for the purpose of a quantization.

[18] See https://computation.llnl.gov/projects/floating-point-compression, last visit: 24.05.2017.

values of any precision while being significantly faster and more memory-efficient. The algorithm is also usable for lossy compression, though it is often outperformed by ZFP in this regard.[19] In [Ratanaworabhan et al. 2006], a fast lossless algorithm for 64-bit floating-point data is presented. It predicts each value in the sequence and XORs it with the true value. If the prediction is close, the sign, the exponent, and the first few bits of the mantissa are the same. Due to the XOR operator, a substantial number of zeros can be expected in this case. In [O'Neil and Burtscher 2011], the GFC scheme for the very fast and highly parallel compression of huge amounts of 64-bit floating-point data on GPUs is presented.
At this point, we do not dive further into the lossless compression of floating-point data and refer to [Lindstrom 2014, ch. 2.1] for further reading. There, a small survey for lossless floating-point data compression is given. It can serve as a starting point for additional studies.

2.2. Approximating a Point Set with Bounding Volumes

Compressing the coordinates of a resource's data points can *possibly* reduce storage space requirements but there is *no guarantee* it is advantageous in each case. Furthermore, the compression ratios to be expected for occuring cases are not very high. Additionally, a simple compression does not facilitate the *efficient* pruning of entire resources from search since every single one of its data points still must be examined for its relevance.[20] Furthermore, also the efforts for the decompression have to be considered. Hence, solely relying on compression is not suitable for very efficient searches in spatial data spaces. In contrast, concise resource summaries delineating the area(s) in which the data points of a resource are located can help both with reducing storage space requirements as well as with efficient pruning. The latter is because the indexed area(s) facilitate computing lower bound distances for the whole data point set. One option to design such resource summaries is using an appropriate bounding volume. A bounding volume can be used to concisely describe an arbitrary set of data points since it simplifies potentially very complex object representations (such as arbitrary data point clouds) by using a simpler geometric form that completely encloses the complex object(s).
Appropriate bounding volumes are obtainable from e.g. the field of bounding volume hierarchies. In general, this research area is concerned with employing suitable geometric shapes for applications such as ray shooting, nearest neighbor finding, or collision detection [Langetepe and Zachmann 2006, p. 22]. There is always a trade-off between the simplicity of the shape (for low storage space requirements as well as fast and simple distance and intersection calculations) and the tightness of the shape [Haverkort 2004,

[19] See footnote[18].

[20] Remember that a resource might also be a node in a multidimensional data structure like an R-tree. If it is an internal nodes, all the data points administered in the node's subtree would have to be assessed.

p. 11]. An extensive survey of the most important bounding volumes is given in [Dinas and Bañón 2015, ch. 3]. Further listings of bounding volumes can be found in [Haverkort 2004, ch. 1.1] and [Langetepe and Zachmann 2006]. The aforementioned trade-off always plays a role. For example, spheres allow for efficient intersection and distance calculations, have a small storage footprint, and are invariant to translations and rotations. However, they do not fit tightly for elongated objects for which ellipsoids are a better fit—at the costs of higher computational complexity and storage space requirements. Reasonably, bounding volumes are best kept as small as possible. In [Welzl 1991], algorithms for the (non-trivial) calculation of the smallest enclosing spheres (or balls) and ellipsoids for a set of n d-dimensional points are given. Other shapes used for bounding volume hierarchies include:

— Axis-aligned (minimum) bounding boxes: Also called iso-oriented rectangles, rectilinear rectangles, (minimum) bounding rectangles, or simply boxes. The faces of these rectangles are parallel to the coordinate axes. The extent in each dimension i is defined by a bounding interval $I_i = [l_i, u_i] \in D^d, 0 \leq i < d$ for the d-dimensional data space D.
— Arbitrarily oriented (minimum) bounding boxes: The arbitrary orientation often allows for a much tighter fit but its computation as well as the distance and intersection tests are more expensive. With regard to the efficient computation of these bounding boxes, Freeman and Shapira proved that the minimum oriented bounding box has a side collinear with one of the edges of its convex hull [Freeman and Shapira 1975].
— Convex hulls: The smallest convex polygons containing the object(s).
— k-DOPs: k-discrete orientation polytopes, also called fixed-direction hulls (FDH), which are discretely oriented convex polygons.
— Sphere packing: An arrangement of non-overlapping spheres enclosing a space. The spheres can be of equal or different size.
— Prisms and cylinders: Barequet et al. proposed them for 3D data (see [Barequet et al. 1996]). Both are very expensive to compute.
— Different kinds of swept sphere volumes, such as point swept sphere volumes, line swept volumes, or rectangle swept sphere volumes—also referred to as the Minkowski sum of a box and a sphere (for example in [Haverkort 2004, p. 3]). Also see [Larsen et al. 1999, ch. 4] for further details.

Combinations of several bounding volumes are conceivable, too. Examples are set-theoretic differences of two rectilinear rectangles, intersections of a box and a sphere, shells (which are the space between two concentric spheres), or spherical shells (which are the intersection of a shell and a cone where the cone's apex coincides with the sphere's center). The adequate type of bounding volume is dependent on the input, obviously. In bounding volume hierarchies, overall, rectilinear rectangles often seem to work out

better than more complex shapes despite the (potentially) bad fit to the data, though [Haverkort 2004, p. 11].
The shapes described so far are rather straightforward due to the necessary simplicity of the geometric forms. Obviously, there are also more profound bounding volumes. For the summarization of point data streams—where only a limited amount of storage space can be expended—approximated convex hulls have been proposed [Hershberger and Suri 2004]. These can be sampled uniformly (i.e. the directions of the vertices are only chosen in fixed directions) or adaptively (i.e. there are additionally some dynamically chosen directions which are based on the set of data points to approximate). Not all data points are contained in the approximated convex hull but they are error-bounded by so-called uncertainty triangles—a triangle for each edge of the sampled hull. It is formed by the edge and the tangents at the two endpoints of the edge in the sample directions for which those endpoints are extreme. Thus, the union of the approximated convex hull and its uncertainty triangles contains all data points. Additionally, several generalizations of the convex hull have been proposed for point data. Among these are the alpha shapes [Edelsbrunner et al. 1983]. The idea of the alpha shapes is a response to the shortcomings of defining the outline of the bounding shape by a pivoting line segment (like it is done for the convex hull with e.g. Jarvis' march algorithm [Jarvis 1973]). Instead, a disk of a given radius defines the outline of the shape. Let S be a finite set of points in the plane and α be a positive real number. The α-hull of S is the set of points that do not lie in any open disk of radius α, i.e. the boundary consists of circular arcs of constant curvature $1/\alpha$. By replacing the circular arcs with straights, an α-shape is obtained [Edelsbrunner 2010, p. 1f]. Both of these alpha shapes can lead to high storage space as well as computational requirements and might also contain holes. Alpha shapes are often considered for reconstructing a shape from a given finite, unordered set of points. In general, a lot of solutions exist for this problem. However, the so-called shape reconstruction problem is out of scope for this work since the focus is often on 3D shapes and the resulting shapes are too storage-intensive, too complex, and partly not even suited for our scenario (when for example non-closed or open curves are reconstructed). The interested reader is referred to [Edelsbrunner 1998]. There, it is stated that most of the solutions for the shape reconstruction problem follow one of four general approaches. Subsequently, the solutions using restricted Delaunay complexes are surveyed, and references to some solutions falling into the other classes are provided.

2.3. Representing a Complex Object with a Set of Simpler Objects

The computational costs for distance and intersection tests for 'complicated' shapes such as complex polygons or alpha shapes can be very high, not to mention the storage space requirements. The high computational

costs and possibly also the storage space requirements can be reduced by partitioning a complex shape into a set of simpler shapes such that the union of these simple shapes results in the complex shape again.
The field of possible partitions is wide and also dependent on the prerequisits and goals. One important property is whether the domain of the data is discrete or continuous. In the continuous domain, various partitioning shapes can be considered (such as convex, star-shaped, hexagonal, rectangular, etc., see [Keil 2000]). In the discrete domain, the methods should only use rectangular blocks because of the native rectangular structure of the discrete image domain [Höschl IV and Flusser 2016, p. 251]. Furthermore, there are partitions of complex objects into maximum sets and minimum sets of simpler objects, and these simpler objects may or may not overlap. For example, the triangulation of a simple polygon P (i.e. a polygon that does not contain holes) in the continuous domain is a *decomposition* (i.e. a non-overlapping partition) into triangles by a maximum set of non-intersecting diagonals where a diagonal is an open line segment that connects two vertices of P and lies in the interior of P [de Berg et al. 2008, p. 46]. In contrast, the polygon *cover* problem (see section 2.3.2) is concerned with the partitioning of rectilinear polygons into the minimum set of (possibly overlapping) rectilinear rectangles in a discrete domain.
For our application, we omit from partitions of complex shapes in the continuous domain: The approaches described in section 2.4 are much better suited for resource summaries due to their inherent reduction of indexed dead space. For the discrete domain, there are useful applications related to quantization-based[21] hybrid approaches described in section 4.3. Therefore, three research areas which are relevant for the partitioning of complex objects in the discrete domain are discussed in the following.

2.3.1. POLYGON DECOMPOSITION PROBLEM. We already touched the problem of *decomposing* general polygons. There, a polygon is to be represented as the union of a number of simpler, disjoint component parts. Suitable algorithms began to appear in computational geometry in the 1980s [Suk et al. 2012, p. 4282]. The problem also has been studied for dimensions higher than $d = 2$. Nevertheless, we focus on the two-dimensional case which has been studied in computational geometry for years [Höschl IV and Flusser 2016, p. 251]. More specifically, we consider the rectilinear polygon decomposition in the discrete domain. There, commonly accepted measures for the quality of the decomposition are the number of resulting blocks[22] (most important criterion) and the time complexity of the decomposition (usually as a secondary criterion).
Several authors ([W. Lipski Jr. and Pagli 1979; Ohtsuki 1982; Ferrari et al. 1984]) independently proposed basically the same algorithm which guar-

[21] Note that the continuous domain of our spatial data space can be transformed into a discrete domain by rasterizing the continuous data space.
[22] A block is a rectilinear rectangle.

antees an optimal number of blocks for an arbitrary rectilinear shape (even for polygons with holes[23]) and runs in polynomial time. In principle, the algorithm transforms the decomposition problem into a graph partitioning problem and then employs tools known from graph theory for a solution. First, all 'concave' vertices (i.e. those having an inner angle of 270°) of the rectilinear polygon are detected. Afterwards, pairs of 'cogrid' vertices (i.e. those having the same horizontal or vertical coordinates) are identified. The input polygon is then divided into subpolygons by constructing chords which connect certain cogrid concave vertices. A solution is optimal if the chords do not intersect each other and the chord number is the maximum possible. An optimal solution can be found by first constructing a bipartite graph based on the horizontal as well as vertical chords and their intersections. Then, the maximum independent set for this bipartite graph has to be found. The resulting subpolygons are further divided by building a vertical or horizontal chord to the opposite subpolygon boundary for each concave (non-cogrid) vertex left. See for example [Suk et al. 2012, p. 4283] for details. The basic algorithm has been improved by Imai and Asano ([Imai and Asano 1986]) and Eppstein ([Eppstein 2009]) to run in $\mathcal{O}(n^{\frac{3}{2}} \log n)$ time[24] [Höschl IV and Flusser 2016, p. 251]. Soltan and Gorpinevich provide an $\mathcal{O}(n^{\frac{3}{2}} \log n)$ time algorithm that works even if the holes degenerate to points ([Soltan and Gorpinevich 1993]). If the polygons do not contain holes, faster algorithms are possible [Keil 2000, p. 14]. See for example [Liou et al. 1991] for an $\mathcal{O}(n)$ time optimum decomposition algorithm. Additionally, some methods applied for binary image compression can be seen as ways of decomposing rectilinear polygons into rectilinear rectangles (see section 2.3.3). Most of them are suboptimal in the number of resulting blocks, though.

2.3.2. POLYGON COVER PROBLEM. Closely related to the rectilinear polygon decomposition problem is the rectilinear polygon *cover* problem. The decisive difference is that for the latter, the overlap of rectangles is allowed. The aim is to cover a rectilinear polygon with the minimum number of rectilinear rectangles. Within the cover problem, it can be distinguished whether the interior, the boundary, or the corners shall be covered [Berman and DasGupta 1997, p. 3].[25] For polygons with holes and even for simple polygons, the rectangle cover problem is NP-hard and MaxSNP-hard, i.e. there is not even a polynomial time approximation scheme. The best known approximation algorithms achieve an approximation guarantee of a factor of $\mathcal{O}(\sqrt{\log n})$ for general polygons[26] (see [Kumar and Ramesh 2003]) and a factor of 2 for simple polygons (see [Franzblau 1989]) [Heinrich-Litan and

[23]The holes in the rectilinear polygons also have to be rectilinear.

[24]Note that in this paragraph, n corresponds to the number of vertices in the rectilinear polygon.

[25]Note that for our work, the cover of the interior is relevant.

[26]Here, n is the number of edges of the polygon.

Lübbecke 2006, p. 3]. For some restricted types of rectilinear polygons, the problem is exactly solvable in polynomial time. However, this is out of scope for this work. The interested reader is referred to [Keil 2000] and [Berman and DasGupta 1997] for further information on exact solutions for special cases and heuristics for general cases.

2.3.3. BINARY IMAGE COMPRESSION. Binary images are images only consisting of black and white pixels. They can be represented in a more efficient way than as full-sized matrix of zeros and ones representing the white and black pixels of the image. Generally, the representation of a binary image coincides with the representation of the grid cells of rasterized areas which do (i.e. black pixel) and which do not (i.e. white pixel) contain data points in our application scenario.[27]
Methods for efficient binary image representation can generally be divided into two groups [Suk et al. 2012, p. 4279]. The *boundary-based methods* utilize the property that the boundary of a binary object contains complete information on the object (all other pixels are redundant). The *decomposition methods* try to express an object as a union of simple disjoint subsets called blocks. As for the polygon decomposition, the blocks are rectilinear rectangles (due to the rectangular structure of the discrete image domain) and the methods differ from one another by the decomposition algorithms (and consequently by how many disjoint blocks result). In the following, we focus on the decomposition methods for which a survey is given in [Suk et al. 2012].
For a method known as *Delta-Method* [Zakaria et al. 1987] or simply RLE, the blocks are continuous row segments for which only the coordinate of the beginning and the length is stored. Slight improvements and generalizations exist, see [Suk et al. 2012, p. 4280]. The *Generalized Delta-Method* [Flusser 2000] is a rectangular-wise object representation instead of the row-wise representation of the Delta-Method. It unifies matching adjacent rows. For the *quadtree decomposition*, the image is iteratively divided into four equal quadrants (regular decomposition) until all quadrants are homogeneous, i.e. fully black or white. It is not clearly stated in [Suk et al. 2012] how the quadtree is represented in the end. Note that it could either be by representing the single blocks directly (i.e. for example capturing the lower left and the upper right corner of each block) or by a linear quadtree encoding (see section 3.2).[28] The *morphological decomposition* [Sossa-Azuela et al. 2001] is based on morphological erosion. It works in an iterative manner and basically finds the largest square inscribed in an object (i.e. a coherent black area), removes it, and looks for the largest square inscribed in the rest of the object. This process is repeated until the ob-

[27] In our application scenario, areas can be rasterized due to the utilization of quantization.

[28] Note that in general, linear quadtree encodings have been extensively researched in the binary image compression domain.

ject is completely decomposed. The *distance transform decomposition* [Suk and Flusser 2010] is a sped up version of the morphological decomposition which ends up with the same set of blocks. The *graph-based decomposition* presented in [Suk et al. 2012] is based on the group of block-optimal decomposition algorithms for rectilinear rectangles described in section 2.3.1.
The representation of binary images is relevant to our work in two ways. As a first touching point, we could use these approaches for the design of summarization approaches. For example, we could rasterize the entire data space. For each resource, the resulting 'image' cells containing at least one data point could then be colored black. The binary image decomposition methods would thus be applicable to index the areas in which the data points of a resource are located as blocks. Alternatively, a *further development* of the quantization technique used by for example the LSD^h-tree (see section 2.4.3) to overcome indexed areas which have large empty portions is conceivable.[29] In contrast to rasterizing the entire data space, here, only subspaces (defined by a bounding volume or a cell of a space partition) occupied with data points are rasterized—in order to further refine these indexed areas. If we interpret the quantization raster of such a subspace as a binary image (with black cells being cells of the raster containing at least one data point) and apply block-based binary image encoding techniques, a *set* of quantized rectangles refines this subspace. This way, dead space can be reduced more effectively than by the usage of a *single* quantized MBR for all the data points located in this subspace (like in the LSD^h-tree) which still might contain a lot of dead space. Naturally, this comes at the cost of increased storage space requirements since each block has to be encoded.
As a second touching point, some of the approaches developed by us are easily adaptable for utilization in the field of binary image compression, for example $\text{QTMBR}^b_{c,a}$ or $\text{QTMAR}^{b,k}_{c,a}$ (see section 4.3).

2.4. Dividing a Set of Points into Groups and Representing Each Group by a Separate Area

Obviously, the suitability of summarization approaches respectively their summaries can vary depending on the query type to support (see section 2.5 for different query types on spatial data). As a non-query-specific design approach, it is reasonable to reduce the dead space indexed by the single resource summaries as far as possible while simultaneously spending the least possible amount of storage space—which obviously are contradictory objectives.
Reducing dead space is generally suitable for all query types. One example are range queries. There, for each resource, it has to be tested whether the query ball intersects the indexed area (i.e. a simple yes/no-decision which

[29] The LSD^h-tree refines cells of a space partition with a single, quantized minimum bounding rectangle (MBR). See the 'KDMBR^b_n'-paragraph of section 4.3 beginning on page 83 for details on this procedure.

is not dependent on the eventual previous query processing has to be made for each resource).[30] The processing of query types which rely on a ranking of the resources (like kNN queries) also profits from more accurately delineated resource descriptions. In contrast, when the indexed area(s) of many different resources overlap to a large degree and additionally contain a lot of dead space, the ranking—which is usually lower-bound-based (also see section 5.1)—becomes very inaccurate. Obviously, minimizing the overlap *between* different resource summaries requires additional knowledge besides knowing a single resource's set of data points and ideally also control on the assignment of data points to resources. This is not the case for all conceivable application scenarios. In contrast, the minimization of dead space for a given data point set of a resource is an objective which can be achieved independently from additional knowledge. Furthermore, it should usually correlate with the overlapped surface area—the less indexed area, the less overlap between summaries (most likely).
Overall, *single* bounding volumes are not suitable for this task—despite arbitrarily complex forms (which additionally can result in high demands regarding computational and storage space costs). For example, for a resource administering data points in Europe and Australia, an MBR is completely unsuitable as a description since large parts of Africa and the Middle East would also be indexed. The use of a convex hull is considerably better with regard to the surface of the indexed area. Still, there would be an indexed 'corridor' between Europe and Australia which—depending on the locations of the data points—for example might contain India. Such elongated, skinny areas which only contain data points in their opposite corners are very disadvantageous for the processing of any query type. Moreover, a convex hull potentially can have very high storage space requirements. Overall, a better approach in the given situation is to divide the set of data points into two groups—one for Europe, one for Australia—and to describe each group by an MBR of its own.
In general, a reasonable strategy for the *efficient* reduction of dead space is to divide a set of data points into 'spatially coherent' groups and to describe each group by a bounding volume of its own—which preferably fits tightly and requires low amounts of storage space. There are several practicable approaches to divide a set of data points into groups. For example, most clustering methods (see section 2.4.1) arrange data into groups based on the attribute values of the data (such as its spatial location). This is a very data-centric approach which does not take (the extent of) the data space from which the data points are drawn into account. In fields like the multidimensional data structures (see section 2.4.3), there also exist very data-centric approaches which aggregate objects (like data points)

[30] Note that an intersection of the resource summaries with the query ball is only the first step in processing range queries. In the second step, the data points of this resource have to be tested if they are located inside the query ball. Resources with non-intersecting summaries can be pruned without further examination.

into object pyramids and thus organize data point groups in a hierarchical multiway search tree structure. From these index structures, both the algorithms to *partition the data* as well as their ways of indexing the data-point-containing areas can serve as basis or inspiration for resource summaries.

Besides data partitioning approaches, there are also approaches which—in addition to a resource's data points—take the entire data space into account. They *partition the data space* into a set of disjoint subspaces whose union yields the complete data space again. These approaches can be further differentiated by whether they consider the 'importance' of the embedding space to a 'greater' (resulting in a *regular* decomposition of the data space) or to a 'lesser' extent (resulting in an *irregular* decomposition of the data space). Either way, the data points can then be grouped by the subspaces they are located in. In addition, a group obviously can be described by this very subspace to efficiently minimize the indexed dead space. Suchlike approaches exist in the field of multidimensional data structures, too, but also in the domain of computational geometry (see section 2.4.2).

The minimization of dead space is not specifically tailored for a particular query type and only considers the data points of a single resource. When a ranking of resources is assumed and an additional focus on the 'interplay' of the whole set of resource summaries is taken into account[31], there are also other conceivable design approaches besides minimizing the indexed dead space for every single resource. For example, for an efficient processing of kNN queries, the data space can be partitioned such that in each subspace, a fairly homogeneous number of data points of the *entire data collection* is located. For a data collection which is non-uniformly distributed, this results in very fine-grained subspaces for regions which are densely populated with data points and coarse-grained subspaces for sparsely populated regions. This means that the summaries of the resources administering data points in these coarse subspaces would index a lot of dead space. Furthermore, with this design approach, the basic set of subspaces is the same for all resources. Thus, the corresponding resource summaries only differ by in which subspaces of this predefined set the data points of the respective resource are located. In the following, we label this design approach as 'global space partitioning'. It is discussed in more detail in section 4.2. As a result of a homogeneous distribution of data points to subspaces, usually a relatively consistent, low number of resources should maintain data points in a single subspace. In general, the kNN query processing can primarily focus on the very subspace the query point falls into and its surrounding subspaces. Consequently, the number of resources to consider is also limited in case of homogeneous distributions of data points to subspaces—and the kNN query is efficiently solved.

[31] Directly represented resources are not ranked. See section 6.4.1 for more details.

Of course, this design approach is not as widely applicable as the minimization of dead space. For example, it would be rather unsuitable for range queries. It also requires the availability of additional information concerning the global data point distribution. Furthermore, it is required that the ratio of data points to subspaces is not too great. Nevertheless, for specific query types such as kNN queries, it might be suitable. In general, this design approach is also based on *partitioning the data space*—for which again methods known from computational geometry and multidimensional data structures are relevant.
The remainder of this section is organized as follows: In section 2.4.1, clustering is discussed, followed by section 2.4.2 which outlines the field of computational geometry. Finally, section 2.4.3 concludes with an extensive overview of multidimensional data structures.

2.4.1. CLUSTERING. Clustering is a division of data (points) into groups of 'similar' objects which then are called clusters [Berkhin 2006, p. 2]. Generally, the following broad categories of clustering techniques are distinguished in clustering literature (such as [Berkhin 2006], or [Amini et al. 2014] and [Jeong et al. 2016] as more recent ones): hierarchical clustering, partitioning-based clustering, density-based clustering, model-based clustering, and grid-based clustering.
Hierarchical methods build a cluster hierarchy or a tree of clusters (known as dendogram). Every cluster contains child clusters which partition the data points covered by the parent. It can be distinguished between agglomerative (starting with one-point-clusters and recursively merging these) and divisive (starting with one cluster for all points and recursively splitting the most appropriate clusters) methods [Berkhin 2006, p. 6]. It is hard to apply hierarchical methods on big data due to their computational complexity [Jeong et al. 2016, p. 2].
In contrast to the gradually cluster-building hierarchical methods, *partitioning methods* learn the clusters directly. The data is divided into several subsets and greedy heuristics in the form of iterative optimizations are applied which are relocation schemes that iteratively reassign points between k clusters [Berkhin 2006, p. 12]. The partitioning method k-means is by far the most popular clustering tool [Berkhin 2006, p. 15]. Given a data set S of d-dimensional entities $s_i \in S$, for $i = 1, 2, \ldots, N$, k-means generates k non-empty disjoint clusters $C = \{C_1, C_2, \ldots, C_k\}$ around the centroids $c = \{c_1, c_2, \ldots, c_k\}$ by iteratively minimizing the squared Euclidean distances within clusters while maximizing the distances between clusters. In the original k-means algorithm [Lloyd 1982], the k initial centroids are chosen randomly. Improved seeding techniques for selecting the initial centroids exist, for example the k-means++ algorithm [Arthur and Vassilvitskii 2007]. Likewise diverse other methods, k-means requests the number of clusters to be specified beforehand. This is often hard to decide for the user. A popular solution for this problem is to run the given clustering algorithm using different values for k and to analyze

the generated clusterings afterwards. The estimation of how well a partition fits is also called cluster validation for which different approaches exist [de Amorim and Hennig 2016, p. 2]. Examples are the Silhouette index [Rousseeuw 1987], the Dunn's index [Dunn 1973], the Calinski-Harabasz index [Caliński and Harabasz 1974], the Hartigan index [Hartigan 1975], or intelligent k-means [Mirkin 2012] (which additionally addresses the issue of finding 'good' initial centroids).

Density-based methods also learn clusters directly—but clusters are found as dense areas which are separated from sparse regions. They are flexible in terms of their shape and less sensitive to outliers. Therefore, they can discover clusters of irregular shapes. A well-known representative is the DBSCAN algorithm [Ester et al. 1996].

Cluster analysis can also be based on *probabilistic models* (as opposed to exploratory, heuristic, or algorithmic approaches). These model-based clustering methods attempt to optimize the fit between the given data and some mathematical model like the EM (for Expectation Maximization) algorithm [Dempster et al. 1977] which can be viewed as an extension of the k-means method. However, EM assigns the objects to a cluster based on a weight representing the membership probability. A broad survey on probabilistic models for hierarchical and partitioning-based clustering structures can be found in [Bock 1996].

Instead of performing data partitioning like the other methods, the *grid-based methods* utilize space partitioning [Berkhin 2006, p. 21]. As the name implies, the data space is partitioned into a number of cells or segments which form a grid. Then, appropriate space segments are aggregated. Grid-based methods work indirectly with the data by constructing summarizations of the data over the attribute space subsets. Many of these methods use hierarchical agglomerations as one phase of processing [Berkhin 2006, p. 5]. The advantages of grid-based methods are that they are insensitive to data ordering (in contrast to relocation methods or incremental algorithms) and work well with attributes of different types (in contrast to density-based methods which work best for numerical attributes) [Berkhin 2006, p. 21]. For further reading on clustering methods, we refer to [Berkhin 2006] and [Xu and Wunsch II 2005]. The latter introduces a finer categorization scheme distinguishing nine broad categories and several subcategories of clustering methods.

2.4.2. COMPUTATIONAL GEOMETRY. The field of computational geometry emerged in the 1970s and is concerned with the systematic study of algorithms and data structures for geometric objects. Various fundamental geometric concepts and structures for *space partitioning* are described in the literature for computational geometry such as [de Berg et al. 2008]. Some of these are relevant for us.

[32]The Voronoi diagram is also known as Thiessen polygon or Dirichlet domain [Samet 2005, p. 346].

One of the most fundamental concepts is the Voronoi diagram[32]. A Voronoi diagram is the subdivision of the $\mathtt{R}^d$ space into n cells by a set of n sites $S = \{s_1, s_2, \ldots, s_n\}$, i.e. one cell per site. Each region includes all points p in $\mathtt{R}^d$ with a common closest site s_i in the given set S according to a distance metric $\mathtt{D}$ [Samet 2005, p. 69]. That is, the region corresponding to the site $s_i \in S$ contains all the data points $p \in \mathtt{R}^d$ for which we have $\forall s_j \in S, s_j \neq s_i, \mathtt{D}(p, s_i) \leq \mathtt{D}(p, s_j)$ [Sharifzadeh and Shahabi 2010, p. 2]. A more detailed discussion about Voronoi diagrams, their structure, and their computation can be found in [de Berg et al. 2008, ch. 7].
Given the n sites defining the Voronoi diagram, also some other space partitions beside the 'standard' Voronoi diagram can be derived. The farthest-point Voronoi diagram divides the plane into cells in which the same site is the farthest reference point. Not every site necessarily has a cell in the farthest-point Voronoi diagram. Hence, the number of cells can vary for a set of n sites—depending on the location of the sites [de Berg et al. 2008, p. 164]. In higher-order Voronoi diagrams, the space is not partitioned according to one closest site but according to the k closest sites ($1 \leq k \leq n$). For a given k, the resulting diagrams are called order-k Voronoi diagrams. Note that the order-1 Voronoi diagram is the standard Voronoi diagram whereas the order-($n - 1$) Voronoi diagram is the farthest-point Voronoi diagram [de Berg et al. 2008, p. 169]. Hierarchical Voronoi diagrams (see for example [Weng 1995, p. 5f]) are hierarchical partitions where every cell is further partitioned at a deeper level by the sites which did not already define the cell at the current level. For example at level 2, the space of a cell c_i which is defined by the site s_i is further divided by the sites $s_j \in S$, $i \neq j$. This recursive approach of space partitioning is also known as permutation-based partitioning [Esuli 2009] or recursive Voronoi-like partitioning [Novak and Zezula 2014] in the metric space domain. The Delaunay triangulation is the dual graph of the Voronoi diagram. All pairs of sites whose Voronoi cells are adjacent are connected. The resulting set of segments forms a triangulation of the site set [de Berg et al. 2008, p. 168]. The triangulation of a data point set is a very well-known problem in other domains besides computational geometry—such as numerical analysis and computer graphics. Generally, long and skinny resulting triangles should be avoided. Of course, the decomposition of point sets is not restricted to triangulations: quadrilaterals or other polygons are also possible. If additional points (called Steiner points) besides the given points are allowed, the problem is also known as meshing [de Berg et al. 2008, p. 214]. We do not go further into detail about meshes at this point and refer to [Bern et al. 1994] and [Goodman et al. 2017, ch. 29] for further reading.

2.4.3. MULTIDIMENSIONAL DATA STRUCTURES. With regard to the design of resource summaries for sets of spatial data points, most relevant to our work are multidimensional data structures [Samet 2005] which are often also referred to as spatial (data) access methods [Ahn et al. 2001; Oos-

terom 1999], spatial indexes [Ramakrishnan and Gehrke 2003], or multidimensional access methods [Gaede and Günther 1998].
In general, spatial databases contain multidimensional data which is usually represented in some vector-based format. Multidimensional data structures support search operations on (centralized) spatial databases [Gaede and Günther 1998, p. 171], i.e. the efficient selection of objects based on spatial properties [Oosterom 1999, p. 385]. The main problem with spatial data is that there exists no total order that preserves spatial proximity—there is no mapping from a two- or higher-dimensional space into one-dimensional space such that all objects which are spatially close in higher-dimensional spaces are also guaranteed to be close to each other in the one-dimensional space [Gaede and Günther 1998, p. 173]. Therefore, traditional one-dimensional data structures cannot be used efficiently and special data structures—the multidimensional data structures—have been developed to handle multidimensional data efficiently. Most of the corresponding research has been conducted from the late 70s to the late 90s. Various survey articles on multidimensional data structures exist, such as [Oosterom 1999], [Ahn et al. 2001], [Böhm et al. 2001], or in particular [Gaede and Günther 1998] as well as the renowned encyclopedia on multidimensional data structures by Samet ([Samet 2005]). In multidimensional data structures, the data is usually organized in tree- or hashing-based structures. Consequently, the multidimensional data structures are relevant to our work in two respects: a) how are the data points assigned to nodes (i.e. how is the data point set split into groups), and b) how are the spatial contents of a node indexed (i.e. how are the data-point-containing areas described geometrically).
There are many ways to classify the different existing approaches. For example, early multidimensional data structures did not take paged second memory (i.e. disk storage) into account [Gaede and Günther 1998, p. 185] as they assumed all the data to fit into main memory. For data structures considering secondary memory (i.e. PAMs and SAMs, see below), the data is organized into a number of pages or buckets each of which corresponds physically to a disk page and logically to a subset of the entire data space (or universe) [Gaede and Günther 1998, p. 185]. These page or bucket regions can for example be hyperspheres, hyperrectangles, multidimensional cylinders, or set-theoretical combinations of these. Regardless of their shape, they are conservative approximations of the stored data objects [Böhm et al. 2001, p. 16].[33] Generally, the methods can also be distinguished whether they partition the data (also called *data partitioning*, DP) or the data space (also called *space partitioning*, SP). The SP methods can be further differentiated whether they *organize* the data or the embedding space. In both cases, the space from which the data is drawn is decomposed into subspaces (irregularly for the methods organizing the data,

[33]The bounding volumes discussed in section 2.2 are also conceivable shapes for page regions.

regularly for the methods organizing the embedding space) [Samet 2005, p. 184]. Most commonly, when considering secondary memory, it is differentiated between Point Access Methods (PAMs) and Spatial Access Methods (SAMs). PAMs are designed to perform searches on point databases. The point objects are embedded into the d-dimensional space and do not have a spatial extent (obviously). For classical PAMs, the data space is usually partitioned into a set of mutually disjoint subspaces, with their union corresponding to the universe [Gaede and Günther 1998, p. 191]. Hence, SP methods are used for PAMs which are then often referred to as bucket methods. In contrast to PAMs, SAMs can manage extended spatial objects such as lines, rectangles, polygons, and so on [Gaede and Günther 1998, p. 174]. For SAMs, it can be further distinguished whether the representation of an extended object o is interior-based (using the locations in the space that make up o) or boundary-based (using the locations in the space that are adjacent to the boundary of o) [Samet 2005, p. 192f]. The interior-based methods are closer to our work. Thus, we omit the boundary-based methods in the remainder of this thesis. Generally, PAMs can also be transformed to SAMs by using one of the following techniques [Gaede and Günther 1998, p. 200ff]:

— **transformation**: The extended objects can be mapped to a higher-dimensional space. For example, a two-dimensional rectangle can be represented by a four-dimensional point. Midpoint and endpoint transformations are possible for embedding the objects into the higher-dimensional space. Another option is the use of space-filling curves (explained later in this section) for extended objects, i.e. mapping the objects into a one-dimensional space.
— **overlapping regions (object bounding)**: The key idea is to allow different data buckets to correspond to mutually overlapping regions. This allows assigning any extended object directly and as a whole to one single bucket region.
— **clipping (object duplication)**: No overlap between bucket regions is allowed, they have to be mutually disjoint. Any data object that spans more than one bucket region has to be inserted in all the respective buckets of the corresponding regions.
— **multiple layers**: The space is partitioned several times and each partition is termed a layer. The layers are organized in a hierarchy. Each layer partitions the universe in a different way. The data regions within a layer are disjoint. The data regions do not adapt to the spatial extents of the data objects. Generally, a data object is inserted into the lowest layer in the hierarchy whose hyperplanes do not split the object.

There are still many other ways to distinguish between the approaches. A selection of these are the following:

— Are the approaches designed for static or dynamic data?

— Is the organization of the data (buckets) hierarchy-/tree-based or grid-/hashing-based?
— Which performance goals are pursued (for example search performance, insertion performance, or high storage utilization)?
— Which query types shall be supported?
— What is the dimensionality of the data to index (low-dimensional or high-dimensional)?

Furthermore, the classifications are often ambiguous as many approaches are hybrid and try to combine strengths of different classes.
The variety of the applications for spatial data and their requirements, as well as of the properties of the spatial data itself have led to a plethora of approaches. Nevertheless, most of the approaches can be associated to a number of certain families of multidimensional data structures—each being adapted to a respective scope of application. The patterns in which the data or the space are partitioned and in which the data-point-containing areas area indexed—which are the relevant parts for our work—are generally recurring for a single family. In the following, we present the most relevant and important multidimensional data structure families along with their most important and interesting manifestations. The main focus is on how a set of data points can be arranged into groups—i.e. how the data or the data space are partitioned—and on how the data-point-containing areas are indexed—i.e. how they are geometrically described. Yet, for comprehensibility, we may also discuss some essential additional properties of specific multidimensional data structures, the history of inter- and intra-family evolutions, or adjustments made for specific approaches in specific application scenarios and their corresponding goals.

Origins. For almost all multidimensional data structures, the origins lie in the field of *classical one-dimensional access methods* [Gaede and Günther 1998, p. 179]. The most important one-dimensional structures are *linear hashing* [Litwin 1980], *extendible hashing* [Fagin et al. 1979], and the *B-tree* [Bayer and McCreight 1972]. Many multidimensional data structures are generalizations of these techniques for a higher-dimensional space.

Hashing-Based Approaches. For multidimensional data structures based on *hashing*, the *grid file* [Nievergelt et al. 1984] is a typical representative. It superimposes a d-dimensional orthogonal grid on the universe, resulting in a decomposition of the space into d-dimensional rectangles. The hyperplanes of the grid are oriented on the data points, i.e. the grid might be non-regular. The grid itself is represented by d one-dimensional arrays called linear scales which depict the positions of the hyperplanes in the respective dimensions. A grid directory associates one or more of the grid cells with a data bucket [Gaede and Günther 1998, p. 187f]. The only constraint is that the union of these cells forms a d-dimensional hyperrectangle. The bucket regions are mutually disjointed and their union

spans the entire data space. The buckets of the directory correspond to decomposing the underlying space into an *adaptive k-d-tree* (see 'k-d-tree-based approaches'-paragraph beginning on page 41). The space partitioning of *multipaging* [Merrett 1978] is similar to the grid file but there is no grid directory. Instead, a data page and its potential overflow chain are determined by computing an address directly from the linear scales [Samet 2005, p. 136]. Grid file variants which make use of multiple grid files also exist. Examples are the *two-level grid file* [Hinrichs 1985], the *twin grid file* [Hutflesz et al. 1988b], and the *multilayer grid file* [Six and Widmayer 1988]. However, there is no conceptual difference in partitioning the underlying space. Closely related to the grid file is *EXCELL* [Tamminen 1982]. In contrast to the grid file, the data space is decomposed regularly, i.e. the grid cells are of equal size. Therefore, the linear scales are avoided. In order to maintain the property of equal size, all partitions of a dimension are split in case a cell must be split (in contrast to the grid file, where only one additional hyperplane is introduced) [Gaede and Günther 1998, p. 188f]. The buckets of the directory in EXCELL correspond to decomposing the underlying space into a *bucket PR k-d-tree* or *bintree* partition (see 'k-d-tree-based approaches'-paragraph) [Samet 2005, p. 160].
All of the approaches mentioned so far are based on *extendible* hashing. Multidimensional *linear* hashing methods also exist. In contrast to multidimensional extendible hashing methods, they only use very small or even no directories [Gaede and Günther 1998, p. 190]. They also do not permit a bucket to contain the contents of more than one grid cell but do allow a grid cell to be associated with more than one bucket (the additional buckets are then known as overflow buckets) [Samet 2005, p. 130]. Examples are *MOLHPE* [Kriegel and Seeger 1986], *quantile hashing* [Kriegel and Seeger 1987], *dynamic z-hashing* [Hutflesz et al. 1988a], or *PLOP hashing* [Kriegel and Seeger 1988] which all apply different strategies to perform the required address computation [Gaede and Günther 1998, p. 190]. Generally, they do *not* result in new methods of decomposing the underlying space compared to the techniques based on extendible hashing, though. Primarily, they are implementations of grid directory methods that provide an alternative in the form of a one-dimensional directory instead of a d-dimensional directory [Samet 2005, p. 187]. Therefore, we close the review of hashing-based approaches at this point.

Space-Filling Curves. Closely connected to grid partitions are *space-filling curves*. They are heuristics which define a total order on spatial objects that preserves spatial proximity to at least some extent, i.e. there is a high probability that 'originally' close objects are also close in this total order [Gaede and Günther 1998, p. 199]. Technically, the data space is decomposed into a uniform grid and a total order is imposed on the single grid cells (and thus of the objects located in these cells) by means of a space-filling curve. The effect of ordering corresponds to a mapping from d-dimensions (original space) to one dimension (embedded space). There-

fore, one-dimensional access methods such as the B-tree could be used as a (central) index for the spatial objects [Samet 2005, p. 11f]—which would be unsuitable for certain query types such as range queries, though [Gaede and Günther 1998, p. 199]. Space-filling curves can also be seen as way to eliminate empty cells of a grid [Samet 2005, p. 90].
A lot of different orderings exist. There are simple ones such as *bit interleaving*, *row-/column-ordering* (also known as *bit concatenation*) [Samet 2005, p. 90], or *row-/column-prime* [Oosterom 1999, p. 388], as well as complex space-filling curves which are constructed in more sophisticated ways. Examples of these are the *z-ordering* [Orenstein and Merrett 1984], the *Hilbert order* [Faloutsos and Roseman 1989], the *Gray order* and the *double Gray order* [Faloutsos 1986], or the *U order* [Schrack and Liu 1995]. For example, for the z-ordering (also known as *Peano curve*, *quad codes*, *N-trees*, or *locational codes* [Gaede and Günther 1998, p. 199]), the space is first partitioned into two equal volume halves which are perpendicular to the d_0-dimension. The lower-valued volume gets the name $<0>$ (as a bit string), the higher-valued volume gets the name $<1>$. Then, each volume is partitioned perpendicular to the d_1-axis the sub-partitions of $<0>$ ($<1>$) get the names $<00>$ ($<10>$) and $<01>$ ($<11>$). When all axes have been used for splitting, d_0 is used for a second split, and so on. The process stops when a predefined basic resolution of the implicitly constructed grid is reached. The other more complex space-filling curves are defined similarly but their numbering scheme is slightly more sophisticated. Spatial objects are 'transformed' by assigning the number of the grid cell they are located in [Böhm et al. 2001, p. 53f].
Desirable properties of space-filling curves are listed in [Samet 2005, p. 199f] and [Oosterom 1999, p. 388ff]. For example, the ordering should be stable (i.e. the relative order is preserved even when the grid resolution is e.g. doubled) or the process of retrieving the neighbors of a location in space should be simple. In general, the z-ordering and especially the Hilbert order are regarded as most suitable space filling curves in the context of multidimensional data structures [Gaede and Günther 1998, p. 199].

Quadtree-Based Approaches. A *quadtree* is a multidimensional data structure which is based on a hierarchical tree structure. It decomposes the universe by means of d iso-oriented hyperplanes. In the tree structure, each (internal) node has 2^d descendants, each of which corresponds to an interval-shaped partition of the given node's subspace. For $d = 2$, the space decomposition therefore results in four rectangular-shaped quadrants that are typically referred to as NW, NE, SW, and SE quadrant. The partitions do not have to be of equal size even though it is the case for many quadtree variants [Gaede and Günther 1998, p. 183f]. If the partition is of equal-sized cells, the corresponding quadtrees are called *trie*-based variants. When the positions of the decomposing hyperplanes are based on the values of the data (i.e. possibly arbitrarily-sized cells result), they are labeled *tree*-based variants [Samet 2005, p. 28,37]. The recursive decomposi-

tion continues until each partition complies to some homogeneity criterion for which many adaptions are possible (depending on the type of the data objects). All this results in tree structures which are possibly arbitrarily unbalanced [Gaede and Günther 1998, p. 184]. Quadtrees can be seen as a way to make grids adaptive by merging spatially adjacent empty (or sparse) grid cells into larger empty (or sparse) grid cells whilst splitting grid cells that are too full. Hence, a k-ary (where usually $k = 2^d$) tree access structure is used instead of an array access structure (like for example the linear scales of the grid file) [Samet 2005, p. 11].
One of the first quadtree variants is the *point quadtree* [Finkel and Bentley 1974] which is used for indexing point data. The first point is inserted into the tree structure as a root, the second point is inserted into the relevant quadrant of the tree rooted at the first point, and so on [Samet 2005, p. 28]. The space is partitioned by hyperplanes which are located at the coordinates of the data points of the respective nodes [Gaede and Günther 1998, p. 184]. Therefore, it is a tree-variant with arbitrarily sized regions. In [Samet 2005, p. 222f], an (unnamed) adaption of the point quadtree for region data is presented. As another variant, the *region quadtree* [Klinger 1971] is a trie-based quadtree for rasterized approximations of polygons. The universe is decomposed into equal-sized regions until the data satisfies some homogeneity condition. The *PR quadtree* [Orenstein 1982] (for point region) is a trie-based adaption of the region quadtree for the use with point data [Samet 2005, p. 42]. Originally designed for containing at most one data point in a leaf node, it can be easily adapted to contain an arbitrary number of data points in a region. This variant then is called *bucket PR quadtree* [Samet 2005, p. 45]. There are still more quadtree variants for specific applications such as the *PM quadtree* (for the boundary-based representation of polygons) [Samet and Webber 1985], the *MX quadtree* (for discrete and finite data domains) [Samet 1984], the *MX-CIF quadtree* (using a multilayer approach for extended objects) [Kedem 1982], or the *filter tree* (similar to the MX-CIF quadtree but sorting the objects by Hilbert codes) [Sevcik and Koudas 1996]. Nevertheless, at this point, we conclude the review of specific variants since the latest listed approaches are already thematically rather irrelevant for our purposes.
In connection with the homogeneity criterion, it is often distinguished between two general classes of quadtrees: In binary quadtrees, the leaf nodes are labeled black (white) in case they are fully (non-)occupied with data. In case it needs to be distinguished between the objects located in the single regions, multicolored trees are possible [Samet 2005, p. 211].

k-d-tree-Based Approaches. With a high dimension d, the fanout (i.e. the number of children of an internal node in the hierarchical tree structure) of quadtrees is very high (2^d). To alleviate this problem, *k-d-trees* have been introduced which partition the underlying space on the basis of just one attribute at each level instead of on the basis of all d attributes. Therefore, the order in which the various axes are partitioned becomes

important. Consequently, both the split position and the split axis must be considered for k-d-trees. Similar to quadtrees, with regard to the position, it can be distinguished between tree-based variants (arbitrary positions, subspaces of irregular size) and trie-based variants (split into equal-sized halves) of a k-d-tree. For the split axis, the options are to cycle through the axes in a predefined order or to split the axes in an arbitrary order (for the latter, the respective techniques are labeled with the qualifier 'generalized' in [Samet 2005, p. 48ff]).

The basic form of the *k-d-tree* [Bentley 1975] stores d-dimensional points. Each splitting hyperplane (cycling through the d dimensions) has to contain at least one data point which is also used for its representation in the tree (i.e. each internal node of the tree structure contains one point) [Gaede and Günther 1998, p. 180]. Hence, it is a tree-based variant with arbitrary split positions. In the *adaptive k-d-tree* [Bentley and Friedman 1979], the split position is chosen such that about the same number of data points is on both sides of the splitting hyperplane. Also, there is no need of strictly alternating the split dimensions [Gaede and Günther 1998, p. 181]. The *PR k-d-tree* [Orenstein 1982] (for point region) splits the data space into boxes of equal size and thus is a trie-based variant. For the splits, it cycles through the dimensions. These splits are repeated until each leaf contains at most one data point. Bucket-based variations of the PR k-d-tree are the *bucket PR k-d-tree* [Orenstein 1982] and the *PMR k-d-tree* [Nelson and Samet 1986]. In the latter, if a region contains more than b data points, the region is split once, and only once (even if a resulting cell contains more than b data points) [Samet 2005, p. 74f]. Another interesting variant of the PR k-d-tree is the *sliding-mid-point k-d-tree* [Maneewongvatana and Mount 1999]. In case a trivial split occurs (i.e. one of the sides has no data points), the splitting hyperplane is moved into the direction of the data until the first data point is encountered. One resulting cell contains this point, the other contains the remaining points. Thus, empty cells are avoided [Samet 2005, p. 72f]. The *bintree* [Knowlton 1980] is the counterpart of the region quadtree and is therefore a trie-based k-d-tree that is used for region data [Samet 2005, p. 49]. A variant which takes secondary memory into account—likewise the remaining variants in this paragraph—and which is suited for point data is the *k-d-B-tree* [Robinson 1981]. It combines properties of the adaptive k-d-tree and the B-tree [Gaede and Günther 1998, p. 192]. The data points are stored in the leaf nodes corresponding to buckets of capacity b. The non-leaf nodes (making up the subtrees of the k-d-structure and representing regions of the data space) are grouped into buckets of capacity c (termed region pages). The nodes at the same level of the perfectly balanced tree represent mutually disjoint regions whose union is the complete universe [Gaede and Günther 1998, p. 192]. Possibly, many nodes need to be split when a bucket overflow occurs [Samet 2005, p. 98]. The *hybrid tree* [Chakrabarti and Mehrotra 1999] is a modification of the k-d-B-tree that reduces the number of split nodes when

an overflow occurs. Basically, it slightly relaxes the disjointness requirement and keeps track of some extra information such that two partition lines and the partition axis are associated with one partition region. Furthermore, quantized MBRs are utilized for the page regions defined by the k-d space partition to eliminate indexed dead space (also see section 4.3) [Samet 2005, p. 101f]. The *SKD-tree* [Ooi et al. 1987] can store extended objects by allowing regions to overlap. The *extended k-d-tree* [Matsuyama et al. 1984] is a clipping-based variant that can handle polylines and polygons.

Another multidimensional data structure assignable to the k-d-tree family is the *LSD tree* [Henrich et al. 1989; Henrich 1990]. It also takes secondary memory into account. Its directory is organized as an adaptive k-d-tree[34], i.e. the universe is partitioned into disjoint cells with possibly varying sizes. Two different split strategies are applied. The *data*-dependent strategy tries to achieve a balanced, generalized split in a local split decision, i.e. it takes the data located in the region to split into account. The *distribution*-dependent strategy splits at fixed dimensions and positions as it assumes an underlying distribution. Therefore, the actual data located in the region is not considered [Gaede and Günther 1998, p. 193f]. The *LSD^h-tree* [Henrich 1998] introduces quantized MBRs to overcome empty indexed regions of the data space. Generally, the split axes cycle through the dimensions except in case there are too few distinct values in the current dimension (which is then left out) [Böhm et al. 2001, p. 46ff]. A related data structure is the *buddy tree* [Seeger and Kriegel 1990] which is a dynamic hashing scheme with a tree directory (also termed hash tree). The universe is recursively split into halves by means of $(d-1)$-dimensional iso-oriented hyperplanes. In each interior node, the MBR of the points or intervals below are stored. Following the terminology of [Samet 2005], it is a variant of a k-d-B-trie which is not always balanced to prevent the existence of page regions with only one child. The buddy tree avoids overlap in the directory nodes by using a generalization of the buddy system (i.e. two adjacent regions can be merged into a rectangular region if the joined region can be obtained by a regular binary division of the universe).

The *BSP-tree* [Fuchs et al. 1980] (BSP for binary space partitioning) is another data structure bisecting the space by means of $(d-1)$-dimensional hyperplanes. In contrast to the other k-d-methods presented so far, arbitrary orientations are allowed for the splitting hyperplanes. A direction must be associated with each hyperplane to distinguish between the 'left' and the 'right' subtree [Samet 2005, p. 66]. Each division is independent of its history and other subspaces. Usually, the decomposition continues until the number of objects in each subspace is below a given threshold. Therefore, the tree is not necessarily balanced [Gaede and Günther 1998,

[34] Furthermore, the directory is distinguished into an internal directory (which resides in main memory) and an external directory (which is stored on disk)—but further details are out of the scope of this work.

p. 182]. In two dimensions, every region is a convex polygon. The BSP-tree is only suitable for storing a collection of (unrelated) line segments [Oosterom 1999, p. 387]. Nevertheless, there are many possible adaptions for point data [Samet 2005, p. 67]. The *multi-object BSP-tree* [Oosterom 1990a] is an extension of the BSP-tree which can represent objects such as polygons. In the *conjugation tree* [Edelsbrunner and Welzl 1986] storing point data, an arbitrarily oriented $(d-1)$-dimensional hyperplane is used which bisects the space S into the three subsets S_{left}, S_{right}, and S_{on}, corresponding to points that are left of, right of, and on the bisecting hyperplane. S_{on} must not be empty if the number of data points is odd. In the next step, a new hyperplane that *simultaneously* bisects each of the two sets S_{left} and S_{right} has to be found. Therefore, there is no independent split for sibling nodes [Samet 2005, p. 88].
The *BD-tree* [Ohsawa and Sakauchi 1983] (for binary division) is also assignable to the k-d-tree family. It bisects the space by means of $(d-1)$-dimensional hyperplanes. The bucket regions are encoded with bit strings. For two dimensions, it complies to a z-ordering with alternating dimensions x, y as well as the literals '0' and '1' corresponding to '$<$' and '$\geq$'. The bit strings are associated with one of the interior nodes of the BD-tree. Also termed discriminator zone expressions (DZE), the bit strings are of variable length and give freedom in the choice of which data items appear in the left and right subtrees (and hence enable the creation of a more balanced tree). The left child is assessed when the sequence of tests for the bit strings of the interior node matches, the right child is assessed when the test does *not* match. Concerning the space partitioning, let us assume a node α represents a subspace x and is labelled with DZE s. Then, the left child of α corresponds to the region formed by the intersection of x and the space represented by s whereas the right child of α corresponds to the region formed by the intersection of x and the complement of the space represented by s. Therefore, only the left child corresponds to a rectangular region whereas the region of the right child is not necessarily rectangular shaped. The overall structure is similar to a PR k-d-tree even though the interior nodes contain more information and not all regions have to be hyperrectangles [Samet 2005, p. 79ff]. The *BBD-tree* [Arya et al. 1998] (for balanced box-decomposition) is closely related to the BD-tree but differs by the shapes of the regions: they are hyperrectangles or the set-theoretic difference of two hyperrectangles where one is enclosed within the other. The division of the space is achieved by split and shrink operations. A split partitions its region by means of an iso-oriented hyperplane (midpoint split, orthogonal to the largest side of the cell). A shrink partitions its region by a box that lies within the original region, resulting in an inner box and an outer box (which is no longer rectangular as the inner box is cropped from it) [Samet 2005, p. 83ff]. The *BANG file* [Freeston 1987] (for balanced and nested grid) can be seen as a paginated version of the BD-tree [Gaede and Günther 1998, p. 195]. With its regular decomposition of the space while

cycling through the dimensions, it can also be regarded as a variant of a k-d-B-*trie* [Samet 2005, p. 114]. The *hB-tree* [Lomet and Salzberg 1989] (for holey brick) is similar to the BANG file but utilizes a data-dependent split strategy such that each region page is at least one-third-full. It can therefore also be seen as a variant of a k-d-B-tree [Samet 2005, p. 97,114]. The *GBD-tree* [Ohsawa and Sakauchi 1990] (for generalized BD-tree) is a paginated version of the BD-tree for extended objects.
The *ATree* [Bogdanovich and Samet 1999] is in some way residual as it can be regarded as a hybrid of a quadtree and a k-d-tree. The ATree is the result of the insight that in some applications, there is a need for the space partition to be finer along a certain subset of dimensions while for the remaining subset, the partition needs to be coarser [Samet 2005, p. 220]. This is achieved by stipulating that all blocks at the same depth i of the ATree are partitioned into 2^{c_i} ($1 \leq c_i \leq d$) congruent blocks (instead of steadily 2^d blocks like in a quadtree or 2^1 blocks like in the k-d-tree). At this point, we conclude the review of the approaches belonging to the k-d family.

R-tree-Based Approaches. One of the biggest families of multidimensional data structures is the *R-tree* family. The original R-tree [Guttman 1984] takes secondary memory into account and has been designed for extended objects. It is easily adaptable to point data, though. The tree structure corresponds to a balanced hierarchy of nested boxes (d-dimensional intervals) which are used to index page regions. The boxes are minimal approximations, i.e. they are MBRs. Each face contains at least one data point or a face of an extended object (if the data objects are for example iso-oriented rectangles).[35] Generally, in the tree structure, each internal node v has between m and M children ($m \leq \lceil M/2 \rceil$). For each child, an MBR is stored in v. The leaf nodes maintain the actual data points. When a node overflow occurs, the node has to be split into two new nodes. In case it is a leaf node, the overflowing node's associated set of data points has to be appropriately divided into two disjoint groups. Diverse node splitting algorithms exist for this purpose. The R-tree is based on data partitioning, i.e. the applied node splitting algorithm directly divides the set of data points into groups—or alternatively aggregates the data points into an object pyramid based on their spatial locations—instead of partitioning the data space. Hence, the data space is described neither completely nor disjoint [Böhm et al. 2001, p. 38]. The general goals of a node split are to minimize coverage (i.e. the total surface area indexed by the MBRs) and overlap (i.e. the area covered by *both* nodes' MBRs)—which can be contradictory [Samet 2005, p. 282f]. The node splitting algorithms are typically described for iso-oriented rectangles as underlying data objects. Usually, they are easily adaptable to point data (with similar runtime characteris-

[35] Note that in the following, the discussion of the R-tree structure is restricted to point data.

tics). In his original R-tree paper, Guttman proposed three different node splitting algorithms for splitting nodes. The exhaustive algorithm tries all possible combinations of object assignments (with costs of $\mathcal{O}(2^M)$). The quadratic as well as the linear cost algorithm both utilize a two-step approach based on picking seed objects at first and assigning the other objects afterwards. Several proposals for improvements exist, differing by runtime characteristics and/or the number of goals attempted to meet. For example, the algorithm of Becker et al. ([Becker et al. 1991])[36] finds the optimal split requiring only costs of $\mathcal{O}(M^3)$. Greene's algorithm ([Greene 1989]) is another split strategy proposal and is reported to yield similar performance as the quadratic cost algorithm [Böhm et al. 2001, p. 39]. Besides handling page overflows by appropriate node splitting, the insertion operations are most important for the performance of the R-tree. Ordering-based, partly static aggregation techniques exist for the R-tree. Examples are the (static) *VAMSplit R-tree* [White and Jain 1996a] or the (dynamic) *Hilbert R-tree* [Kamel and Faloutsos 1994]. They are not directly relevant for our work, though.

A much more important variant is the *R*-tree* [Beckmann et al. 1990]. It introduces significant changes to the R-tree construction algorithm, including a different node splitting strategy. Similar to the linear and quadratic strategies of the original R-tree, a two-step approach is used. The nature of the steps differs, though, as in the first step, the split axis is determined and in the second step, the split position is determined. The other changes involve a revised strategy for the insertion of new data objects and the endeavor to avoid node splits by forced reinsertion. The *RR*-tree* [Beckmann and Seeger 2009] (for Revised R*-tree) presents an update of the algorithms for the R*-tree. The R^+*-tree* [Sellis et al. 1987] avoids overlap at the expense of more nodes and multiple references to some objects and thus is a clipping-based R-tree variant. The *X-tree* [Berchtold et al. 1996] is a R-tree variant for high-dimensional spaces. It introduces two kinds of splitting policies, both of which result in splitting along a single dimension: For the first, it is tried to minimize a combination of perimeter and overlap. For the second (which is applied if the overlap of the nodes resulting from the split exceeds some threshold), the node is split along the dimension that was used when the first two children of the node were created. If the split is unbalanced, a supernode (which can occupy two or more disk pages) is created instead of splitting [Samet 2005, p. 566].

Generally prevailing as most widely used family of data structures for spatial data, the approaches discussed above are by far not all of the existing R-tree variants. Multiple surveys on R-tree variants exist—each with a different focus. Hwang et al. present a comparative study on the performance of several main memory R-tree variants ([Hwang et al. 2003]). Half of them make use of the so-called quantized relative minimum bounding rectangle

[36] Note that the RecMAR$_{k,sl}$ approach described in section 4.1 is based on this algorithm.

(QRMBR) technique which is the compressed, approximated representation of an MBR allowing more entries in an R-tree node (i.e. the fanout is increased). A similar technique has been proposed independently for the *A-tree*[37] [Sakurai et al. 2002]. Manolopoulos et al. discuss dynamic and static versions of R-trees as well as R-tree variants in spatiotemporal databases and modern applications (such as multimedia databases, data warehousing, and data mining [Manolopoulos et al. 2006]). A more recent study is presented by Balasubramanian and Sugumaran which give an overview of 'state-of-the-art R-tree variants for spatial indexing' ([Balasubramanian and Sugumaran 2012]). The reviewed variants are classified whether they a) changed the general process during any step of the construction, b) are hybrid (i.e. they combine concepts of different index structures such as quadtrees, hash indexes, k-d-trees, Voronoi diagrams, etc. with concepts known from the R-tree or R-tree variants)[38], or c) are extended to store additional information such as temporal information.

There are also several R-tree modifications that utilize polygons for the page regions.[39] In the *JP-tree* [Jagadish 1990], m fixed orientations exist ($m > d$). Objects are approximated by bounding polytopes whose faces are parallel to the m orientations. Therefore, a d-dimensional polytope can be mapped into an m-dimensional box. Then, many multidimensional data structures can maintain the polytopes, e.g. the R-tree [Gaede and Günther 1998, p. 208f]. In the *SP-tree* [Schiwietz 1993], an unbounded number of vertices for convex polygons can be stored. There is a conflict between using more vertices for 'good' approximations and a guaranteed minimum fanout. In diverse studies, pentagons or hexagons are presumed to offer the best trade-off [Gaede and Günther 1998, p. 209f]. The *cell tree* [Günther 1989] is similar to an R^+-tree but uses convex polytopes instead of bounding rectangles [Samet 2005, p. 312]. The convex polytopes are restricted to subspaces of a BSP to avoid some of the disadvantages resulting from clipping. It can therefore also be seen as a BSP-tree mapped on paged second memory [Gaede and Günther 1998, p. 215f].

Multidimensional data structures utilizing hierarchies of nested d-dimensional spheres also exist. The *sphere tree* [Oosterom 1990b] is very similar to the R-tree but requires less storage space to index areas and is orientation-insensitive. Nevertheless, the computation of the minimum bounding sphere is considerably more difficult compared to the MBR [Oosterom 1999, p. 395]. The *SS-tree* [White and Jain 1996b] also utilizes spheres as page regions but they are no longer minimum bounding spheres. Instead, it uses the centroid point (i.e. the average value in each dimension)

[37] Not to be confused with the ATree of Bogdanovich and Samet mentioned before.

[38] Examples are the Hilbert R-tree or the *VoR-tree* [Sharifzadeh and Shahabi 2010] which incorporates Voronoi diagrams into an R-tree.

[39] Originally, both of the following variants (JP-tree and SP-tree) have been proposed using the term *P-tree*. We use the terminology introduced in [Gaede and Günther 1998] to differentiate between them.

and chooses the minimum radius to include all data objects. When an overflow occurs, a forced reinsert is raised (like in the R*-tree). For a split, the split axis is determined by the highest variance. The split plane is determined by encountering all possible split positions which fulfill space utilization guarantees. Then, the sum of variances on each side of the split plane is minimized [Böhm et al. 2001, p. 48f]. The *SR-tree* [Katayama and Satoh 1997] utilizes an intersection between an MBR and a sphere (minimum sphere around the centroid point of the stored objects) as a page region. Therefore, $2d + d + 1$ values are required for the representation of a page region. Its motivation is that on the one hand, spheres are better suited for NN and range queries in Euclidean distance but on the other hand, they tend to overlap in splitting and are difficult to maintain. Consequently, the SR-tree shall overcome both shortcomings. It can also be seen as combination of the R*-tree and the SS-tree [Böhm et al. 2001, p. 51ff]. With the SR-tree, we conclude the review on data structures associable with the R-tree family.

Hybrid Approaches. There are also hybrid structures which are not clearly associable to one of the big data structure families. The *SH-tree* [Dang et al. 2001] (for super hybrid) is a structure combined of k-d-tree-based techniques, the SS-tree, and the SR-tree. It employs a SKD-tree-like representation for partitions of internal nodes, i.e. the k-d partitions may overlap. 'Balanced nodes' are just above the leaf nodes and have a similar structure to that of internal nodes in the SR-tree. A leaf node in the SH-tree has the same structure as that of the SS-tree. The authors claim that the SH-tree overcomes the respective shortcomings of data partitioning and space partitioning approaches while making use of their respective positive aspects. Since no fundamentally new methods to divide a data point set into groups or to index areas are introduced, we already conclude the review of hybrid approaches at this point.

Voronoi-Diagram-Based Approaches. Some multidimensional data structures are based on Voronoi diagrams. The *D-tree* [Lee et al. 2004] indexes data regions (i.e. Voronoi cells) based on the divisions between them, i.e. they are neither decomposed nor approximated. The D-tree can be seen as a generalization of the BSP tree where the subdivision lines are poylines (sequences of connected line segments of arbitrary orientation) instead of being restricted to straight lines [Samet 2005, p. 68]. Generally, the solution space for nearest neighbor problems can be precomputed by using a Voronoi diagram [Lee et al. 2004, p. 3]. Therefore, some Voronoi-based indexing methods try to take advantage of the Voronoi partition of the underlying space to facilitate finding nearest neighbors in high dimensions [Samet 2005, p. 566]. The *os-tree* [Maneewongvatana and Mount 2001] is such a method. Again, it can be seen as a generalization of the BSP-tree. Each node δ in a BSP-tree is associated with a cell and the subset of points S_δ in this cell. In the os-tree, each node additionally is associated with a

convex polytopal region called a cover. The cover is constructed to contain a significant *fraction* of the Voronoi cells of the points of S_δ. Computing the exact Voronoi cells is too expensive in high-dimensional spaces. Therefore, the covers are computed with the aid of a large set of training points. The covers are not stored explicitly but are defined by a set of bounding hyperplanes stored at the nodes of each of the ancestors.

Data Structures for High(er)-Dimensional Data. There are some further attempts to adapt multidimensional data structures for high-dimensional data. In e.g. the *TV-tree* [Lin et al. 1994], regions are described by 'telescope vectors' which may dynamically be shortened [Böhm et al. 2001, p. 50], i.e. a fixed number of attributes are tested whose identities can be varied from level to level and from node to node at the same level [Samet 2005, p. 572]. This means its deciding characteristic is that not all dimenions are used to construct the tree but only those which are 'promising' for pruning and discrimination [Manolopoulos et al. 2006, p. 121]. The *pyramid tree* [Berchtold et al. 1998] maps d-dimensional points into a one-dimensional space and then uses a B^+-tree to index the one-dimensional space. It is particularly designed for medium to high dimensionality, starting from $d = 10$ [Böhm 2008, p. 929]. In the first step, the d-dimensional data space is partitioned into $2 \cdot d$ pyramids which have the center point of the data space as their top. In the second step, the single pyramids are cut into slices parallel to the basis of the pyramid. Each slice forms a data page. For acquiring the one-dimensional code, the pyramids are numbered. The number of the pyramid and the height of a point in the pyramid (orthogonal distance to the center of the data space) are merged to the one-dimensional code which is used for indexing [Böhm et al. 2001, p. 55f]. The *extended pyramid technique* for non-uniform data takes the d-dimensional median of the data as a center point [Samet 2005, p. 589]. Nevertheless, for low-dimensional data, approaches designed for or adapted to higher dimensional data are mostly only suboptimally applicable. Hence, we conclude the review on data structures for high-dimensional data.
This also completes the review of multidimensional data structures. There are many more approaches, but the most important families as well as some of their most popular and interesting representatives are covered. In section 4, we discuss how the different approaches for dividing data point sets into groups and indexing areas are converted into summarization approaches.

2.5. Querying for Spatial Data

As shown in section 2.1 to section 2.4, the field of possible approaches for designing resource descriptions is wide. Ultimately, resource descriptions are used to facilitate effective and efficient query processing in a spatial database management system. In this section, we focus on the specific characteristics of queries for spatial data. In particular, the conceivable query

types (especially kNN queries) and applicable distance functions for ordering a set of descriptions for kNN queries are discussed.
In general, when designing a search system, two of the most important questions are which query types shall be supported and how they shall be specified. Of course, this is heavily dependent on both the application scenario and the data itself. In classical relational database management systems, SQL exists as a standard query language which allows the specification of more or less arbitrary queries (though most of the specific systems also provide proprietary extensions). Also, a standard relational algebra exists.
In contrast, for spatial databases, neither a standard spatial algebra nor a standard spatial query language exist. Thus, the specification of spatial queries is always system-dependent, and the set of operators strongly depends on the application domain [Gaede and Günther 1998, p. 176]. Nevertheless, there are some common query types for spatial data. Depending on whether the subject of interest is point data or extended data (such as lines or regions), the types of queries and their definitions (partly) differ. Since our focus is on point data, we concentrate on the point data query types in the following.
Generally, there are three main query types for spatial data: spatial range queries, nearest neighbor queries, and spatial join queries [Ramakrishnan and Gehrke 2003, p. 70]. Both spatial range queries and nearest neighbor queries can also be classified as spatial selection queries, i.e. a set of (point) objects that satisfy the query criteria is selected from the database [Ahn et al. 2001, p. 3].
A *point query* is an exact match query which specifies a point in the data space and retrieves all point objects in the database with identical coordinates. A *range query* is defined by a query point q, a distance q_{rad}, and a distance function D. The result set contains all point objects whose distance to q is smaller than or equal to q_{rad} according to D. In a *window query*, an iso-oriented rectangular region of the data space is specified from which all point objects are retrieved [Böhm et al. 2001, p. 10f]. The point query can be seen as a specialization of the range query and the window query, and all three query types fall into the *spatial range query* category. Also, they can all be simplified to determine only the boolean answer whether the database contains corresponding data points.
The *nearest neighbor query* (NN query) returns the one point object in the database with the smallest distance to the query point. If there are several nearest neighbors, it is common to choose an arbitrary object from this set. For the k *nearest neighbor query* (kNN query), a natural number k of closest points is retrieved. A kNN query can be *precise* (sometimes also termed *exact*) or *approximated*. For the latter, the user is not stringently interested in the exact closest points but demands points which are not much farther away than the exact kNN. The requirement of 'only' approximately correct results can be used for improving the efficiency of the query processing

[Böhm et al. 2001, p. 11f]. A *ranking query* or *ranking enumeration query* is a generalized kNN query with a previously unknown k ('give-me-more'-query). The request stops according to a criterion which is external to the index-based query processing. Therefore, neither a limited query range nor a limited k can be assumed before the application terminates the ranking query. In a *reverse (k)NN query*, given an arbitrary point q, all point objects of the database for which q is (one of) the (k) nearest neighbor(s) are retrieved. This (k)NN relation is not symmetric. Therefore, the result set for reverse (k)NN queries can be empty or contain an arbitrary number of data points [Böhm et al. 2001, p. 29f].
For *spatial joins*, given two collections of spatial objects R and S, a spatial property P, and a spatial predicate θ, all pairs of spatial objects $(o, o') \in \mathtt{R} \times \mathtt{S}$ where θ ($o.P$, $o'.P$) evaluates to *true* shall be retrieved [Gaede and Günther 1998, p. 177]. This query type is more common for extended data, though.
A more recent query type apart from the three main types is the *spatial skyline query* [Sharifzadeh and Shahabi 2006]. Given two d-dimensional points p and p', a distance metric D which obeys the triangle inequality[40], and a set $Q = \{q_1, \ldots, q_n\}$ of d-dimensional query points, p spatially dominates p' iff every q_i is as close or closer to p than to p' and at least one q_i is closer to p than to p'.
Generally, (k)NN queries are the most common and most important query type. The efficient processing of (k)NN queries requires spatial descriptions which capitalize on the proximity of the objects to limit the search to potential neighbors only. Most (k)NN algorithms for centralized, hierarchy-based multidimensional data structures apply either a best-first search or some sort of ordered traversal (possibly in a branch-and-bound approach) through the indexing structure [Samet 2008, p. 1]. Traditionally, they are based on lower bounds—though upper bound pruning techniques were also suggested (see for example [Samet 2008]). Several efficient algorithms have been proposed in the centralized context. One example is the algorithm by Hjaltason and Samet ([Hjaltason and Samet 1995]) which was specifically designed for paged multidimensional data structures. Roussopoulos et al. propose an efficient branch-and-bound algorithm for processing precise kNN queries which is based on a depth-first traversal [Roussopoulos et al. 1995]. The algorithm is primarily described for the R-tree. Nevertheless, it is generally applicable to all other multidimensional data structures based on intervals. Furthermore, Roussopoulos et al. introduce two distance functions usable to guide nearest neighbor searches, the MINDIST and the MINMAXDIST. In the following, we shortly outline both distance functions assuming that data point sets are delineated by MBRs.
Given a query point q and a set of data points S which is enclosed by an MBR m, every face of the hyperarea defined by m must contain a point

[40] The triangle inequality stipulates that for any three points x, y, and z in the data space, it must hold that $dist(x, z) \leq dist(x, y) + dist(y, z)$.

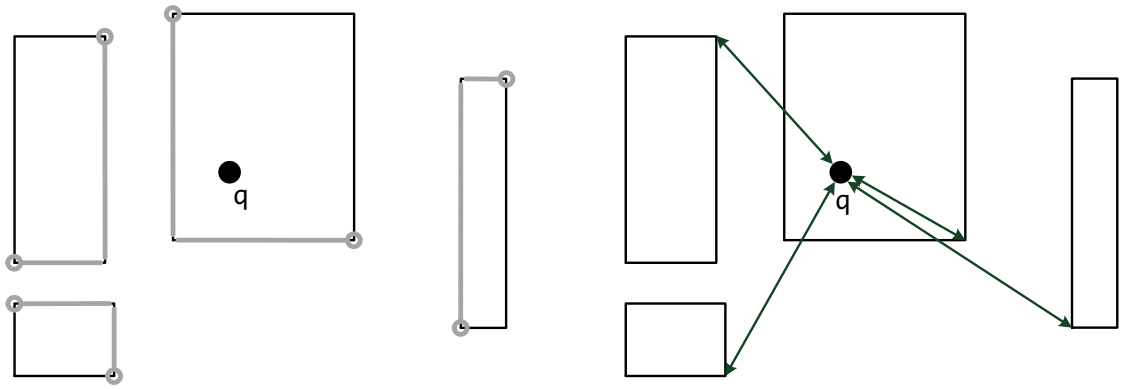


Fig. 6: Example of the MINMAXDIST. On the left, the light-grey lines represent the closest face to the query point q in each dimension (for each of the four MBRs). On the right, the four respective MINMAXDIST distances between q and the MBRs are depicted (by the arrows). This depiction is based on Figure 10 of [Böhm et al. 2001, p. 23].

$p_i \in S$. The MINDIST is an optimistic estimate of the minimum distance between q and a point $p_i \in S$ as it is the minimum distance between q and the MBR m (if q is inside m, the distance is 0). Thus, the MINDIST is a lower bound of any kNN distance. In contrast, the MINMAXDIST is a pessimistic estimate of the minimum distance between q and a point $p_i \in S$. It is the minimum over all dimensions' distances from q to the furthest point of the closest face of the MBR m. See Figure 6 for an example visualization. Thus, the MINMAXDIST is the smallest possible upper bound of distances from the point q to the MBR m. It is guaranteed that there is a point p_i within the MBR m at a distance to q less than or equal to MINMAXDIST. Thus, an ordering of a set of MBRs by the MINDIST is an optimistic ordering whereas an ordering by the MINMAXDIST is a pessimistic ordering. Generally, the MINDIST is applicable for all types of bounding volumes. In contrast, the MINMAXDIST metric can only be applied meaningfully if interval-shaped, non-approximated regions are used.

2.6. Further Related Work

In this section, we gather related works which do not fit into one of the previous sections, either thematically or due to the thesis structure—but which are still relevant or at least noteworthy in the overall context of the work.

For one, this concerns the fields of *P2P multidimensional indexing methods* and *P2P overlay networks* (see section 2.6.2). Although also the former index data based on its spatial properties in a distributed environment, the underyling scenario differs from our *distributed application scenario* in that the assignment of the data to the single resources (i.e. to the peers of the P2P network) is controllable and conducted on basis of the data's spatial properties. In contrast, for our distributed application scenario, we assume that the assignment of the data to the resources is non-controllable

or determined by non-spatial properties. Furthermore, indexing concepts designed for *metric* (see section 2.6.3) or *ptolemaic* (see section 2.6.4) spaces are discussed. Both are more generally applicable compared to coordinate-based multidimensional concepts. Nevertheless, neither metric nor ptolemaic indexing approaches can be used to immediately derive any new concepts for the design of resource summaries for spatial data. The same applies for the *P2P multidimensional indexing methods*. There, in general, approaches known from the centralized multidimensional data structures are adapted to the specific application scenario rather than introducing any new indexing approaches. Consequently, section 2.6.2 to section 2.6.4 can be assigned to the category of 'notable related work'.
The situation is different for the *linear quadtree encodings* which are discussed in section 2.6.1. They allow for the storage-space-efficient representation of quadtree structures. Thus, on the one hand, the *linear quadtree encodings* would fit into the compression chapter (see section 2.1). On the other hand, they are misplaced there without a quadtree discussion, first. Consequently, they are in this section.

2.6.1. LINEAR QUADTREE ENCODINGS. Hierarchical implementations of quadtrees[41] use pointers to nodes and are therefore storage-space-intensive. To alleviate this problem, the linear storage of quadtrees has been proposed. A linear representation of a quadtree is a list of values which stores the hierarchical tree structure [Manouvrier et al. 2002, p. 4f]. As already noted in section 2.3.3, the use of linear quadtrees has been extensively researched in the field of representing binary images. Binary images are $n \times m$ binary arrays (for any positive integers n, m) where an '1' ('0') entry stands for a black (white) picture element [Tzouramanis et al. 1998, p. 2]. Linear quadtrees are based on a regular space decomposition—which is natural in the rasterized image domain—and are therefore also applicable to *trie*-based quadtree variants in the vector space domain (where black nodes commonly contain *some* data).
Generally, there are two categories of linear quadtrees. For the first category, the quadtree is seen as a collection of leaf nodes which contain data, i.e. only the black leaf nodes are (separately) encoded. Each black node is identified by a unqiue key derived from its ordered list of ancestors and the list of node identifiers is sequenced by keys [Manouvrier et al. 2002, p. 4f]. For the ordering, any space-filling curve is applicable, in general. It is common to use a z-ordering, though. Regardless of the ordering, the value of a black node is an address describing the position and the size of the corresponding quadtree region [Tzouramanis et al. 1998, p. 3]. Examples for linear quadtree encodings of this category are the *linear quadtree*

[41] This does not refer to the correspondingly named family of multidimensional data structures but to the quadtree structure representing the split hierarchy of a quadtree space partition. The same goes for the remainder of this thesis whenever we refer to quadtrees.

[Gargantini 1982], the *fixed length* (FL), the *fixed length-depth* (FD), and the *variable length* (VL) linear implementations [Samet 1990], or an (unnamed) encoding described by Abel [Abel 1984].[42]
For the second category, the entire quadtree structure with all its leaf nodes and internal nodes is encoded. The encodings correspond to a traversal of the quadtree to encode, i.e. the quadtree nodes are encoded in the order in which they are encountered when traversing the quadtree. Examples are the *CBLQ code* [Lin 1997], the *Compact-IQ code* [Yang et al. 2000], or the *DF-expression* [Kawaguchi and Endo 1980].
For both categories, it can be distinguished whether the encoding of the node values follows a depth-first order (df-order) or a breadth-first order (bf-order) traversal [Manouvrier et al. 2002, p. 4].

2.6.2. P2P MULTIDIMENSIONAL INDEXING METHODS AND P2P OVERLAY NETWORKS. In this section, we briefly outline two topics related to administering (spatial) data in a distributed search environment: peer-to-peer (P2P) multidimensional indexing methods and P2P overlay networks.

P2P Multidimensional Indexing Methods. There is also research to support multidimensional queries in P2P networks. With respect to the multidimensional data structures discussed in section 2.4.3, this marks a paradigm shift from the centralized domain to the distributed domain. As a motivation, Zhang et al. state that on the one hand, centralized multidimensional data structures need to be decentralized to achieve high scalability and that on the other hand, P2P systems need to be equipped with complex query processing capabilities for multidimensional data [Zhang et al. 2011, p. 2348f]. Consequently, P2P multidimensional data structures are regarded to advance both domains. The basic principle behind the utility of P2P multidimensional data structures is still pruning. In the distributed domain, this results in reducing the communication costs (instead of reducing the I/O costs like in the centralized domain). There are several surveys on P2P multidimensional data structures such as [Bongers and Pouwelse 2015] or [Zhang et al. 2011].
Bongers and Pouwelse divide the approaches into two general classes [Bongers and Pouwelse 2015]. The first class of approaches reduces the dimensionality of the data to index—usually with space-filling curves or locality preserving hashing (where the dimensions are directly mapped to a linear space)—and then use distributed hash tables (DHTs) or skip graphs (which adapt the concept of skip lists to a P2P system) for indexing. For example, *SCRAP* [Ganesan et al. 2004] applies a z-ordering and a skip graph, *CISS* [Lee et al. 2007] and *Squid* [Schmidt and Parashar 2003] apply a Hilbert ordering and DHTs, and *MAAN* [Cai et al. 2004] uses locality preserving hashing with DHTs. The second class of approaches partitions

[42] Note that for only black leaf nodes encodings, structures like the B^+-tree can be used to store the black node codes of the linear quadtree, making it a spatial access method usable for points, lines, or regions [Yin et al. 2011, p. 117].

the multidimensional space. For example, *MURK* [Ganesan et al. 2004] applies a k-d-*tree* space partition (likewise the adaptive k-d-tree). A leaf in the tree structure then corresponds to a node in the P2P network. Furthermore, Bongers and Pouwelse showcase an evaluation of the P2P multidimensional data structures with respect to algorithmic complexity, load balance, robustness, security, and 'practical considerations'. Zhang et al. present multiple classifications differing in the standpoint of view [Zhang et al. 2011]. For example, from the 'view of evolution', it is distinguished between extensions from centralized multidimensional data structures (like the *P2PR-tree* [Mondal et al. 2004] or the *NR-tree* [Liu et al. 2005] which is an R*-tree adapted to the P2P domain), approaches extending the P2P domain (like MAAN or MURK), and combinations of both (like Squid).
The P2P multidimensional data structure approaches are in control of how the multidimensional data is distributed to the nodes in the network. For our distributed application scenario, in contrast, non-spatial properties or simply the origin of the data decide on the assignment of the data to the nodes in the network. Therefore, there is a big conceptual difference between both application scenarios. Furthermore, the P2P multidimensional data structures do not introduce new approaches to data partitioning or space partitioning but adapt well-known concepts of the centralized domain to the distributed domain. Consequently, we do not further pursue this field here and refer to the mentioned survey articles for further reading.

P2P Overlay Networks. In a P2P network, different types of architectures (also called P2P overlay networks) are employable. They refer to how the peers are connected to each other. Resource description (and selection) schemes are generally applicable in all of them. Lu distinguishes four general architectures, namely *brokered*, *completely decentralized*, *hierarchical*, and *structured* network architectures [Lu 2007]. The first three categories are sometimes also collectively referred to as *unstructured* architectures [Lu 2007, p. 6]. In this work, we focus on the design of resource summaries and are not immediately concerned with the distribution mechanisms required for certain P2P overlay networks. Therefore, the interested reader is referred to [Lu 2007, ch.2] for further information.[43]

2.6.3. METRIC ACCESS METHODS. Related to multidimensional data structures are *metric access methods* (MAMs). In the spatial domain, objects are represented as vectors of a coordinate space of a fixed dimensionality. In the metric domain, less is known about space. More precisely, only distances between objects are known. Hence, in contrast to the mul-

[43]Note that the approaches supporting multidimensional indexing in P2P networks can also be classified based on the four different general P2P architecture categories in which they fall into. For example, MURK would fall into the structured category whereas the NR-tree would fall into the hierarchical category. Also see [Zhang et al. 2011, table 4] where the topic is discussed in greater detail.

tidimensional approaches (where strong assumptions about the data are made), the metric approaches work directly with distances and are generally applicable to a broader class of distances [Hetland 2015, p. 165]. There are many distances that satisfy the metric axioms or properties, i.e. non-negativity, identity of indiscernibles[44], symmetry, and the triangle inequality (see [Zezula et al. 2006, p. 8]). In general, MAMs are used in any areas that require efficient query-by-example but where traditional (coordinate-based) multidimensional data structures are not applicable—since the data is of very high dimension or nothing beyond pairwise distances between objects can be measured (sometimes deemed 'distance-only data' [Zezula et al. 2006, p. 3]). Thus, MAMs have a different (possibly wider) field of applications.
Since no coordinate information is available in metric spaces, sites[45] and distance information are used to partition the space. There are two basic decomposition schemes [Blank 2015, p. 23ff]:

— The generalized hyperplane partitioning where the universe $\mathtt{U}$ is partitioned into two subsets by a (implicit) hyperplane between two corresponding sites s_1 and s_2 ($s_1, s_2 \in \mathtt{U}, s_1 \neq s_2$). The multiway generalized hyperplane partitioning is an extension of the generalized hyperplane partitioning with a set of more than sites [Hetland 2009, p. 212]. It results in a Voronoi-like space partition.
— The ball partitioning where $\mathtt{U}$ is partitioned into two subsets by a site s ($s \in \mathtt{U}$) and a covering radius r^{out}. The latter is an upper bound of the distance $dist(s, x)$ from the center s to any object $x \in \mathtt{U}$ in the ball region. Closely related is the concept of a shell where the center s and two radii are captured: the inner radius r^{in} (which is a lower bound for any object $x \in \mathtt{U}$ in the ball region) and the outer radius r^{out} [Blank 2015, p. 29f].

The triangle inequality is used to define lower bounds (i.e. pruning rules) which allow to filter out irrelevant objects from the search cheaply, i.e. without the actual distance computations which are assumed to be highly expensive [Hetland et al. 2013, p. 990]. Therefore, the focus of metric access methods is usually on minimizing the number of distance computations before reducing I/O or general CPU time (like for the multidimensional data structures).
In metric spaces, cluster pruning (excluding certain regions of the data space and thus all the objects within the region from search) is one means of optimizing the query processing [Blank 2015, p. 32ff]. It is also interesting in the context of Voronoi-like space partitions in the spatial domain. Thus, we briefly introduce two corresponding pruning rules. Generally, in the spatial domain, both pruning rules allow for a cheaper pruning of Voronoi cells (and their data points) from search compared to explicitly constructing and testing the polygons of the cells. The trade-off is that the

[44] A distance where *different* objects can have a distance of 0 is called pseudometric.
[45] The sites are also often called reference objects, centers, or centroids.

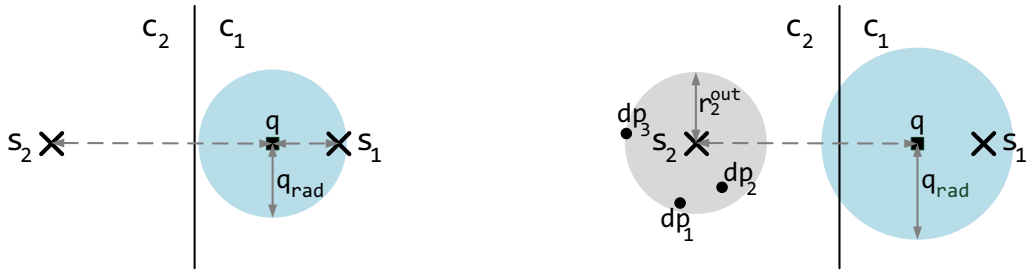


Fig. 7: Example visualizations depicting the succesful application of the Double-Pivot Distance Constraint (left) and the Range-Pivot Distance Constraint utilizing the r_i^{out} radius (right). In both cases, the cell c_2 can be successfully pruned.

pruning rules do not allow for as much pruning as pruning rules using the polygons.[46]

The *Double-Pivot Distance Constraint* is used to prune Voronoi cells by exploiting the triangle inequality. Let c_1 be the query cell (i.e. the closest site to the query point is the site s_1 defining the cell c_1), c_2 be any other cell ($c_1 \neq c_2$), and q_{rad} be the query radius. c_2 can be pruned from search if $(dist(q, s_2) - dist(q, s_1))/2 > q_{rad}$—or, in other words, the query ball does not intersect c_2 (see Figure 7 on the left). The application of the *Range-Pivot Distance Constraint* requires the covering radii r_i^{out} and r_i^{in} which can be captured for each site s_i. r_i^{out} is the maximum distance between s_i and any data point located in the cell c_i, r_i^{in} is the minimum distance between s_i and any data point located in the cell c_i. Pruning rules can be applied based on r_i^{out}, r_i^{in}, or combinations of both. In the following, we limit the discussion to the rule derived from the r_i^{out} radius.[47] Again, let c_2 be any other cell than the query cell c_1. The cell c_2 can be pruned from search by the Range-Pivot Distance Constraint if $dist(q, s_2) - q_{rad} > r_2^{out}$—or, in other words, the query ball does not intersect the cluster ball around s_2 with radius r_2^{out} which contains all the data points of c_2 (see Figure 7 on the right).

In [Hetland 2009, p. 3], it is stated that MAMs may be better suited than multidimensional data structures for indexing in high-dimensional traditional vector spaces because metrics bypass the so-called representational dimensionality and deal directly with the intrinsic dimensionality [Hetland 2015, p.165]. We do not further pursue MAMs at this point since our data is of low dimensionality and it can be assumed that for low-dimensional spaces, using the maximum amount of available information will lead to the best results. MAMs disregard the information provided by the coordinate space in which the multidimensional data is embedded since they only consider distances between objects. Nevertheless, we revisit this topic in the course of this thesis. Some excellent publications that cover the

[46] This issue is discussed in a bit more detail in footnote[102] on page 105.

[47] See [Blank 2015, p. 33ff] for the pruning rules derivable from other covering radii information.

domain of MAMs are [Chávez et al. 2001], [Samet 2005, ch. 4.5], [Zezula et al. 2006], [Hetland 2009], and [Blank 2015].

2.6.4. PTOLEMAIC ACCESS METHODS. Strongly related to metric indexing is a fairly recently introduced field called *Ptolemaic indexing*. Similar to the metric approach, the Ptolemaic approach does not make any kind of coordinate-based assumptions. Yet, it substitutes the triangle inequality with the Ptolemy's inequality.[48] The Ptolemy's inequality is used to construct lower bounds for filtering—greatly increasing the filtering power when applicable. Hetland states that especially higher dimensionalities seem to give Ptolemaic indexing a greater edge over the triangular form of the metric approaches [Hetland 2015, p. 180].
For vector spaces, the Ptolemy's inequality is directly applicable to quadratic form distances (QFDs) which take into account possible correlations between the dimensions of the vector space. The Euclidean distance is a special case of a QFD [Hetland 2015, p. 165]. Many other Ptolemaic distances besides quadratic form distances exist. Furthermore, there are non-Ptolemaic distances which are 'sufficiently Ptolemaic' and can be used for Ptolemaic indexing—although only for approximate queries [Hetland 2015, p. 169f]. Due to the high filtering costs, the Ptolemaic approach is mainly suitable for expensive distances where the extra complexity becomes insignificant. Therefore, it may not be suitable for the (cheap) Euclidean distance as well as static QFDs (since these can be mapped to the Euclidean case). This leaves dynamic versions—such as the signature QFD (SQFD)—as an important field of application.
Note that a distance can be Ptolemaic without being metric and vice versa [Hetland et al. 2013, p. 990]. A Ptolemaic distance for which the triangle inequality additionaly applies is called Ptolemaic metric [Hetland et al. 2013, p. 993]. For Ptolemaic metrics, it is possible to combine upper and lower bounds generated by the Ptolemaic approach with various metric regions which can lead to better results than using either approach on its own [Hetland 2015, p. 174]. Similar to the MAMs, we do not further pursue Ptolemaic indexing at this point and refer to [Hetland et al. 2013] and [Hetland 2015] for additional reading.

[48] Note that Ptolemy's inequality is named after Claudius Ptolemy which we already referred to in the introduction of this thesis (see section 1.1).

Part II:

Distributed Application Scenario

We now approach the core of this thesis—the summarization approaches we develop for the purpose of resource description and selection. This point also marks the beginning of Part II of the work which is dedicated to the main subject of this thesis: The distributed application scenario. It is organized as follows: In section 3, some preliminaries necessary for the subsequent sections are discussed. The various summarization approaches developed specifically for the distributed scenario are described in section 4. Afterwards, the resource selection on basis of resource summaries is showcased (section 5). Then, the setup for the evaluation is outlined in detail in section 6. It is followed by an evaluation of *all* 14 resource description approaches in section 7. In this overall evaluation, a subset of six approaches is selected for a very extensive and detailed further evaluation which is presented in section 8. This further evaluation also concludes the consideration of the distributed application scenario.

3. PRELIMINARIES FOR THE DISTRIBUTED APPLICATION SCENARIO

Before we get to the summarization approaches, we need to consider diverse general preliminaries as well as some specific preliminaries which are important with regard to the concrete implementation of certain summarization approaches. This is all done here. First, some general preliminaries are discussed in section 3.1. Then, some quadtree-specific preliminaries are presented in section 3.2. This is necessary since a lot of our approaches make use of quadtree structures. Finally, we conclude with an excursus on dealing with 'non-perfect' spatial data in section 3.3.

3.1. General Preliminaries

With regard to the distributed application scenario, we concretely assume that every resource or peer in the resource network or P2P network[49] maintains a set of geotagged media items such as images or texts. Geotagged means that each media item is enhanced with a single pair of lat/long-coordinates specifying its geographic context which is the subject of interest here. The geographic context, for example, could be the place where an image has been taken. For a query, the k most relevant media items in the resource network are to be *precisely* retrieved. A query is specified by a pair of lat/long-coordinates. The closer the location of a media item is to the location of the query point, the more relevant the media item is with regard to the query. A resource description is the means to depict the locations of the media items administered by a resource effectively yet efficiently. *Effectively* refers to the accurate description of the locations which allows for the selective targeting of the relevant resources while safely pruning irrelevant resources. *Efficiently* means that the resource descriptions shall be concise, i.e. they must not consume more storage space than is necessary to encode the spatial coordinates of the media items.

The geo-coordinates of the geotagged media items are treated as Plate-Carrée-projected data points. Consequently, the lat/long-coordinates correspond to x/y-coordinates in a two-dimensional Cartesian coordinate system (with *long* being x). Thus, the resource description problem for geographic data is transformed to a more general spatial data problem:[50] The data is of multiple yet low dimensionality ($d = 2$) and all dimensions feature the same type of unit—which is distance in space. This allows for queries which involve proximity [Samet 2005, p. 4]. Therefore, the relevance of media items for specific queries is determined by the spatial proximity of the corresponding spatial data points. Our distributed application scenario can hence also be seen as a similarity search on a set of spatial

[49] Note that in the following, we use the more general wording 'resource' of a 'resource network' to refer to the member nodes of a P2P network.

[50] See section 1.1 again for the explanation of the interrelationship between *geographic data* and *spatial data*.

data points distributed over a set of independent resources. Due to the Plate-Carrée-projection, the data points are encoded in a two-dimensional Euclidean space. Consequently, we use the *Euclidean distance* for distance calculations. This is a common property when dealing with spatial data [Gaede and Günther 1998, p. 174]. Accordingly, the data points' locations are measured from a defined origin ($x = 0$ and $y = 0$) along axes that are perpendicular rather than using angular coordinates defined relatively to a particular Earth geodetic datum[51]. Note that for the given distributed application scenario, Blank and Henrich ([Blank and Henrich 2012]) investigated the utilization of distance measures that are better suited for distance calculations on the Earth's surface, namely the *Vincenty distance* [Vincenty 1975] and the *Haversine distance* [Sinnott 1984]. Nevertheless, their usage did not result in noticeable differences compared to using the Euclidean distance which is computationally less complex and more suitable for the more general spatial approach, anyway.
Following the terminology of Samet ([Samet 2005, p. 2]), the point data in our scenario is bounded (-90 to 90 in $lat = y$ and -180 to 180 in $long = x$), discrete (spatial point data), and of real numbers. Nevertheless, to take into account that the data stems from lat/long-coordinates on the Earth, we consider the International Date Line when calculating distances (point-to-point, point-to-rectangle, point-to-polygon). The consequence is that, for example, the distance between two points $d_1 = (179; 0)$ and $d_2 = (-179; 0)$ is 2 instead of 358.
Our perspective on the multi-database is static, i.e. the number of data points per resource as well as the number of resources do not grow and shrink at will. Also, the identities of the resources do not change, i.e. there is no arrival or departure of resources (also known as *churn* in a P2P network [Stutzbach and Rejaie 2006, p. 189]). Nevertheless, our resource description approaches are qualified for dynamic 'real-world' scenarios. In case the set of data points of a resource alters such that not all of the data points are contained in the spatial extents depicted by the respective resource summary, the resource summary is calculated anew. This also applies if the set of data points of a directly represented resource alters. The churn has to be managed by the overlay network—which is out of scope for this work.

3.2. Quadtree-Related Preliminaries

As mentioned before, we develop various approaches which make intensive use of quadtrees. Before we can utilize quadtrees to design summarization approaches, we need to consider some of their inherent charac-

[51] Geodetic datums define the size and shape of the earth, and the origin and orientation of the coordinate systems used to map the earth [Dana 1995]. Hundreds of geodetic datums exist. A comprehensive list of datums and their properties can be found e.g. at https://www.colorado.edu/geography/gcraft/notes/datum/edlist.html, last visit: 04.07.2018.

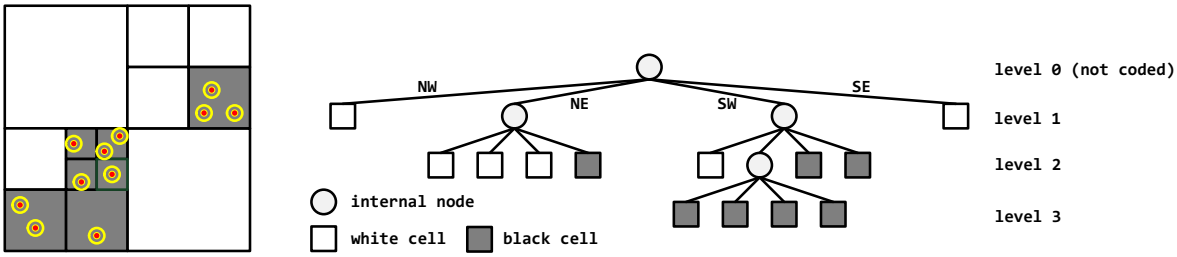


Fig. 8: Exemplary data point set which is described by a trie-based quadtree space partition (left), and the matching quadtree structure (right). The quadtree space decomposition has 13 cells and correspondingly, 13 leaf nodes exist in the quadtree structure.

teristics. This is done in this section. These considerations are valid for all the quadtree-utilizing approaches presented in the further course of this thesis. As already mentioned in the 'Quadtree-Based Approaches'-paragraph[52] of section 2.4.3, there are two basic types of quadtrees:

— The *tree*-based quadtrees. For these, the positions of the splitting hyperplanes are obtained from the values of the data points contained in the quadtree region[53] to split, i.e. possibly arbitrarily-sized cells result. This type of quadtree cannot be efficiently encoded into a single resource's summary since in addition to the quadtree structure itself, the coordinates of the split position have to be captured for each split. Without further adjustments, this would require 32-bit single precision floating-point numbers for our underlying data.
— The *trie*-based quadtrees. These form a regular decomposition of the data space into congruent cells of equal size (see Figure 8). As already discussed in section 2.6.1, the *linear storage* of trie-based quadtrees has been proposed as a very storage-space-efficient representation. For this type of quadtree, it is an option to fit the *entire*[54], linearly encoded quadtree information into a resource's summary.

For the concrete utilization of trie-based quadtrees as a means for local space partitioning, it requires some consideration concerning a) the decomposition of the space into subspaces and b) the concrete linear quadtree encoding which shall be applied. In the distributed application scenario, trie-based quadtrees are used both as a 'solitary' approach (see section 4.2) as well as in combination with other approaches, resulting in several hybrid

[52] This paragraph begins on page 40.

[53] Note that in the remainder of the thesis, we may also use the term 'quadtree cell' to refer to the subspaces of a quadtree space partition. Hence, the terms 'quadtree region' and 'quadtree cell' are equivalent. The usage depends on which term appears to be more suitable in the corresponding context.

[54] This is important for the question whether *local* (i.e. resource-individual) space partitioning can be applied. Conceptually and also in the concrete implementation, it differs strongly from *global* space partitioning (which has already been briefly discussed in the introduction of section 2.4). See section 4.2 for further details on the differences between local and global space partitioning as well as their consequences.

approaches (see section 4.3). For the hybrid approaches utilizing quadtrees, c) the various application possibilities of the trie-based quadtrees are of importance. The topics a) to c) are briefly discussed in the following.

Rules for the Space Decomposition. In general, we are interested in binary quadtrees. Consequently, it is only differentiated between black and white quadtree cells depending on whether they contain data objects (black) or not (white). For binary quadtrees and *region data*, the space decomposition is applied until each resulting cell is homogeneous, i.e. the cells are either fully black or fully white [Samet 2005, p. 211]. Since we are concerned with *point data*, we have to define a sensible 'homogeneity criterion' or rather stopping criterion to abort the quadtree construction at some point because no cell will ever be fully black with point data.
For our approaches, we apply two stopping criterions in combination: The first stopping criterion is applied when a certain amount of quadtree cells c is reached. Thus, it is storage-space-oriented as it limits the number of nodes in the quadtree structure which have to be described in a linear code. Initially starting with one cell for the entire data space to partition, in consecutive steps, the black cell (i.e. a quadtree region containing at least one data point) with the greatest surface area is equally divided into four. The simple assumption is that the greatest black cell also contains the most dead space which can be further minimized by a finer space partition. If there are several equal-sized black cells, a random choice is made. After the divide, the resulting sibling leaf nodes are labelled black or white depending on whether the corresponding cells contain at least one data point or not. The space decomposition continues until an amount of c cells has been reached but ends prematurely if the surface area of each black cell is below a certain threshold surface area a. The threshold surface area a serves as the second stopping criterion and limits the maximum spatial accuracy of a resource summary. Thus, it is a stopping criterion oriented towards the spatial accuracy of a resource summary.[55]

Linear Quadtree Encoding. As already discussed in section 2.6.1, the linear storage has been proposed for a storage-space-efficient representation of quadtrees. For these, a list of values stores the hierarchical tree structure [Manouvrier et al. 2002]. Remember that encoding schemes can be distinguished on the basis of two properties:

— Is the entire quadtree structure encoded (i.e. all leaf nodes and internal nodes), or are only the black nodes encoded (i.e. the black leaf nodes of the quadtree structure)?
— Are the values encoded in a df-order or in a bf-order?

[55] For data point sets which are spatially very narrow (i.e. of a small spatial spread), the adaptive quadtree decomposition can very quickly result in cell sizes below the general GPS accuracy and for which additionally, the limited precision of 32-bit or even 64-bit floating-point numbers can cause problems. The threshold surface area a also helps in dealing with these issues.

Also see [Manouvrier et al. 2002, ch. 3.1] for an overview of linear quadtree encodings. In this work, we evaluate two different encoding schemes: the linear quadtree (LQ code) [Gargantini 1982] and the Constant Bit-length Linear Quadtree (CBLQ code) [Lin 1997].

The **LQ code** is in df-order and encodes only black nodes. As usual for encoding schemes of this class, the black nodes are encoded separately. A black node is identified by a unique key derived from its ordered list of ancestors. The key of successive digits represents the quadrant subdivision from which the black node originates according to a df-traversal. The digits are 0 (for NW), 1 (for NE), 2 (for SW), and 3 (for SE), implicitly corresponding to a z-ordering. Keys consist of up to l digits, where l is the number of levels or depth of the quadtree structure. If a black node is at level i ($i < l$), only i digits are obtained, followed by a marker X. This is an adaption versus the LQ code described by Gargantini ([Gargantini 1982]) where $l-i$ markers would follow. A condensation of the quadtree can be applied—provided it is suitable (see paragraph *Usage* below). Condensation means that four black sibling leaf nodes are conflated into their parent node which therefore becomes a black leaf node itself. For the LQ code, this results in that for these siblings, the last digit is replaced by marker X and the four corresponding codes are pooled into one. For instance, 310-311-312-313[56] becomes 31X. Surplus markers X are removed from the LQ code. As an example for the LQ code, consider the quadtree in Figure 8. It is represented as 13X-210-211-212-213-22X-23X or 13-21-22-23 after condensation. Since there are five different literals to encode (0 to 3 and X), we require 3 bits for the binary representation of a literal.[57]

The **CBLQ code** is in bf-order and encodes all leaf nodes and internal nodes. The authors claim that on average, the required storage space is only 22.2% of the LQ code or 88.5% of the DF-expression [Kawaguchi and Endo 1980] which is in df-order and encodes all leaf nodes and internal nodes. In the CBLQ code, each black (white) leaf node is encoded by the literal 1 (0). The literal 2 encodes an internal node if at least one of its descendants is an internal node. If all descendants are leaf nodes, an internal node is encoded by the literal 3. The root node is not encoded.[58] A condensation of the quadtree can be applied, too, i.e. four black sibling nodes are merged into their father node. Hence, the quadtree in Figure 8 would be represented by 0320-0001-0311-1111[59] or 0330-0001-0111 after

[56] The dashes in the codes are included for readability only and are not part of the real LQ code.

[57] Of course, this would open up the possibility to encode three additional literals. However, Gargantini does not outline any ideas in this regard and the further adjustment of the LQ code is not the subject of this work.

[58] Implicitly, this also applies to the LQ code in which no literal is spent for the root node. Generally, encoding the root node is unnecessary for trie-based quadtrees: The root has to exist as an internal node of the quadtree structure for quadtree partitionings of more than just one cell and does not carry any further information.

[59] The dashes in the CBLQ codes are included for readability only.

condensation. Since there are four different literals (0 to 3), we require 2 bits to encode a literal.

Usage. Hybrid summaries are always built in a two-step process (also see section 4.3): In the first step, the descriptive foundation of the resource summary is built by using some approach. In a second step, this foundation is refined by applying some other approach. Hence, hybrid approaches combine two 'solitary' approaches with each other. In the context of hybrid approaches, quadtrees can be used both as a foundation as well as a refinement. If a quadtree is used as a foundation, no condensation of the quadtree structure is applied since the area delineated by a black cell is still refined in the second step.

3.3. Excursus: Dealing with Spatial Errors

Before we finally come to the summarization approaches, we briefly consider an aspect which is sometimes neglected when dealing with spatial data: the imperfection of (non-synthesized) spatial data. The spatial data we use in the evaluations in section 7 and section 8 originates from GPS sensors of consumer devices such as smartphones or digital cameras which of course leads to somewhat inaccurate position data. Generally, imperfections are inherent to spatial data—no measurement system is indefinetly accurate [Devillers et al. 2010, p. 387f]. Therefore, the captured locations of the data points possibly are not the exact positions where the associated media items have been created. This may lower the quality of the data itself as well as the quality (i.e. the accuracy) of the resource summaries. In this section, we provide some background information on spatial data quality and how to deal with these inaccuracies in our application scenarios.

Commonly, different user groups require different levels of quality from spatial data. For example, aircraft navigation requires highly accurate data whereas for human navigation, an accuracy of a 'few meters' is good enough. As another example, for overlaying weather forecasts to a map, the map is only giving a general indication of a place [Barnaghi et al. 2017, Best Practice 14]. The assessment of the 'fitness for use' [Devillers et al. 2010, p. 390] of a given data set for specific purposes is only possible if the quality of the data is documented. This is broadly recognized in both the academic as well as the 'practical' domain (see e.g. [Devillers et al. 2010, p. 388] or [Nebert 2004, p. 24]). Consequently, spatial data quality has become a significant focus area in international spatial data standards. Examples are 'ISO 19113' and 'ISO 19114' which emphasize on the measurement of spatial data quality and its documentation. In the academic domain, there are conferences such as the *International Symposium on Spatial Data Quality* where the research community meets [Devillers et al. 2010, p. 389].

In general, metadata is 'data about data' and describes the characteristics of a data set [Nebert 2004, p. 25]. It allows for communicating some of the quality information to the end users of the data. Ideally, metadata struc-

tures and definitions are referenced to a standard [Nebert 2004, p. 28ff]. Standards for (geo)spatial metadata have been generated by a number of organizations. Examples are the 'Content Standard for Digital Geospatial Metadata' by the Federal Geographic Data Committee (FGDC), or the 'ISO 19115' about geographic information metadata by the International Organization for Standardization (ISO). Besides the ISO Technical Committee 211 (ISO/TC211), the Open Geospatial Consortium (OGC) is the main player in the area of standardization and unification of concepts related to (geo)spatial data [Management Association 2016, p. 425]. The OGC is a consortium of over 500 companies, government agencies, and universities.[60] It works closely with the FGDC and the ISO/TC211 to develop formal, global spatial metadata standards [Nebert 2004, p. 30].

Part of these standards and related documents is the identification of the elements of spatial data quality, and how to describe this metadata. An example is the well-known list of five to seven data quality items (see e.g. [Guptill et al. 1995, p. 8ff]) which—depending on the classification scheme—includes lineage, logical consistency, completeness, positional accuracy, attribute accuracy, semantic accuracy, and temporal accuracy [Devillers et al. 2010, p. 393]. As another example, the data quality working group of the OGC is concerned with defining a framework for describing spatial data quality by a number of categories such as accuracy (positional, temporal, thematic), completeness, projection, scale, and so on.[61] In the work of Barnaghi et al. ([Barnaghi et al. 2017]) which defines an 'OGC Best Practice' on spatial metadata and constitutes an official position of the OGC membership on this particular topic (→ OGC Document Number: OGC 15-107), it is specified that at least spatial extent, coverage, and representation should be included in the description of a spatial data set. Further properties include the number of dimensions, the coordinate reference system, or the spatial resolution (which describes the positional accuracy of spatial data). The very general definition of metadata—data about data—includes an almost limitless spectrum of possibilities ranging from human-generated textual descriptions to machine-generated data for describing spatial data. Furthermore, different types of (geo)spatial data (e.g. vector, raster, boundary, etc.) may require different levels and forms of metadata to be collected [Nebert 2004, p. 25f]. In general, spatial data quality is, first of all, about data quality in general [Devillers et al. 2010, p. 397]—the major difference is the emphasis on the spatial component [Nebert 2004, p. 25].

In our given application scenario, describing spatial data quality mainly involves the positional accuracy of the data points for which additional meta-

[60] http://www.opengeospatial.org/ogc, last visit: 02.11.2017.

[61] http://www.opengeospatial.org/projects/groups/dqdwg, last visit: 02.11.2017.

data could be provided.[62] According to 'Best Practice 14' of OGC 15-107 ([Barnaghi et al. 2017]), the accuracy of the spatial data can be described in the metadata as a quantitative measure, as a statement of conformance to a standard or a policy, or an assertion or report of 'fitness' for a particular purpose. For observed (i.e. measured) data sets, it is possible to make specific quantitative statements about positional accuracy based on a) the knowledge of the equipment used to make observations and b) any processing carried out. Quantitative statements about positional accuracy should be reported in ground distances (such as meters or feet) [FGDC 1998, p. 5]. The absolute positional accuracy—which is the closeness of reported coordinate values to values accepted as or being true, as per 'ISO 19159-2'—generally is most important (even though the relative positional accuracy can be important for some uses, too [Barnaghi et al. 2017, Best Practice 14]).
For handling non-perfect positional accuracy within the resource description context, there are two plausible possibilities of adding additional information:[63]

— The resource descriptions could incorporate a quantitative measurement (in meters) of the positional accuracy of the data points.
— The resource descriptions could incorporate the information to which standard or policy the positional accuracy of the data points conforms such that quantitative measurements can be derived.

Based on these (derivable) quantitative specifications, an error band ε can be introduced which considers the positional inaccuracy of the described data points for the distance calculations. The considered distance $cdist$ for the distance between a query point q and a resource description res_{desc} is then calculated as:[64]

$$cdist(q, res_{desc}) = max(dist(q, res_{desc}) - \varepsilon; 0)$$

The ε-value must be converted from the quantitative measurement (e.g. given in meters) to a 'distance in space' value in our two-dimensional Cartesian coordinate system (which ranges from -180 to 180 in the x-dimension and from -90 to 90 in the y-dimension, see section 3.1). In sophisticated systems, the conversion factor of meters to this 'distance in space' might be variable depending on the position on the Earth the data stems from and the particular geodetic datum (e.g. WGS 84) used to locate the data points.

[62] Note that some suggested aspects of spatial metadata are already 'naturally' covered in our application scenarios—such as the spatial extent of the data point set which is depicted by a resource summary—or have already been specified in section 3.1—such as the projection of the data points into the two-dimensional Cartesian coordinate system.

[63] A possibility that does *not* require incooperation of information into summaries would be to agree a priori on an error band within the scope of an application.

[64] Note that the *max*-operator in the following equation is added to avoid possibly negative result values for $cdist$.

Additionally, it has to be considered that quantitative measurements for the positional accuracy are usually based on sampling, involving a comparison with some ground truth data (that itself is inaccurate). This ground truth is very hard to come by in our application scenarios. Furthermore, for example a 90% confidence interval for the given distances also means that 10% of the data points do not fall within the given threshold—the query results in such a system might therefore not be exact.
We do not go further into detail at this point since we do not address positional inaccuracies of the data points in our application scenarios (as also 'non-perfect' results are reasonably acceptable in the given application scenarios). Nevertheless, this section shall raise the awareness of data quality issues in the spatial domain ('all spatial data are wrong, but some are useful' [Devillers et al. 2010, p. 395]) and their backgrounds, and gives an idea of possible solutions for considering this uncertainty with regard to the resource descriptions.

4. SUMMARIZATION APPROACHES FOR SPATIAL DATA POINT SETS

In this section, the summarization approaches developed specifically for the distributed application scenario are presented. Previous works ([Henrich and Blank 2010; Kufer 2012; Kufer et al. 2012; Kufer et al. 2013; Kufer and Henrich 2014; Blank et al. 2016; Kufer and Henrich 2017]) already created and evaluated various of suchlike summarization approaches. The approaches described in this thesis are comprised of the most promising of these pre-evaluated approaches, several further developments, as well as a basic approach which serves as a benchmark for comparison.

In general, a resource description is a brief, contentual[65] representation of a resource in a multi-resource environment. Obviously, it is the solution to the task associated with the *resource description problem*. If all resources of a multi-database system have to be queried for their relevant database items in a sequential scan, this is expensive (both computationally as well as in terms of network load) and inefficient as presumably, only a small fraction of the resources contribute to the query result. In particular, this applies when kNN queries are assumed for which k is only a small fraction of the amount of the database items which are administered in the network.

A number of considerations with regard to the design of resource descriptions have already been conducted in section 2, particularly from section 2.1 to section 2.4. Concretely, the following requirements are demanded from resource descriptions as we assume a distributed search system supporting kNN queries:

— For each resource, the position(s), an area, or several areas in which the resource's data points are located must be described. The descriptions shall enable an accurate distinction between relevant resources—which contribute to the final query result and *must* be contacted—and irrelevant resources—which do not contribute and *shall* not be contacted. The quality of how well the different resource description approaches achieve this distinction is referred to as *effectivity* or *selectivity* of an approach in the evaluation.

— Simultaneously, the storage space consumption of the resource descriptions has to be kept as low as possible. A summary consisting of indexed areas is an inadequate description of a resource if it consumes more storage space than the coordinates of the data points which it shall delineate concisely. The accomplishment of low storage space consumption is considered as the *efficiency* of an approach in the evaluation.

[65] In our concrete distributed application scenario, we assume the 'content' of a resource to be the spatial properties of the geotagged media items. Consequently, a resource description depicts the spatial footprint of a resource—either by a direct representation or by a summary representation. Remember that the term 'resource description approach' encompasses both the summarization approach as well as the direct representation.

— It is desirable to keep the computational costs for determining the query result at a low level. Thus, simple shapes are to be preferred for indexing areas by resource summaries.[66]

For a classification of the different *summarization approaches* presented in this section, we differ between approaches *partitioning the data*—utilizing a (set of) bounding volume(s)—and approaches *partitioning the data space*—decomposing the data space into a set of subspaces. For the latter, an important further subcategorization is into *global* and *local* space partitioning approaches (see section 4.2). The *hybrid* approaches constitute the third main class. They are characterized by combining the properties of two arbitrary 'solitary' approaches. Generally, the descriptive power of one approach builds the foundation of a hybrid resource summary which then is refined by the descriptive power of a second approach. The combined approaches together form a new, hybrid approach.
In the following, the different summarization approaches which are evaluated for the distributed application scenario are outlined. Hereby, they are structured accordingly to the three main classes introduced—*data partitioning approaches* (section 4.1), *space partitioning approaches* (section 4.2), and *hybrid approaches* (section 4.3). Note that generally, we assume that the concrete parameter values of the parameterizable summarization approaches are known in the network. Consequently, this information does not have to be transmitted.

4.1. Data Partitioning Approaches

When designing summarization approaches which utilize bounding volumes to delineate the spread of a resource's 'point cloud', two basic properties have to be decided on a priori:

— The shape of the bounding volume (also see section 2.2).
— The number of bounding volumes used.

As described in section 2.2, the shape property is generally a trade-off between simplicity of the shape and tightness of the fit. Axis-aligned rectangles are most common in the context of indexing spatial data. They are easy to compute, have a small storage space footprint, and facilitate fast intersection and distance tests. The tightness of the approximation can vary depending on the data point set to represent. Nevertheless, these rectangles are usually more flexible and adequate compared to spheres which offer a slightly smaller storage space footprint and similar computational complexity. More complex shapes like convex hulls can be a better fit but

[66] Yet, note that the focus of the evaluation is on the *effectivity* and the *efficiency* of the resource description approaches. Therefore, the computational costs are not explicitly assessed in most of the evaluation. Depending on the application scenario (e.g. when searching a small-scale P2P network), the query processing time is also rather dominated by aspects such as network latency and not by the CPU time required for calculations.

often come at the expense of a significantly bigger storage space footprint and significantly higher computational costs.
An important characteristic of spatial data sets is that the data is often distributed in an irregular manner [Oosterom 1999, p. 385]. As discussed in section 2.4, using a single, yet arbitrarily complex form to describe the data is often not adequate due to the inevitably enclosed dead space. In contrast, using multiple but simpler forms can be a much more appropriate means. However, this requires a partitioning of the data point set into adequate groups.
In the following, we present the MBR approach which uses a single bounding volume of simple shape—a Minimum Bounding Rectangle. It is a very basic approach which serves as a benchmark for the other approaches. Although it does not partition the set of data points into groups, it still uses a bounding volume enclosing the data points and thus is a very 'data-point-centric' summarization approach—which is the other important property of the approaches falling into data partitioning class. The second approach presented, RecMAR$_{k,sl}$, partitions the set of data points into up to k groups and utilizes an axis-aligned rectangle as a bounding volume for each group. In [Kufer 2012] and [Kufer et al. 2012], it has been shown that RecMAR$_{k,sl}$ is superior to e.g. utilizing a single convex hull or other approaches which also utilize multiple, simple bounding volumes (both spheres and axis-aligned rectangles) but for which (kmeans++) clustering is used to partition the data points into groups.

MBR. The MBR approach[67] is a parameter-free approach as it uses only a single Minimum Bounding Rectangle (MBR) to summarize the spatial footprint of a resource.
The MBR (also known as bounding box or envelope [Caldwell 2005]) is an axis-aligned rectangle, i.e. its sides are parallel to the coordinate axes of the space [Samet 2005, p. 195]. It is the smallest rectangle bounding a set of spatial features (the resource's data points in our case) and can be specified by two coordinates: the lower left corner and the upper right corner [Caldwell 2005]. See Figure 9 for a visualization of the MBR approach for our example resource. Note that the International Date Line is not considered as the boundary of the data space in the x-dimension when determining the MBR. Hence, the MBR might cross the International Date Line.[68]

[67] The MBR approach is the standard approach to summarize spatial data whose improvement is part of thesis objective ❶ (see section 1.4).

[68] The indexing of areas crossing the International Date Line can also occur for other data partitioning approaches or hybrid approaches utilizing bounding volumes as a foundation. This is because the computation of bounding volumes—in contrast to most of the data partitioning approaches—does not require bounded data spaces. Since we strive for the most accurate summaries possible, for these approaches, the International Date Line is not considered as a boundary in the x-dimension when computing the resource summaries (for distance calculations, the International Date Line is *never* considered as a boundary in the x-dimension). This also applies to the polygons of the Voronoi cells of the Voronoi-based approaches, UFS$_{n,cc}$ and DFS$^{b}_{n,cc}$ (see section 4.2 and section 4.3).

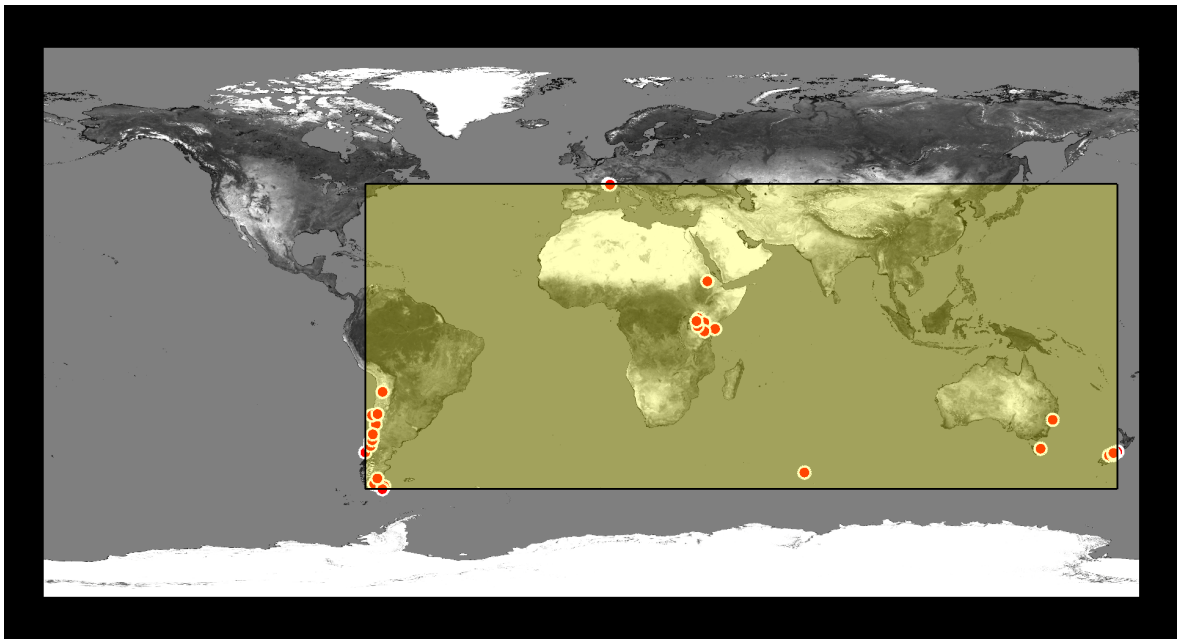


Fig. 9: Visualization of the MBR approach for the example resource which administers 2,186 data points. The white bordered red dots denote the locations of the data points of the resource, the yellow surface indicates the data-point-containing area which is indexed by the resource summary.[69]

For encoding the resource summary, the coordinates of the MBR's lower left corner and upper right corner are captured in single-precision floating-point format which occupies 32 bits for each value. The four necessary values for encoding the two-dimensional rectangle are sequentially stored in a bit vector (see Figure 26 on page 96 for an overview of the storage formats for the approaches which do not utilize quadtrees).

RecMAR$_{k,sl}$. As mentioned several times before, the use of a single, arbitrarily complex form can be inadequate since it potentially results in that a lot of dead space is enclosed in the bounding volume, or massive storage space requirements, or both. These problems can be addressed by utilizing multiple but simple bounding volumes which fit tighter around single data point accumulations and significantly reduce the dead space enclosed while consuming predictable amounts of storage space. This requires an arrangement of the data points into separate groups which are then approximated by a bounding volume, each.
The Recursive Minimum Area Rectangles (RecMAR$_{k,sl}$) approach[70] is a parameterizable approach which divides the data points into up to k groups

[69]The image of the world map in the background is an edited image of the 'Visible Earth' catalog of the NASA. Specifically, the original image is the 'Blue Marble: Land Surface, Shallow Water, and Shaded Topography' image of the catalog. Source: https://visibleearth.nasa.gov/view.php?id=57752, last visit: 28.11.2018.

[70]The RecMAR$_{k,sl}$ approach has already been developed until the year 2012. Thus, it is one of the approaches whose improvement is part of thesis objective ❶.

while applying parameter sl for a stopping criterion. RecMAR$_{k,sl}$ is based on an algorithm developed by Becker et al. ([Becker et al. 1991]) for approximating a set of axis-aligned rectangles by two axis-aligned MARs. As mentioned in the 'R-tree-Based Approaches'-paragraph[71] of section 2.4.3, it yields the same result as Guttman's exhaustive split algorithm but runs in polynomial time. Among all pairs (s, t) of enclosing rectangles inside the overall MBR m, it finds the pair for which the sum of the surface areas of s and t is minimal. The algorithm separately looks for three different types of solutions and then selects the best one. The types of solutions are distinguished based on which sides of s touch the sides of m and on the overlap between s and t:

1. s has at least two adjacent sides on the boundary of m, and s and t have no common interior points.
2. s has at least two adjacent sides on the boundary of m, and s and t have some common interior point.
3. s has no two adjacent sides on the boundary of m.

We adjusted the algorithm for the use with point data and introduced a fourth type of solution in case all the points of m are on an axis-parallel line (and hence the surface areas of s and t are 0). Here, the solution for which the sum of the perimeters (or rather the sum of the line lengths) of s and t is minimal is selected.
The algorithm can be applied recursively to compute up to k MARs since it decomposes an arbitrary MBR m into two MARs s and t which contain all of m's data points. Each of m's data points is assigned to either s or t. In the set $M = \{m_0, \ldots, m_n; n < k\}$ of already computed rectangles, the rectangle m_i whose surface area or sum of line lengths (if the rectangles all have a surface area of 0) is largest is taken from M. m_i is then decomposed into the two MARs $m_{i.s}$ and $m_{i.t}$ which afterwards replace m_i in M. For the recursion, a stopping criterion has to be defined. The recursion stops if either the maximum of k rectangles has been reached, or the distance between the center of a rectangle and each of its associated data points is smaller than a threshold value sl for all rectangles.[72] See Figure 10 for an example visualization of the RecMAR$_{k,sl}$ approach.
The set of the l rectangles which have been computed at the end of the recursion process form the resource's summary ($1 \leq l \leq k, 4 \cdot 32$ bits for each rectangle). The rectangles are sequentially encoded into a bit vector, each described by its lower left corner and its upper right corner.

4.2. Space Partitioning Approaches

The principle behind this class of approaches is to partition the data space into a set of disjoint subspaces which are then exploited to index the area(s)

[71] This paragraph begins on page 45.
[72] This might result in that a resource is described by only a single rectangle.

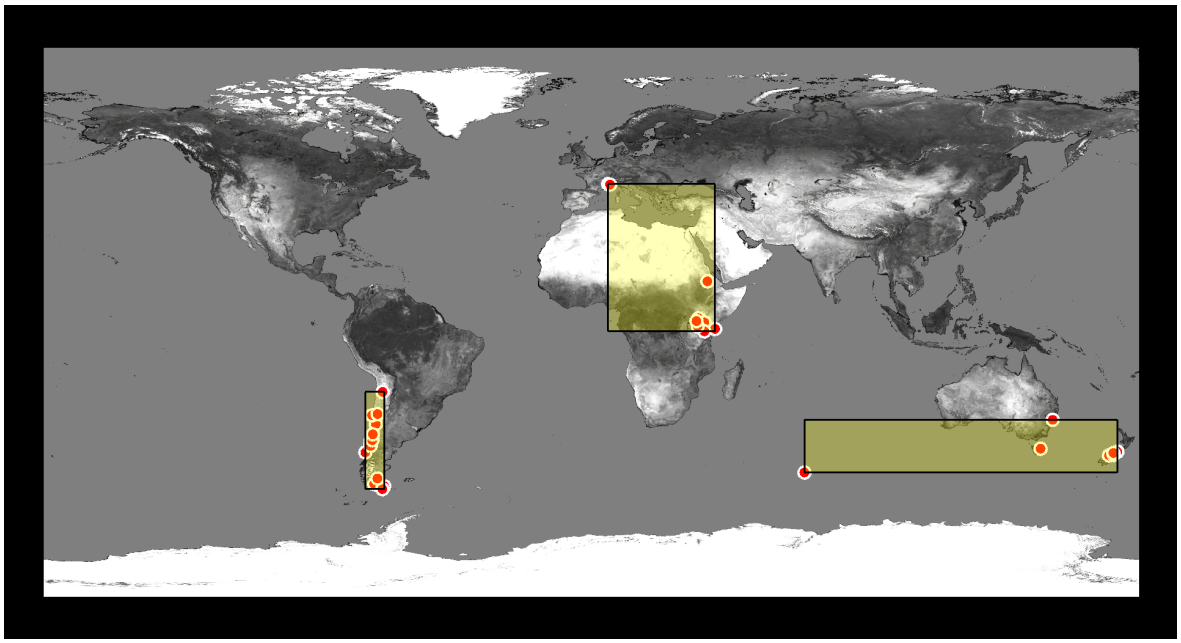


Fig. 10: Visualization of the example resource summarized by $\text{RecMAR}_{3,0.1}$.

containing a resource's data points as a means of its summarization. Most of these space partitioning approaches are only applicable if the data space is bounded, i.e. the International Date Line must be treated as a closed boundary in the x-dimension when decomposing the space (but *not* for calculating distances, as described in section 3.1).
For a subcategorization of the space partitioning approaches, different properties can be assessed:

— Is the space decomposition hierarchical or non-hierarchical?
— What is the shape of the resulting subspaces?
— Is the space partitioning *global* or *local*?

Hierarchical decomposition is based on the principle of recursive decomposition of the universe `U` (similar to divide and conquer methods) [Samet 2005, p. XIX]. This allows to focus on the 'interesting' (i.e. data-point-containing) regions of `U`. The hierarchical decomposition of `U` is represented by a corresponding tree structure which can be binary (such as for the k-d-tree family) or multiway (such as for the quadtree family). Non-hierarchical decomposition partitions the universe without the utilization of a congruent tree structure. Grid-based methods for example impose a d-dimensional orthogonal grid on the universe. Another means of non-hierarchical decomposition is the use of sites, resulting in a Voronoi diagram.
The resulting subspaces of a space partition are often rectangular-shaped, and can be of either equal or arbitrary size. In contrast, the subspaces of a Voronoi diagram are arbitrarily shaped convex polygons. In several parts of section 2.4, the shapes of space partitions have already been discussed.
The most important distinction for this work is between *global* and *local* space partitioning. This distinction does not relate to structural proper-

ties of the space partition itself (or its construction process) but refers to the *individuality* of the space partition for the single resources. It is about whether the space partition is the same for all resources (global) or if it can also be resource-individual (local). This is dependent on whether the information necessary to (re)construct the subspaces of a space partition can be represented with very low storage space requirements—such that it can be incorporated into a resource's summary without violating the *efficiency* requirement. If so, resource-individual, local space partitioning is applicable. Generally, it is only feasible for a regular space partition. For example, the linear encoding of (trie-based) quadtrees (see section 3.2) enables local space partitioning. In contrast, for Voronoi diagrams, a resource-individual space decomposition is not applicable since the Voronoi cells are determined by the sites whose exact coordinates must be known. Apart from the unresolved question of where these resource-invidual sites could come from, this would consume too much storage space. The explicit storage of the Voronoi cells' polygons would consume even more storage space. Thus, for space partitioning approaches for which the subspaces' region information cannot be efficiently represented in a resource's summary, global space partitioning has to be applied. The information on the global space partition has to be distributed separately in the resource network. A resource summary then only contains the data which is captured for each subspace but not the region information of the global subspaces. For local space partitioning approaches, all of the information is captured in the resource summaries themselves.

The last pending issue is what information shall be captured for each subspace of a space partition in the resource summaries. Regardless of the concrete decomposition method, the resulting subspaces are generally distinguishable into areas containing data points (*occupied cells*) and areas not containing data points (*non-occupied cells*). This distinction is the base for the expressiveness of a corresponding resource summary. Two types of information are reasonably associable with each subspace in a resource summary:

— Binary information: Does the subspace contain at least one data point (occupied $\rightarrow$ 1) or not (non-occupied $\rightarrow$ 0)?
— Quantitative information: How many data points does the subspace contain?

In our scenario, assuming an equal-sized storage space footprint, the use of binary information leads to significantly more effective summaries since occupancy information on much more subspaces can be captured which allows for decomposing U into many more, finer regions.[73] A tight delineation

[73]Note that theoretically, integrating quantitative information into the resource summary is also possible for data partitioning approaches. Nevertheless, similar to the space partitioning approaches, it is preferable to utilize the storage space for a tighter delineation of the area(s) in which a resource's data points are located.

of the areas containing data points is far more valuable for the distinction between relevant and irrelevant resources than quantitative information for loose areas (assuming 4 B per quantity value) [Henrich and Blank 2010, p. 25].

UFS$_{n,cc}$. The Ultra Fine-grained Summaries (UFS$_{n,cc}$) approach[74] is a global space partitioning approach using a Voronoi diagram to partition the underlying space.
As just discussed, a Voronoi space decomposition is not suited for local space partitioning since either the coordinates of the sites or the corresponding polygons of the Voronoi cells are required to determine the subspaces' regions. This information cannot be encoded efficiently in a resource's summary given hundreds or thousands of sites, especially since the cell occupancy information also has to be encoded somehow (which would be more of a problem for a site-storing local approach).
Hence, the UFS$_{n,cc}$ approach utilizes a global Voronoi diagram of a given set of n sites $S = \{s_1, \ldots, s_n\}$ which is the same for all resources. See Figure 11 for an example visualization. As a global space partitioning approach, the distinctive quality of the space decomposition is highly dependent on the selection of appropriate sites. The selection of sites is discussed in section 6.4.3. The second parameter cc (for centroids considered) is only relevant for a 'metric-domain-like' resource selection and has no effect on the resource summaries.[75]
Binary information about the cell occupancy is captured in the resource summary. It is represented by a bit vector, sequentially containing the occupancy information for all n subspaces. The sequence of the sites (and

[74] The UFS$_{n,cc}$ approach has already been developed until the year 2012. Thus, it is one of the approaches whose improvement is part of thesis objective ❶.

[75] Due to of the definition of a Voronoi diagram (see section 2.4.2), a data point's Voronoi cell can be determined by calculating the distances between the data point and all of the sites. Therefore, Voronoi-like space partitioning is also applicable in metric spaces where only distances between objects are known. This has already been discussed in section 2.6.3. In the metric domain, the Voronoi-like space partitioning is also called 'multiway generalized hyperplane partitioning' [Blank 2015, p. 25]). As a consequence, in the context of the resource selection process, this allows for the comparison of resource ranking schemes also suited for metric spaces and resource ranking schemes only suited for vector spaces in order to verify the presumption stated in section 2.6.3 that making use of the coordinate information of the spatial domain will yield the best results for spatial data. See section 5.1 for the respective ranking algorithms. The 'metric-domain-like' ranking is based on the consideration of the distances between the query point and the sites, only. Yet, not all of the sites have to be considered for the ranking. The parameter cc controls how many sites are taken into account and is only used for the 'metric-domain-like ranker' in the resource selection process. For the resource summaries themselves as well as for the 'spatial domain ranker', the parameter cc has no effect. Note that in our evaluations, the pruning of resources for Voronoi-based approaches is *always* based on the explicitly reconstructed polygons of the Voronoi cells—thus, the 'metric approach' only concerns the ranking of the resources.

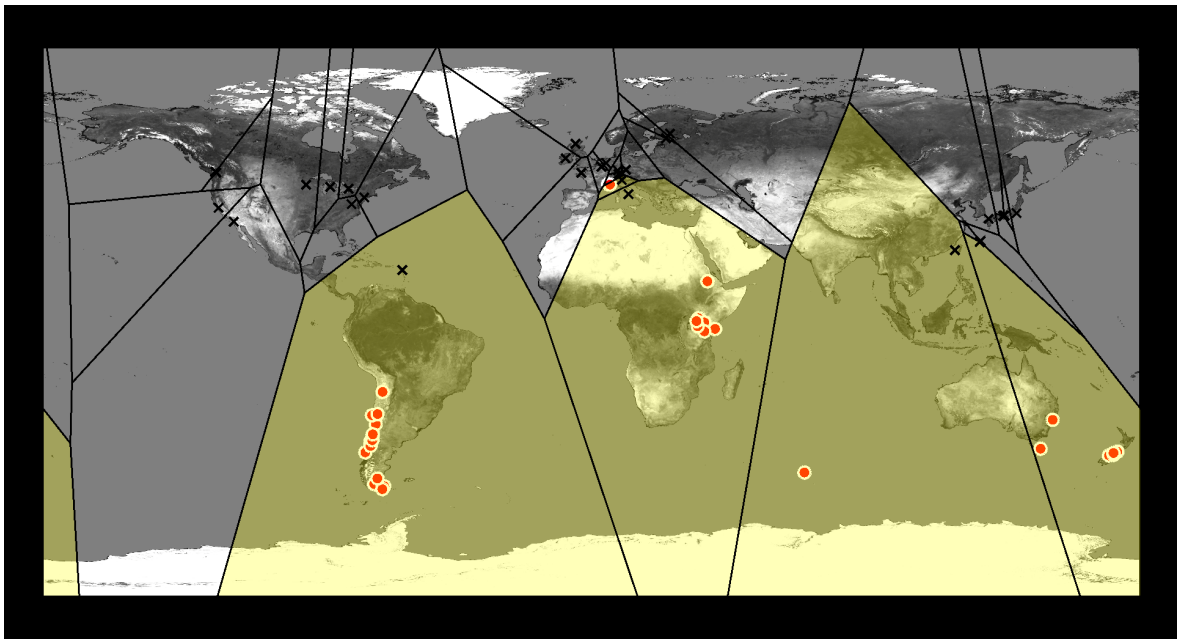


Fig. 11: Visualization of the example resource summarized by $\text{UFS}_{32,cc}$ (the cc parameter has no effect on the resource summary itself). Subspaces which contain at least one data point (i.e. the indexed areas) are highlighted, the black crosses denote the Voronoi sites.

therefore the cell IDs of the corresponding subspaces) and their coordinates must be distributed separately in the network.

KD_n. The KD_n approach[76] employs a k-d-tree-like space partition to decompose the data space.
With regard to the resource summaries, it has to be decided whether the k-d space partition is to be global or local. If the space partition is a regular decomposition corresponding to that of a trie-based k-d-tree variant (i.e. cycling through the dimensions and splitting into halves), the corresponding tree structure provides implicit knowledge of the positions of the partitioning hyperplanes which can be exploited for linear encodings similar to linear quadtrees. Although it has not been researched as extensively as linear quadtrees, research on for example the *linear kd-tree* [Poirrier 2009] or the *linear bintree*[77] [Samet and Tamminen 1988] exists. Nevertheless, for this work, we refrain from local k-d-tree space partitioning for the following reasons:

— In the linear kd-tree, the records identify elements instead of nodes like in linear quadtrees [Poirrier 2009, p. 7]. Therefore, the linear kd-tree is not directly comparable.
— The linear bintree is tailored for higher-dimensional data [Samet and Tamminen 1988, p. 579], whereas $d = 2$ applies for our data.

[76] The KD_n approach has already been developed until the year 2012. Thus, it is one of the approaches whose improvement is part of thesis objective ❶.

[77] Remember that the bintree is a trie-based k-d-tree variant for region data (see section 2.4.3).

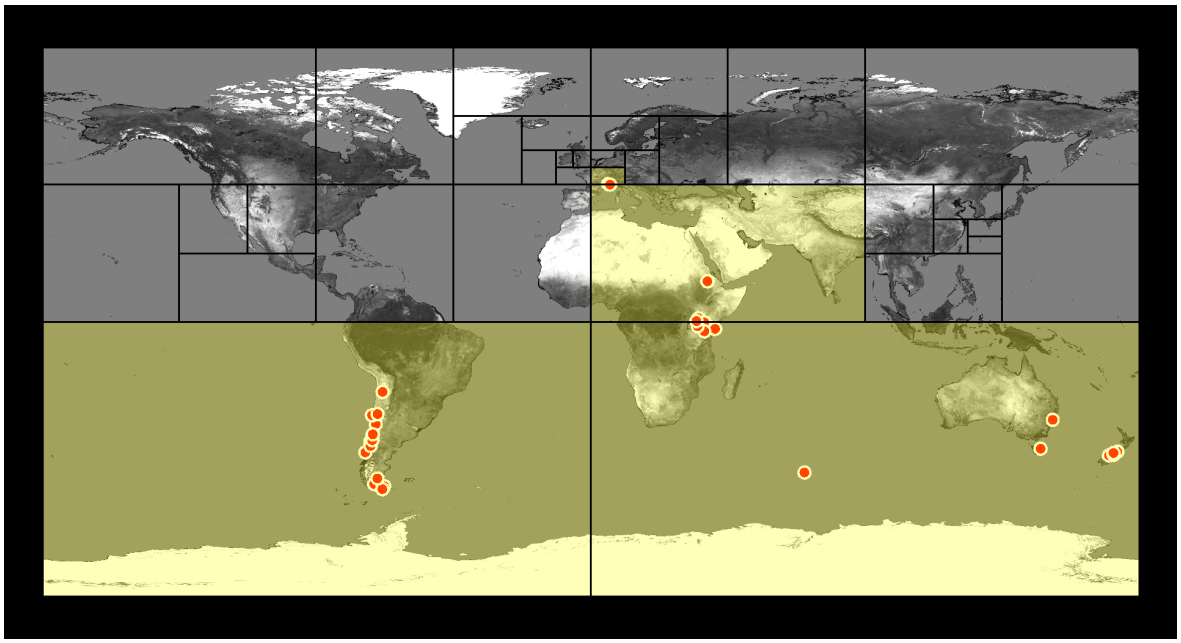


Fig. 12: Visualization of the example resource summarized by KD_{32}.

— A global k-d-tree space partitioning approach is an interesting object of comparison with regard to global Voronoi-based approaches.

Thus, for the KD_n approach, a *global* space partition is applied which is learned from training data. For this, a bucket overflow approach is used. The training data points are successively inserted into buckets of a certain capacity. Each bucket simultaneously represents a subset of the universe (i.e. a k-d cell). Initially, the universe consists of only one bucket. When a bucket overflow occurs (i.e. its capacity is exceeded), the overflowing bucket's k-d cell is split into two and the data points are re-assigned to their appropriate new bucket based on in whose k-d cell they are located in. In the corresponding tree structure representing the split hierarchy, the split dimension is altered cyclically and the respective dimension's interval is always split into halves. After a split, the training data insertion procedure is continued. The whole process carries on until an amount of n buckets has been reached. See Figure 12 for an example visualization of the KD_n approach. Similar to the selection of sites for $UFS_{n,cc}$, the quality of the space decomposition is dependent on the selection of appropriate training data. This is discussed in section 6.4.3.
A KD_n resource summary corresponds to a $UFS_{n,cc}$ resource summary, i.e. it is a bit vector sequentially encoding binary occupancy information for all n subspaces. The information on the global k-d space partition and the sequence of the single subspaces in the summaries have to be distributed separately in the network, too.

$QT_{c,a}$. The $QT_{c,a}$ approach conducts local space partitioning by using quadtrees. This means that the data space is partitioned solely based on the locations of the *individual resource*'s data points.

The quadtree for describing a resource's indexed areas is built according to the general rules discussed in section 3.2, i.e. dependent on the parameters c (specifying the maximum number of quadtree cells) and a (specifying the lower bound threshold surface area after which the quadtree cells are no further partitioned). Condensation is applied. There is no loss of information in merging four black sibling leaf nodes into their father node since it results in the same overall indexed area. See Figure 13 for an example visualization of $\text{QT}_{c,a}$.
For the resource summaries, the quadtree structure is captured in a linear quadtree encoding which is then converted into a bit vector. As for all quadtree-utilizing approaches, the quadtree part of a resource's summary can be encoded in either of the available encoding schemes (LQ code or CBLQ code).[78] The LQ code requires additional metadata besides the LQ-encoded quadtree structure to enable an unambiguous decoding:

— The number of levels (#lvl) or depth of the quadtree's tree structure must be captured. In the LQ code, there is no marker to tag the end of the encoding of a black node at the deepest level of the quadtree. The theoretical maximum level of a quadtree with c cells is $\lfloor c/3 \rfloor = l$. Therefore, we require $\lceil log_2\, l \rceil$ bits to encode the depth of a quadtree. This information is located at the head of the LQ-encoded summary's bit vector.
— Directly behind, the number of occupied quadtree regions (#oqr) is captured, requiring $\lceil log_2\, c \rceil$ bits. At the tail of the bit vector, the actual LQ-encoded linear quadtree data (df-order, only black nodes) is captured.[79]

For the CBLQ code, only the actual CBLQ-encoded linear quadtree data (bf-order, all leaf nodes and internal nodes) is captured (see Figure 27 on page 97 for an overview of the bit vectors' structure for the approaches utilizing quadtrees).
As a local space partitioning approach, all the information required to reconstruct a resource's indexed areas is already contained in the resource summary. Therefore, no additional data needs to be distributed separately in the resource network for $\text{QT}_{c,a}$—as opposed to for $\text{UFS}_{n,cc}$ and KD_n.

[78] In the evaluation, if there are several encoding options for representing the same information, a resource is represented by the option requiring the least storage space (also see section 6.4.1).

[79] This information is not necessarily required to decode $\text{QT}_{c,a}$ resource summaries but for the hybrid approaches utilizing a quadtree. See the 'Grid$\text{QT}_r^{c,a}$'-paragraph in section 4.3 for further details (the paragraph begins on page 88). Nevertheless, we also capture this information for the $\text{QT}_{c,a}$ approach since this allows for a unified encoding and decoding procedure—which is not trivial to implement—while requiring only $\lceil log_2 c \rceil$ additional bits. For the most selective $\text{QT}_{c,a}$ parameterization tested in the evaluation, these are $\lceil log_2\, 8192 \rceil = 13$ bits. This is non-significant considering the average resource description sizes resulting in the evaluation (see section 7; also see Figure 32 on page 118 for the evaluated parameter ranges). It is especially true since the resource descriptions may also be compressed in addition (see section 6.4.1). Nevertheless, this means that for $\text{QT}_{c,a}$, there is still some *slight* optimization potential for reducing the sizes of the LQ-encoded resource summaries.

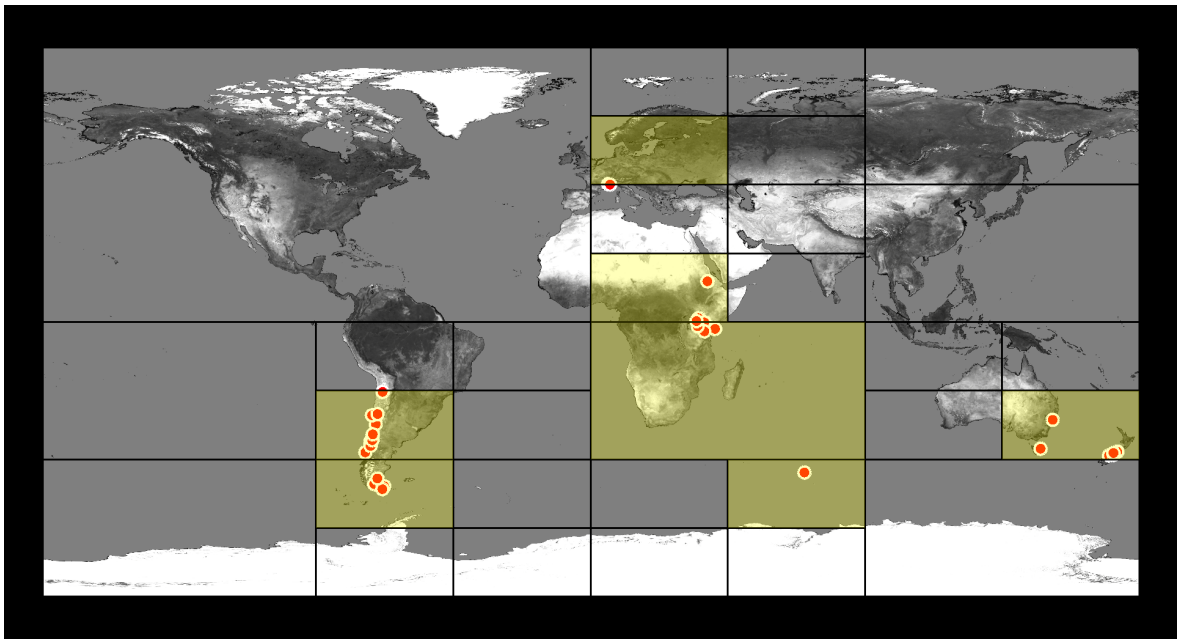


Fig. 13: Visualization of the example resource summarized by $\mathrm{QT}_{32,1.0}$.

4.3. Hybrid Approaches

Hybrid approaches combine properties of two different approaches to form a new approach—which might result in more effective and/or more efficient resource summaries compared to 'solitary' approaches. They might then even be appropriate for use under conditions which were previously unsuitable for one of the involved approaches.

For example, for space partitioning approaches, the union of all subspaces spans the entire data space [Seeger and Kriegel 1990, p. 590]. Consequently, they do not only partition 'live space' but invariably also empty data space (dead space). Furthermore, they are not suited for unbounded data spaces—in contrast to approaches using bounding volumes. On the other hand, approaches describing multiple bounding volumes suffer from difficulties related to dividing the data point set of a resource into adequate groups—such as high computational costs or possibly poor groupings. Space partitioning approaches can bypass these problems with their decomposition of the data space: They group data points 'indirectly' on the basis of cell membership instead of dividing them 'directly' into groups. In addition, they are often able to describe a set of areas much more efficiently in terms of storage space.

Generally, a combination of two approaches—one of each class—can mitigate or even overcome the respective drawbacks. For example, an adaptive space partitioning of the live space of an unbounded data space can become possible. In our application scenario, the data space is bounded, and space partitioning approaches are easily applicable. Therefore, we are rather interested in building more effective and/or more efficient resource summaries than exploring new areas of application. Hereby, the two approaches which are combined do not necessarily have to be from different

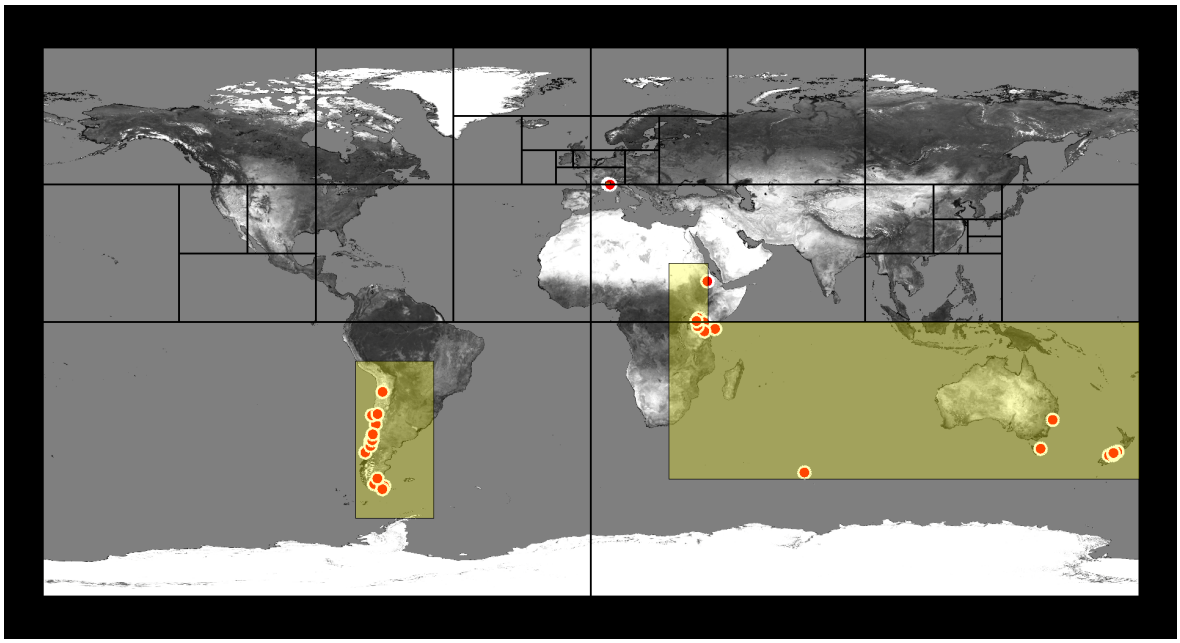


Fig. 14: Visualization of the example resource summarized by KDMBR_{32}^{3}.

classes. For example, the combination of two space partitioning approaches is also feasible.
Independent from which two specific approaches are combined to a new hybrid approach, a two-step process is always applied to create hybrid summaries:

— In the first step, the descriptive power of approach A is used to build the foundation of the resource summary.
— In the second step, the descriptive power of approach B is added as a refinement to the foundation to further increase the expressiveness of the hybrid summary.

Both data partitioning approaches as well as space partitioning approaches can be employed for both foundation as well as refinement. In the following, the hybrid approaches which are evaluated for the distributed application scenario are presented.

KDMBR$_n^b$. The KDMBR_n^b approach[80] is a hybrid combining a k-d space partition as a foundation with a bounding volume MBR for the refinement of an occupied cell.
The k-d space partition of n cells is built exactly as for the KD_n approach. In the refinement step, for each occupied cell of the k-d space partition, the data points located in this cell are additionally bounded by an MBR. These cell-interior MBRs are quantized, exploiting the available region information of the occupied cell for minimizing the additional storage space required. See Figure 14 for an example visualization of KDMBR_n^b.

[80] For an explanation of the arrangement of the parameters in a hybrid approach's name, see the '$\texttt{Approach}_{a,b}^{x,y}$'-entry in the glossary.

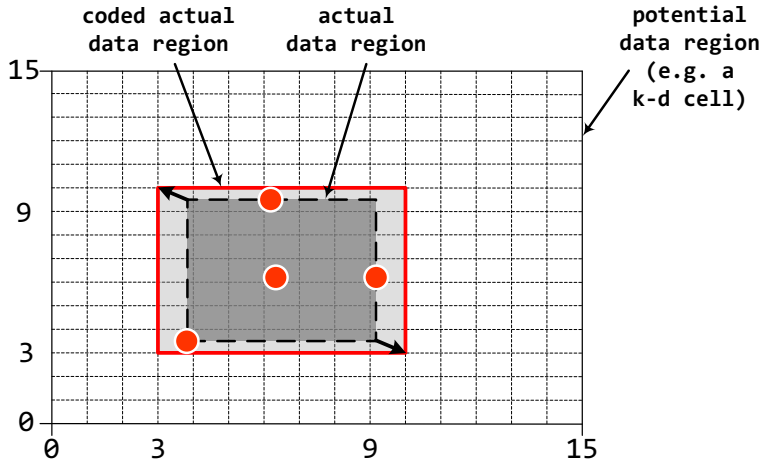


Fig. 15: Exemplary depiction of the quantized approximation of a cell-interior MBR. Parameter b is set to 4, i.e. 16 positions can be distinguished per dimension ($\rightarrow 15 \times 15 = 255$ cells for the quantization grid). The actual MBR (*actual data region*, dotted black lines) is slightly extended (*coded actual data region*, closed red lines) to fit the underlying quantization grid invoked onto the *potential data region*.

The utilization of quantized MBRs embedded into cells of a space partition has already been mentioned as a concept applied for various multidimensional data structures (like the hybrid tree or the LSD^h-tree) in section 2.4.3. For KDMBR$_n^b$, it works as follows (also see Figure 15): The rectangular region of an occupied cell of the k-d space partition is the *potential data region*. The MBR containing all of the cell's data points is the *actual data region*. It is conservatively approximated by a multidimensional interval which is quantized into a grid of $(2^b - 1)^d$ cells which is invoked onto the *potential data region*, exploiting the *potential data region*'s presence. The *coded actual data region* (i.e. the approximated MBR) is slightly bigger than the *actual data region* but requires less storage space to be depicted in the resource summary—both being effects of the quantization. The accuracy of the quantization is dependent on parameter b which specifies the amount of bits spent per bound in each dimension (for the dimensionality, $d = 2$ applies in our case).
In the resource summary, binary information about the cell occupancy of the k-d space partition is captured sequentially in a bit vector. If a cell is occupied, the $4 \cdot b$ bits for encoding the corresponding quantized MBR follow immediately (also see Figure 26 on page 96). Similar to KD$_n$, the global k-d space partition and the subspace sequence must be distributed separately in the network.

KDMAR$_n^{b,k}$. As was last mentioned in conjunction with the RecMAR$_{k,sl}$ approach, the use of a single *full-precision* bounding volume might be insufficient to delineate a set of data points. This issue can also apply to a cell-interior, *quantized* MBR: Possibly, large parts of the cell are covered by the quantized MBR, i.e. the refinement is of little use. It has also been

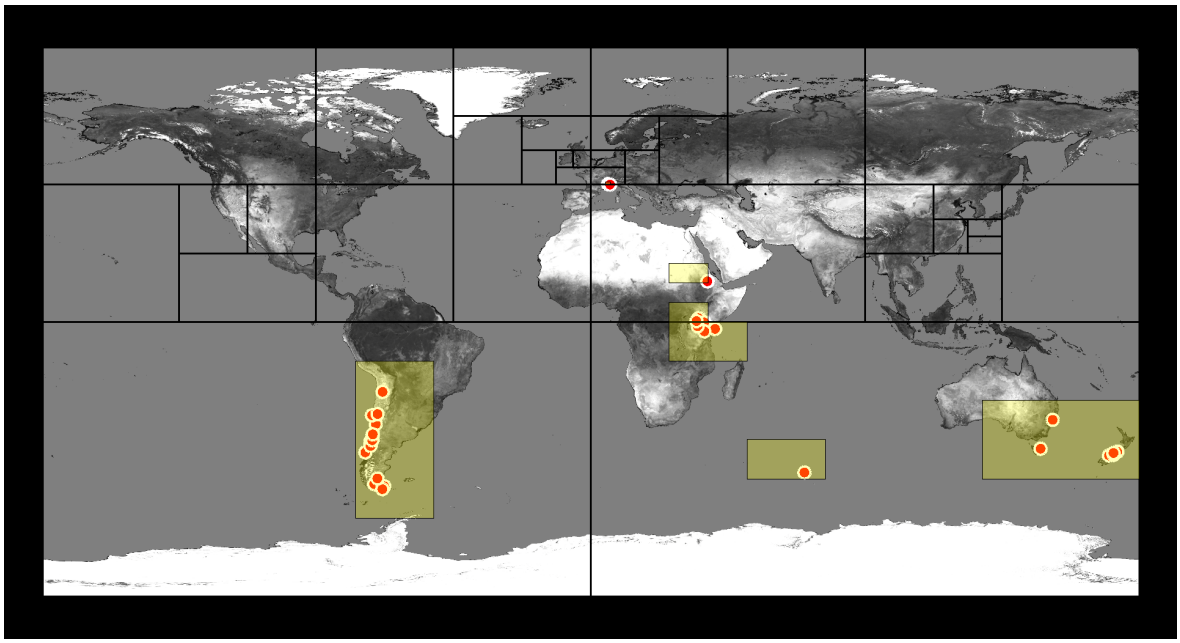


Fig. 16: Visualization of the example resource summarized by $\text{KDMAR}_{32}^{3,3}$.

reported by Dang et al. ([Dang et al. 2001, p. 341]) that generally, there are some cases in which there is no benefit of using quantized MBRs regardless of how many bits b are used since there can be a high remaining dead space ratio (due to the spread of the data in the potential data region). The $\text{KDMAR}_n^{b,k}$ approach is an evolution of the KDMBR_n^b approach which is oriented towards this issue. Hence, in an occupied cell of the k-d space partition, a *set* of quantized MARs is used for refinement instead of just a *single* quantized MBR.

A k-d space partition of n cells is the foundation (built exactly as for KDMBR_n^b). For the data points of an occupied cell, up to k MARs are computed. See Figure 16 for an example visualization of $\text{KDMAR}_n^{b,k}$. For the MAR calculation, basically the same greedy, recursive algorithm as for $\text{RecMAR}_{k,sl}$ is used. Solely the stopping criterion is adjusted: The algorithm *only* stops prematurely if each calculated MAR is a rectangle with a surface area of 0 and an extent of 0 in all dimensions (i.e. if each MAR is already a point; for lines, the algorithm is continued). Afterwards, the *coded actual data regions* are built, i.e. each MAR is extended to fit the underlying quantization grid.

Due to this adjustment to the underlying grid, the MARs of a cell often overlap, or are adjacent and aligned. Consequently, their unions often form rectilinear polygons (which might contain holes). Thus, the number of rectangles indexing the data-point-containing areas of the cell *might* be reduced by *decomposing* the rectilinear polygons into the minimum number of non-overlapping rectangles. Suitable algorithms for this issue have been discussed in section 2.3.[81]

[81] No attempt to find the optimal *cover* is made since the polygon cover problem is NP-hard for polygons with holes (see section 2.3.2).

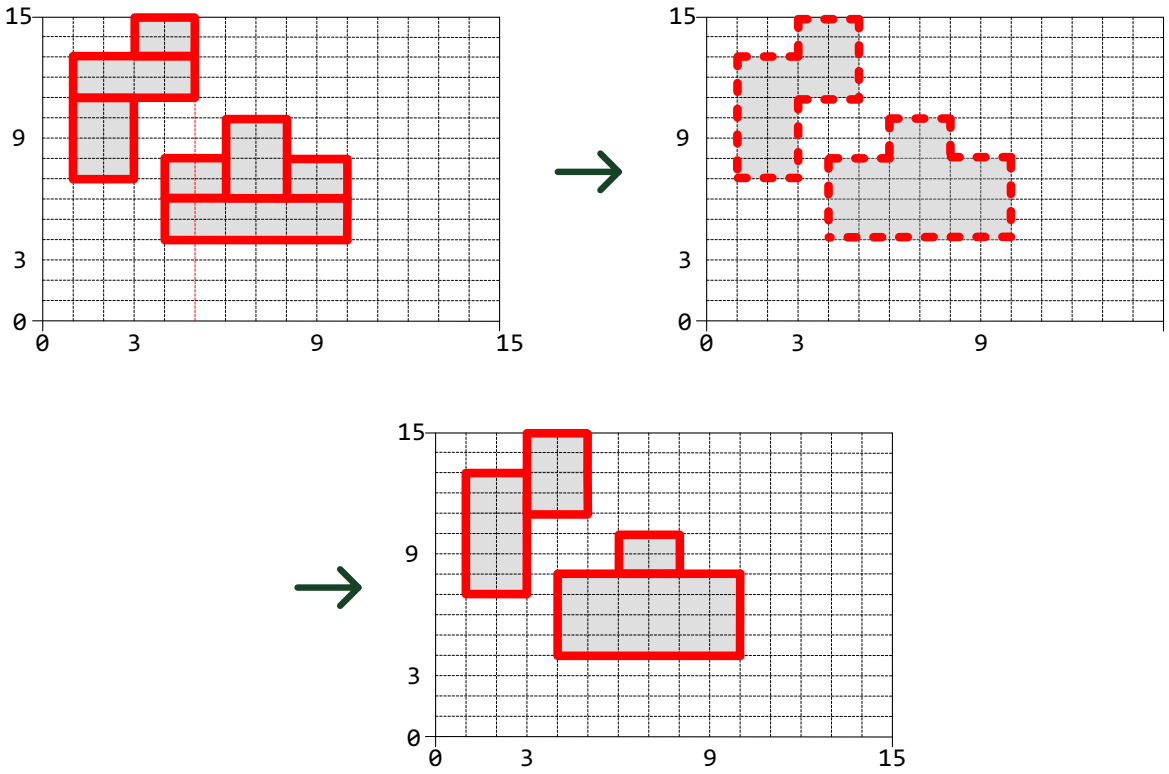


Fig. 17: Exemplary depiction of a rectangle number reduction procedure for an occupied cell. Overlapping as well as adjacent and aligned input rectangles (top, left) are condensed into rectilinear polygons (top, right). Finally, these polygons are decomposed into the minimum number of non-overlapping rectangles (bottom).

Consequently, for each occupied cell, a rectangle number reduction procedure is conducted (see Figure 17 for an example). It works as follows: First, the entirety of coherently overlapping or adjacent and aligned MARs are condensed into rectilinear polygons. Afterwards, each polygon is decomposed into its minimum number of non-overlapping rectangles with the algorithm of Imai and Asano [Imai and Asano 1986, p. 491]. If the overall number of rectangles is reduced, the result of the rectangle number reduction procedure is selected to represent the data located in the occupied cell. Otherwise, the original representation is kept.[82]
Binary information about the cell occupancy of the k-d space partition is captured sequentially in a bit vector. Likewise KDMBR$_n^b$, the refinement information immediately follows the bit for the corresponding occupied cell. Since the number of quantized refinement rectangles may vary, $\lceil log_2 k \rceil$ bits encode the number x of rectangles in the cell ($1 \leq x \leq k, k \geq 2$). Afterwards, the data of the corresponding quantized rectangles—$4 \cdot b$ bits

[82]Note that the input of the rectangle number reduction procedure—the overlapping or adjacent and aligned MARs forming the rectilinear polygons—already might define an optimal or close to optimal solution with regard to the polygon *cover* problem—and therefore might require less rectangles than the solution of the polygon *decomposition* problem.

each—follows. Similar to KD_n and KDMBR_n^b, the information on the global k-d space partition and the subspace sequence must be distributed separately in the network.

$\text{DFS}_{n,cc}^b$. As has already been implied before, the $\text{UFS}_{n,cc}$ approach can be seen as an implementation of the *multiway generalized hyperplane partitioning* scheme known from metric space partitioning (see section 2.6.3) due to the utilization of sites to partition the data space. Since the locations of the sites are known, an obvious extension of the $\text{UFS}_{n,cc}$ approach is to incorporate the covering radius information r^{out}—which in addition to a site is used to partition the data space with the *ball partitioning* scheme in metric spaces (see section 2.6.3)—into a resource summary.[83] This extension is the $\text{DFS}_{n,cc}^b$ (Distance-enhanced UFS) approach.
Consequently, for each occupied cell of the Voronoi diagram of n sites, the covering radius r^{out} is captured.[84] See Figure 18 for an example visualization of $\text{DFS}_{n,cc}^b$.
The r^{out} distances are quantized with b bits to reduce storage space requirements. Thus, a base value to quantize against has to be set. For the summary of each resource, we employ the *resource-individual* maximum covering radius r_{max}^{out} as base value for the quantization (also see Figure 19). The r_{max}^{out} value can vary significantly between the various individual resources. Hence, we consider a resource-individual value to be better suited for the quantization base value than both a pre-fixed global value or the global maximum covering radius (i.e. the maximum covering radius occuring for any resource of the network). This is at the expense of requiring additional storage space for a resource's description as r_{max}^{out} must be captured as a 32-bit single precision floating-point number. On the other hand, it results in more accurate resource summaries due to the individually determined quantization base value. Compared to employing a global maximum covering radius, it additionally eliminates the need to distribute the global maximum covering radius separately in the network as well as the threat of being forced to recalculate all resource summaries in case the global maximum covering radius changes.
In the bit vector of the resource summary, the first 32 bits encode the resource's *individual* maximum covering radius r_{max}^{out}. Afterwards, binary information on the cell occupancy of the Voronoi space partition is captured sequentially. For each bit of an occupied cell, b bits encoding the quantized r^{out} covering radius of this cell follow immediately (also see Figure 26 on

[83] Remember that r^{out} is the radius of the minimum ball centered at the cell's site which covers all the cell's data points. Alternatively, it can be defined as the maximum distance between a site s_i and any of its associated data points.

[84] In [Kufer 2012], the utilization of the inner radius r^{in} as well as of combinations of r^{out} and r^{in} has also been examined. It turned out that the exclusive capturing of the r^{out} values was most promising since the r^{out} information is much more valuable for pruning than the r^{in} information. Note though that in [Kufer 2012], the different covering radii have not been quantized but were captured with full precision.

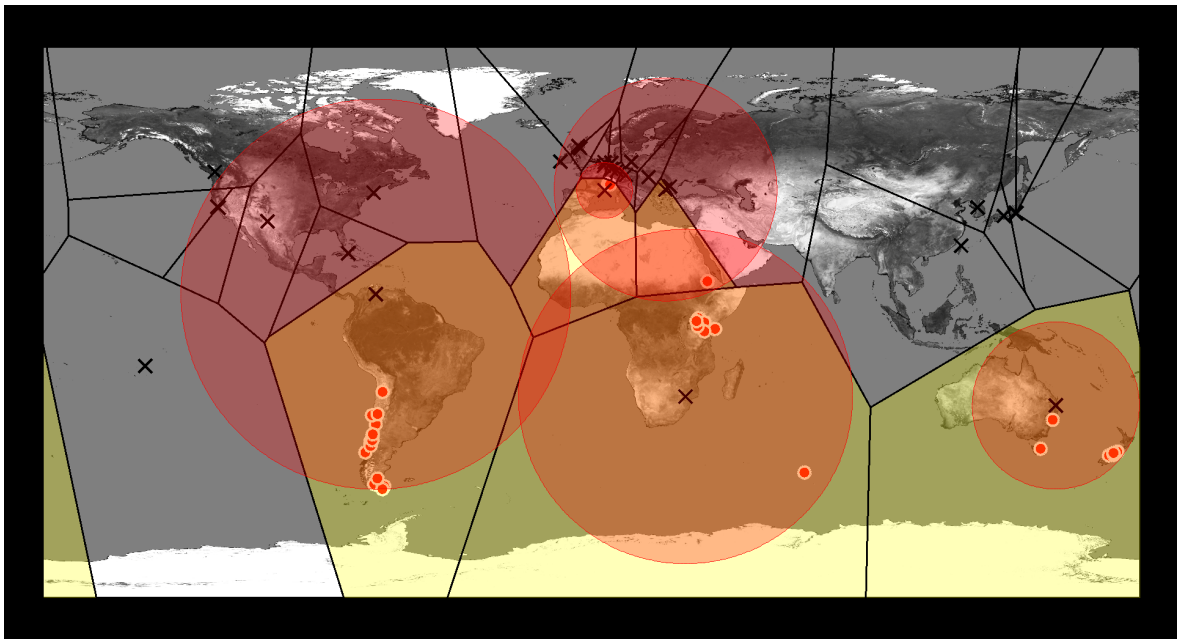


Fig. 18: Visualization of the example resource summarized by $\text{DFS}^{3}_{32,cc}$ (the cc parameter is not relevant for the resource summary itself). Occupied subspaces are highlighted yellow, the black crosses denote the Voronoi sites. The red circles centered at the Voronoi sites depict the 'covering balls' of the occupied cells, created on the basis of the quantized covering radii information (r^{out}) captured in the resource summaries.

page 96). Similar to $\text{UFS}_{n,cc}$, the coordinates and the sequence of the sites must be distributed separately in the network.

GridQT$^{c,a}_{r}$. The GridQT$^{c,a}_{r}$ approach is a hybrid of a (fixed) grid[85] as foundation for which occupied grid cells are refined with a cell-interior quadtree. Fixed grids are generally used as global space partitioning approaches. For GridQT$^{c,a}_{r}$, a uniform fixed grid is the foundation of the resource summary.[86] The uniform grid is imposed onto the universe $\mathtt{U}$ with r rows and

[85] With 'fixed grid', we refer to grids that cover the entire data space.

[86] As discussed in the 'Hashing-Based Approaches'-paragraph of section 2.4.3, there are two ways of building a fixed grid: subdividing the data space into grid cells of equal size (uniform grid), or placing the dividing hyperplanes at arbitrary positions (non-uniform grid) which requires linear scales to capture the positions of the hyperplanes in each dimension.

The uniform grid as a pure global space partitioning approach (named Grid$_r$) has been evaluated in previous works ([Henrich and Blank 2010], [Kufer 2012]). It did result in storage-space-efficient but not very selective resource summaries since the uniform grid does not adapt to erratic data. Thus, its cells are not fine-grained enough for data space regions with a high global data point density. A non-uniform grid approach performed worse than Grid$_r$ in [Kufer 2012]. The non-uniform grid was built in congruence with the one of the *grid file* (see section 2.4.3), i.e. every time a bucket is split, the splitting hyperplane not only spans the interval of the bucket to split but the entire range of the split dimension. Consequently, the split position is utilized as a value of the split dimension's linear scale. As a consequence, the hyperplanes induced by regional splits also trigger the partitioning of grid cells in entirely uninvolved regions which led to subpar results for the non-uniform grid approach.

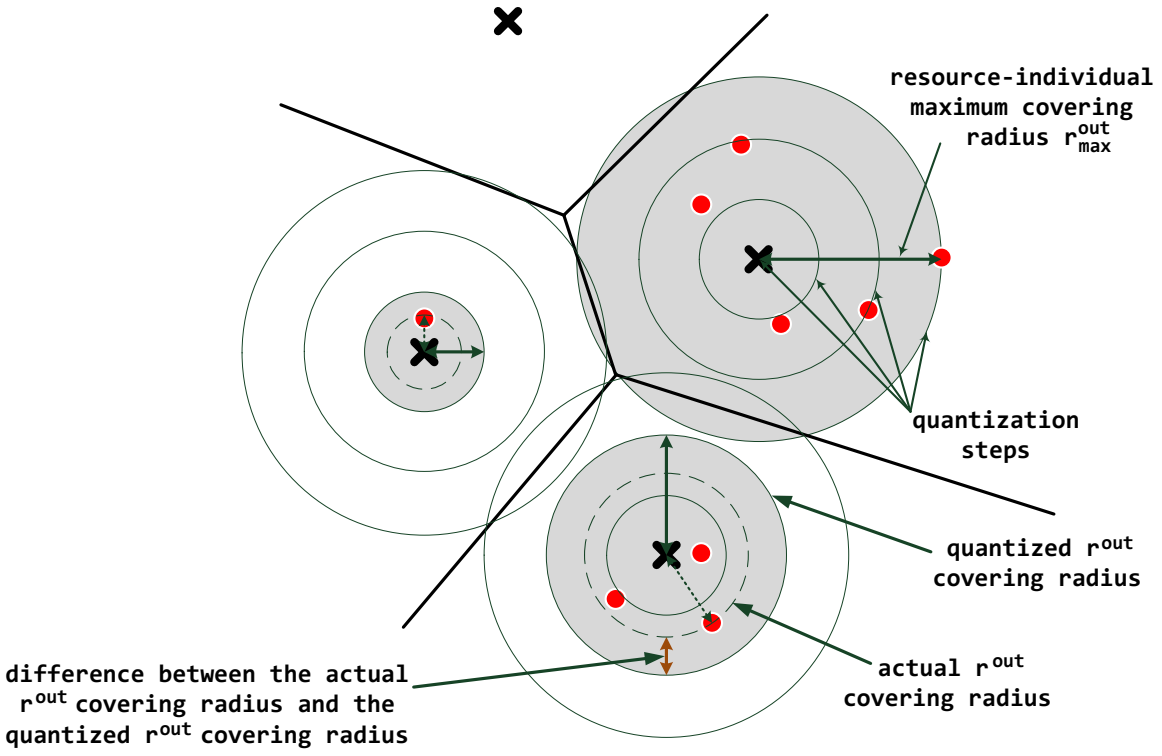


Fig. 19: Exemplary depiction of the covering radii determination for $\text{DFS}^{b}_{n,cc}$. Assume the covering radius of the top right Voronoi cell to be the resource's maximum covering radius r^{out}_{max}. In the summaries, the quantized r^{out} radius is stored for each occupied cell. In the bottom Voronoi cell, the spatial inaccuracy which is introduced by the quantization is explicitly depicted.

$2{\cdot}r$ columns (since the x-dimension has twice the range of the y-dimension). For each occupied grid cell, a cell-interior quadtree is built which is dependent on the parameters c and a. See Figure 20 for an example visualization of $\text{GridQT}^{c,a}_{r}$. For each quadtree, condensation is applied to reduce the storage space requirements.
In the bit vector of the resource summary, binary information about the occupancy of the $2r^2$ cells is captured sequentially. For each occupied cell, the quadtree information follows immediately. Both the LQ code and the CBLQ

[87] When condensation is applied, the maximum number of occupied quadtree regions is $c-1$. This is because if all quadtree regions were occupied, the quadtree would be condensed into its root node. Therefore, only $\lceil log_2\,(c-1)\rceil$ bits are required to encode the actual number unambiguously. Nevertheless, we utilize $\lceil log_2\,c\rceil$ bits to encode the actual number of occupied quadtree regions since it allows for a unified implementation of the non-trivial encoding and decoding procedures with approaches where no condensation is applied (i.e. hybrid approaches that use quadtrees as a foundation). The resulting disparities are dismissable, particularly in proportion to the resulting average resource description sizes (see section 7). Nevertheless, mind that for $\text{GridQT}^{c,a}_{r}$ and all the other approaches using quadtrees as a refinement, there is still *minimal* optimization potential for reducing the LQ-encoded resource summary sizes.

[88] Consequently, the encoding structure of an LQ-encoded refinement quadtree of $\text{GridQT}^{c,a}_{r}$ is equivalent to an LQ-encoded quadtree of $\text{QT}_{c,a}$.

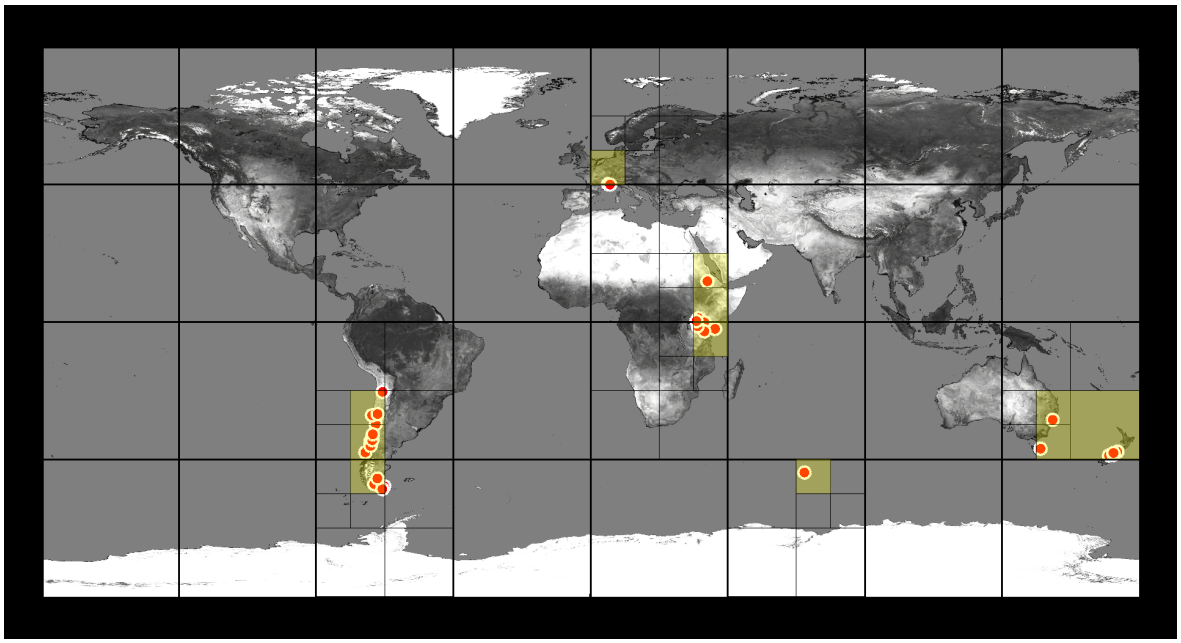


Fig. 20: Visualization of the example resource summarized by GridQT$_4^{8,1.0}$.

code require additional metadata besides the linearly encoded quadtree data to unambiguously distinguish the quadtree data from the continuation of the grid cell occupancy information:

— For the LQ code, the level (#lvl) and the number of occupied quadtree regions (#oqr) must be captured. The #lvl value is encoded with $\lceil log_2\ l \rceil$ bits at the head of the quadtree data block. Directly behind, the #oqr value is encoded with $\lceil log_2\ c \rceil$ bits.[87] The LQ-encoded quadtree data is at the tail of the quadtree's data block.[88]
— For the CBLQ code, the number of internal nodes (#in) is captured. #in is $\lfloor c/3 \rfloor = i$ at a maximum. Therefore, $\lceil log_2\ i \rceil$ bits are required for encoding the actual number of internal nodes. Directly behind, the CBLQ-encoded quadtree data is attached.

Due to its regularity, no additional information on the global space partition must be distributed separately in the network for GridQT$_r^{c,a}$. The space partition can simply be reconstructed from parameter r (in contrast to for example KDMBR$_n^b$). The subspace sequence starts with the cell at the bottom left (cell $ID = 0$) of the data space and goes to the cell at the top right (cell $ID = 2r^2 - 1$), row by row.

KDQT$_n^{c,a}$. The KDQT$_n^{c,a}$ approach is a hybrid which decomposes the embedding space with a global k-d space partition as foundation (just as KDMBR$_n^b$ and KDMAR$_n^{b,k}$). For the refinement, it utilizes a cell-interior quadtree for each occupied cell. Analogous to GridQT$_r^{c,a}$, the cell-interior quadtrees are built dependent on the parameters c and a, and the quadtrees are condensed. See Figure 21 for an example visualization of KDQT$_n^{c,a}$. In case an occupied cell of the k-d space partition is smaller than the threshold

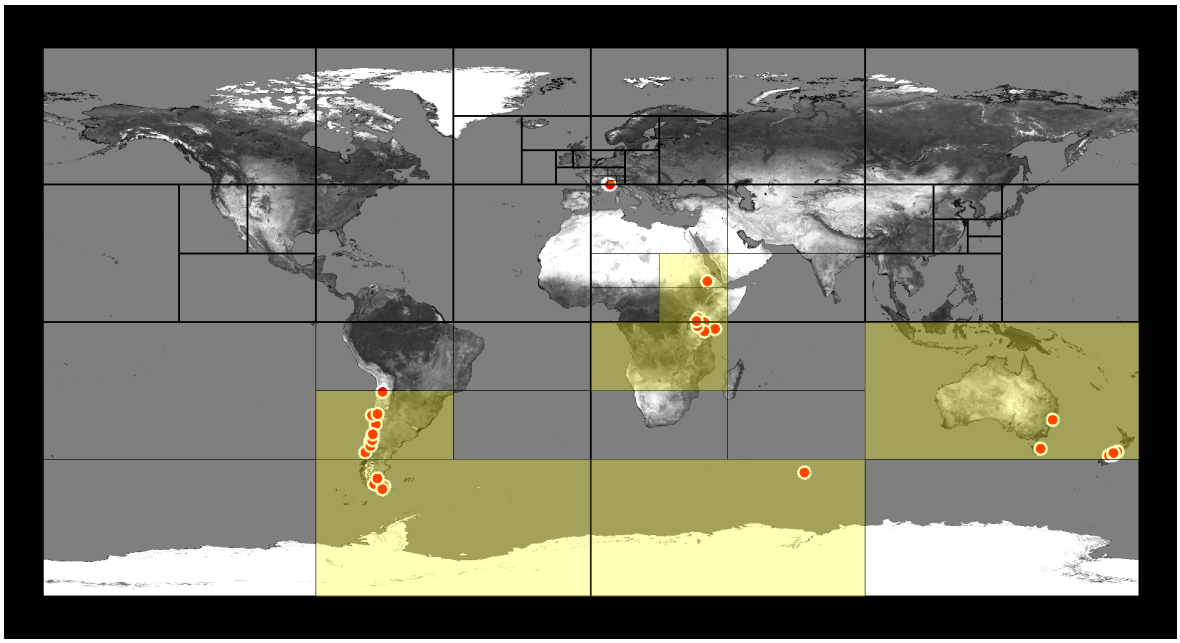


Fig. 21: Visualization of the example resource summarized by $\text{KDQT}_{32}^{8,1.0}$.

surface area a, the root of the corresponding cell-interior quadtree is split once, and only once.[89]
The $\text{KDQT}_n^{c,a}$ resource summaries are structurally identical to the $\text{GridQT}_r^{c,a}$ resource summaries. Due to its non-regularity, information on the global k-d space partition as well as on the subspace sequence must be distributed separately in the network (just as for e.g. KD_n).

$\text{QTMBR}_{c,a}^b$. The $\text{QTMBR}_{c,a}^b$ approach is a hybrid which decomposes the data space with a quadtree space partition that is individual for each resource (and hence a local space partition) as a foundation. In the second step, the occupied quadtree regions are refined with a quantized MBR.
The basic quadtree is built similar to $\text{QT}_{c,a}$, i.e. dependent on the parameters c and a. Nevertheless, no condensation is applied since each occupied quadtree region is still to be refined with a quantized MBR. The quantized MBR is built analogous to KDMBR_n^b, i.e. the embedding subspace's presence is exploited while parameter b determines the accuracy and the storage space requirements. See Figure 22 for an example visualization of $\text{QTMBR}_{c,a}^b$.
In general, the bit vector of the resource summary is designed in such way that the quadtree data is encoded first. Afterwards, the resource's set of quantized MBRs is encoded in a block ($4 \cdot b$ bits for each MBR). The MBRs of this block are in a sequence which corresponds to a z-ordering of the occupied quadtree regions. This way, also the assignment of an MBR to its quadtree region is resolved. Both the LQ code and the CBLQ code re-

[89] Note that for $\text{GridQT}_r^{c,a}$, this cannot happen since the parameters are set accordingly in the evaluation. However, the solution applied for $\text{KDQT}_n^{c,a}$ is also the solution that would be applied for $\text{GridQT}_r^{c,a}$ in such cases.

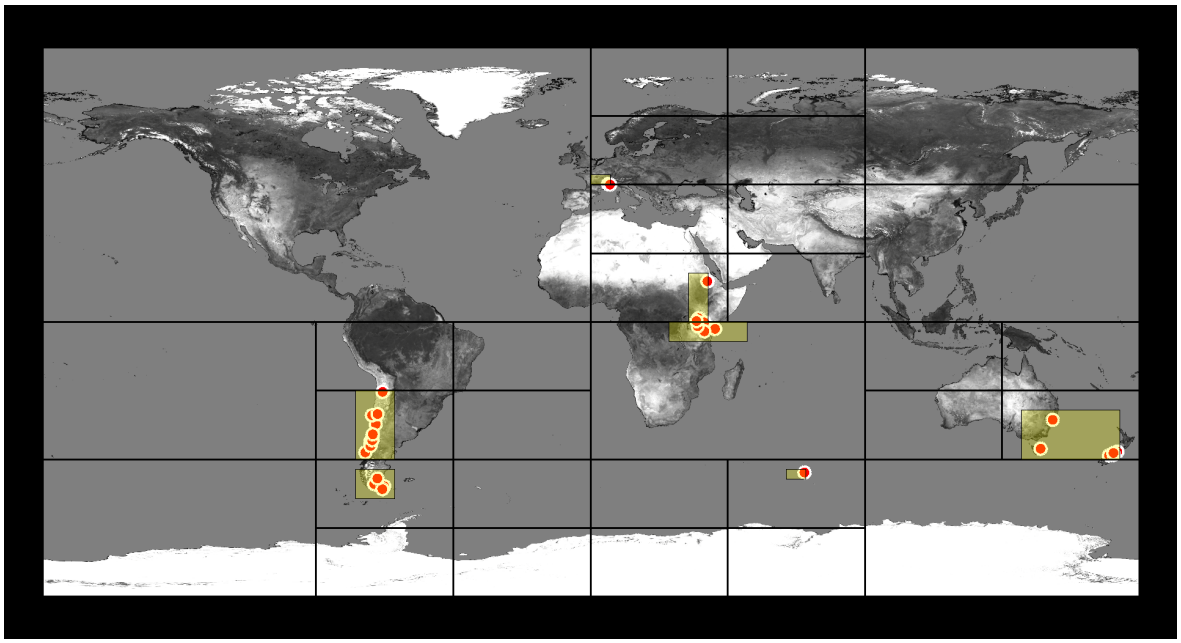


Fig. 22: Visualization of the example resource summarized by $\text{QTMBR}^{3}_{32,1.0}$.

quire incorporating additional metadata to unambiguously distinguish the quadtree data and the data block of the quantized MBRs.

— For the LQ code, the number of levels (#lvl) of the quadtree is captured at the head of bit vector. The maximum number of levels of a non-condensed quadtree with c regions is $\lfloor c/3 \rfloor = l$. Therefore, the #lvl value is encoded with $\lceil log_2\, l \rceil$ bits. Directly behind, the number of occupied quadtree regions (#oqr) is captured using $\lceil log_2\, c \rceil$ bits. The LQ-encoded quadtree data is attached to the metadata. The tail of the bit vector contains the data block of the quantized MBRs.
— For the CBLQ code, the encoding of the number of internal nodes (#in, $\lfloor c/3 \rfloor = i$ at a maximum) of the quadtree occupies the first $\lceil log_2\, i \rceil$ bins of the bit vector.[90] Directly behind, the CBLQ-encoded quadtree data is attached. At the tail of the bit vector, the data block of the quantized MBRs is captured.

As an approach based on local space partitioning, there is no need to distribute additional information separately in the network.

QTMAR$^{b,k}_{c,a}$. The $\text{QTMAR}^{b,k}_{c,a}$ approach is a hybrid which decomposes the space with a local quadtree space partition as a foundation. In the second step, the occupied quadtree cells are refined with up to k quantized

[90] Strictly speaking, the coding of #in is not absolutely necessary since during the bitwise reconstruction of the CBLQ code from the bit vector, it could be checked whether the code describes a complete quadtree. Nevertheless, this would result in additional computational efforts in comparison since by encoding #in, it is defined how many bits belong to the quadtree structure—which can then be simply read out. As #in requires only very few bits, this is a very reasonable trade-off.

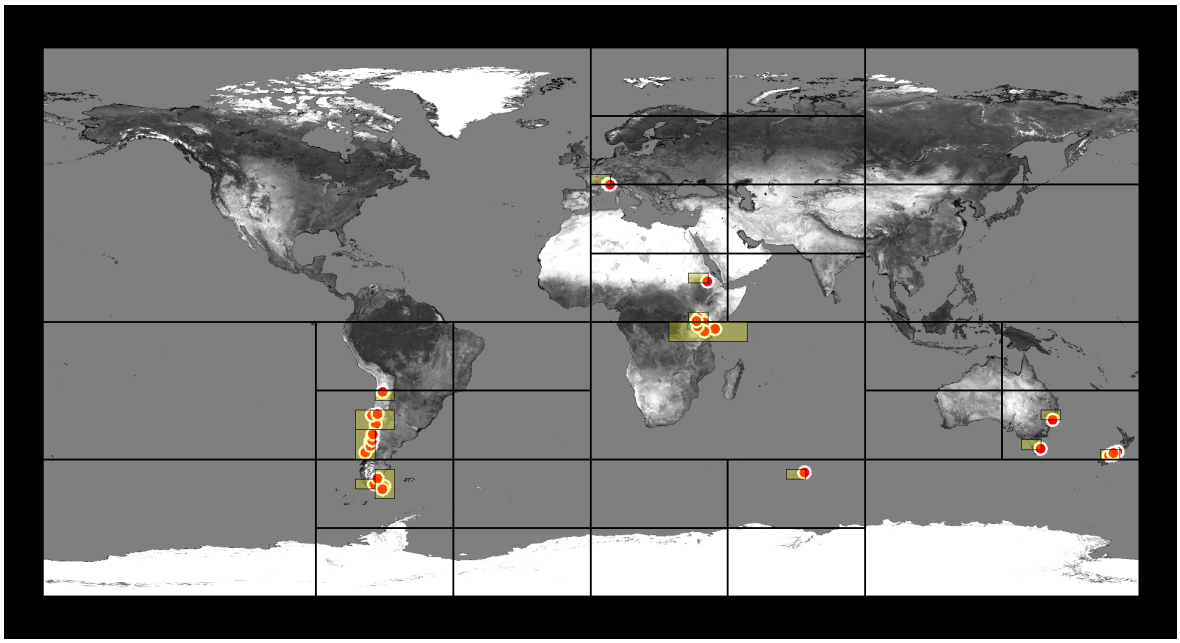


Fig. 23: Visualization of the example resource summarized by $\text{QTMAR}^{3,3}_{32,1.0}$.

MARs. In principle, it is the same evolution to $\text{QTMBR}^{b}_{c,a}$ as $\text{KDMAR}^{b,k}_{n}$ is to KDMBR^{b}_{n}.

The basic quadtree is built dependent on the parameters c and a. No condensation is applied. The accuracy and the storage space requirements for the refinement are controlled by the parameters k (maximum number of quantized rectangles per occupied quadtree cell) and b (number of bits per bound in each dimension). Similar to $\text{KDMAR}^{b,k}_{n}$, a rectangle number reduction procedure is conducted for each occupied cell. See Figure 23 for an example visualization of $\text{QTMAR}^{b,k}_{c,a}$.

In principle, the bit vector of a $\text{QTMAR}^{b,k}_{c,a}$ resource summary is designed just as for $\text{QTMBR}^{b}_{c,a}$: The block of quadtree data is encoded at the head of the bit vector while the block of refinement information follows at the tail of the bit vector. Furthermore, for both schemes, the block of quadtree data is encoded exactly the same as for $\text{QTMBR}^{b}_{c,a}$. The block of refinement information contains the data for the occupied quadtree regions in sequential order. Again, the sequence corresponds to a z-ordering of the corresponding occupied quadtree regions. The only difference is that the number of quantized rectangles can vary for each occupied region. Hence, the information captured for an occupied region consists of two parts. In the first part, the number x of rectangles ($1 \leq x \leq k, k \geq 2$) is encoded by $\lceil log_2\, k \rceil$ bits. In the second part, the data of the corresponding rectangles—$4 \cdot b$ bits each—follows. As an approach based on local space partitioning, there is no need to distribute additional information separately in the network.

MBRQT^{c,a}. The hybrid approaches presented so far all utilize space partitioning approaches as a foundation. Generally, these approaches first need to describe the *entire* universe $\mathtt{U}$ by decomposing it into subspaces—including all its 'uninteresting' regions not containing any data points. As

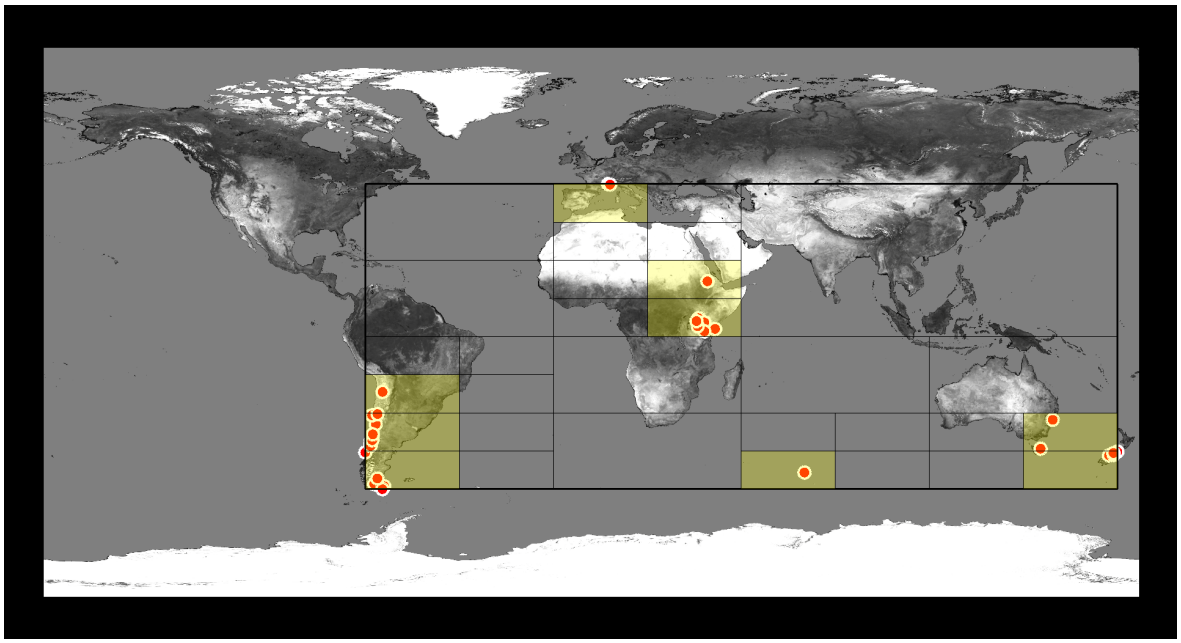


Fig. 24: Visualization of the example resource summarized by MBRQT$^{32,1.0}$.

an alternative, data partitioning approaches can be used as foundation. These envelop the 'interesting' regions of U with one or several bounding volumes, excluding (potentially huge) 'uninteresting' parts of U from any description. The MBRQT^{c,a} approach is a hybrid of this sort. It combines a bounding volume MBR as a foundation with an MBR-interior quadtree as a refinement.

First, the 'region of interest' is determined by computing the MBR of the resource's data points. The MBR-interior quadtree is then built dependent on the parameters c and a—but being bounded only by the MBR and not by U. Note that therefore, single quadtree regions might cross the International Date Line which is not the case for universe-related approaches such as QT$_{c,a}$ or hybrid approaches using a quadtree as a foundation. The MBR-interior quadtree is condensed. In case the exterior MBR is a point or a very small rectangle with a surface area $\leq a$, no MBR-interior quadtree is built and the resource's data points are solely described by the MBR. See Figure 24 for an example visualization of MBRQT^{c,a}.

In the bit vector of the resource summaries, first, the MBR bounds are captured in single precision floating-point format (requiring $4 \cdot 32$ bits). In case the MBR surface area is greater than a, the block of the quadtree data follows. For both schemes, this block is similarly encoded as for e.g. GridQT$_r^{c,a}$.

MARQT$_{k,sl}^{c,a}$. The MARQT$_{k,sl}^{c,a}$ approach is a hybrid which utilizes a set of up to k bounding MARs as a foundation. Each MAR might be further refined by an MAR-interior quadtree. MARQT$_{k,sl}^{c,a}$ can be considered both as an evolution of the MBRQT^{c,a} approach as well as the RecMAR$_{k,sl}$ approach being enhanced with MAR-interior quadtrees.

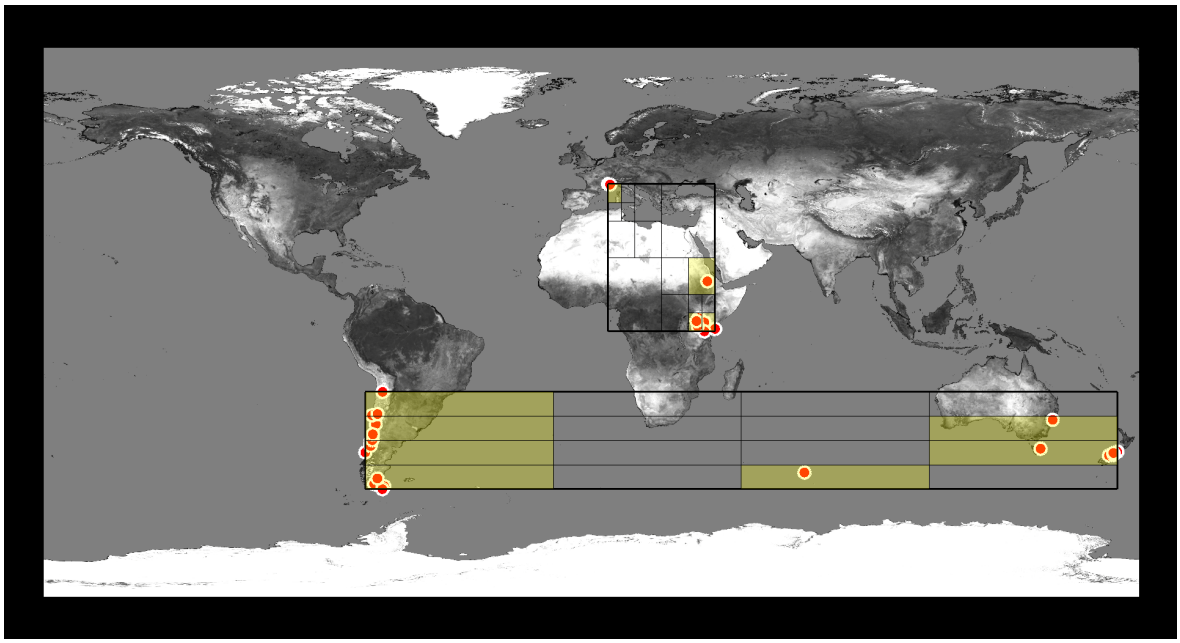


Fig. 25: Visualization of the example resource summarized by $\text{MARQT}_{2,5.0}^{16,1.0}$.

In the first step, up to k MARs are determined by the greedy, recursive algorithm which is also applied for $\text{RecMAR}_{k,sl}$ (see section 4.1). In the second step, for each MAR with a surface area greater than a, an MAR-interior quadtree is built which is dependent on the parameters c and a. These quadtrees are condensed. See Figure 25 for an example visualization of $\text{MARQT}_{k,sl}^{c,a}$.
In the bit vector of the resource summary, the single MARs and their eventual refining quadtrees are captured sequentially. At the head of the bit vector, the bounds of the first MAR are captured ($4 \cdot 32$ bits). If its surface area is greater than a, the data block of the MAR-interior quadtree immediately follows. It is encoded exactly as for the $\text{MBRQT}^{c,a}$ resource summaries for both the LQ and the CBLQ scheme. Thereafter, the data of the remaining MARs and their eventual refining quadtrees is captured in the same way, one after another.

This completes the presentation of the approaches which are evaluated for the distributed application scenario. For a quick overview, Table 1 once again provides a listing of how the 14 different summarization approaches can be classified. Note that the Grid_r approach (which represents the foundation of the $\text{GridQT}_r^{c,a}$ approach) is also depicted for the sake of completeness—even though it is not evaluated in section 7. Furthermore, classes of hybrid approaches which make no sense are flagged with an $\otimes$ symbol. Basically, this applies to all conceivable combinations in which adaptive global space partitioning is utilized as a refinement. It should also be noted that from a certain conceptual vantage point, uniform space partitioning and data partitioning using quantization are very similar to each other since both work on basis of a regular grid which is imposed onto the (sub)space to describe. Furthermore, the summaries' bit vector struc-

Table 1: More detailed classification of which category the single summarization approaches fall into. Note that any approach not listed in the 'solitary'-column is a hybrid approach.

2nd step→ 1st step↓	solitary	DP (using quantization)	SP (uniform)	SP (adaptive, global)	SP (adaptive, local)
DP	MBR $\text{RecMAR}_{k,sl}$			$\otimes$	$\text{MBRQT}^{c,a}$ $\text{MARQT}^{c,a}_{k,sl}$
SP (uniform)	(Grid_r)			$\otimes$	$\text{GridQT}^{c,a}_r$
SP (adaptive, global)	$\text{UFS}_{n,cc}$ KD_n	KDMBR^b_n $\text{KDMAR}^{b,k}_n$ $\text{DFS}^b_{n,cc}$		$\otimes$	$\text{KDQT}^{c,a}_n$
SP (adaptive, local)	$\text{QT}_{c,a}$	$\text{QTMBR}^b_{c,a}$ $\text{QTMAR}^{b,k}_{c,a}$		$\otimes$	

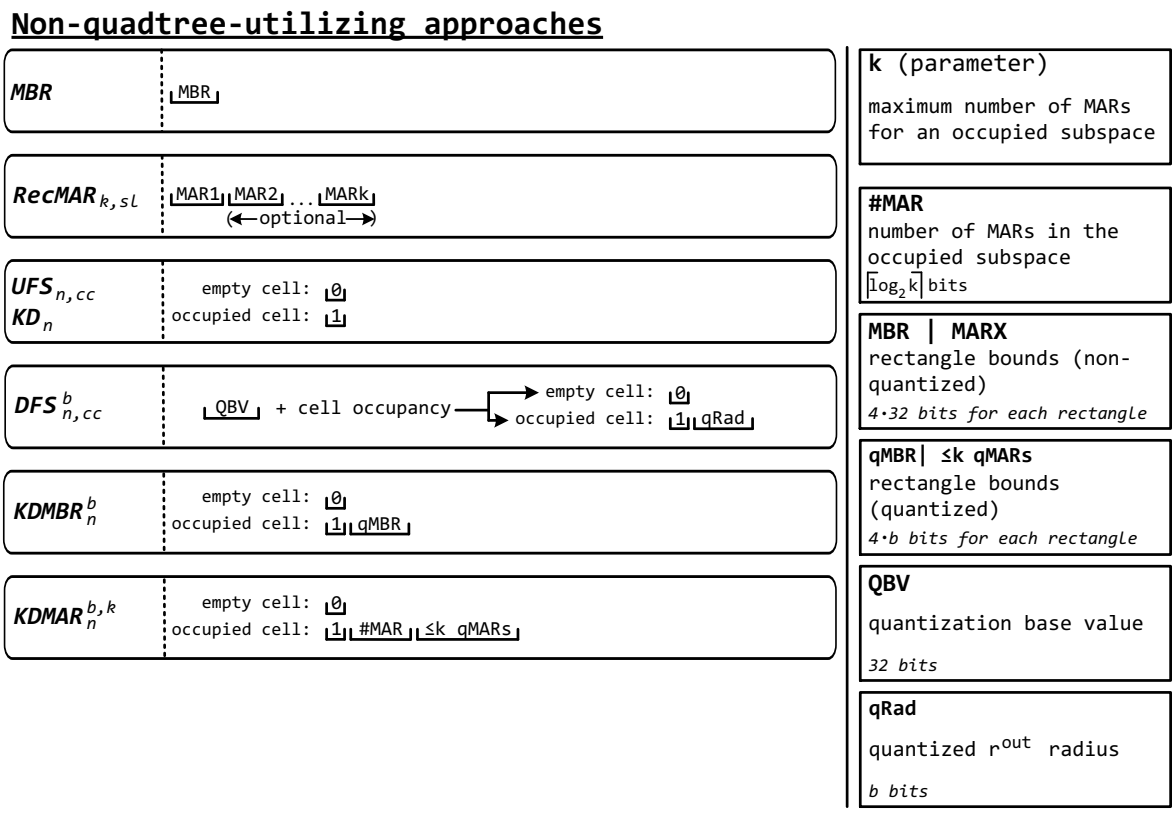


Fig. 26: Structure of the bit vectors representing the resource summaries for the different non-quadtree-utilizing approaches.

tures are depicted explicitly in Figure 26 (for the non-quadtree-utilizing approaches) and Figure 27 (for the quadtree-utilizing approaches).
At different points in the descriptions of the summarization approaches, explanations were given why specific methods outlined in section 2.2 to section 2.4 have not been considered in this work (such as for example the

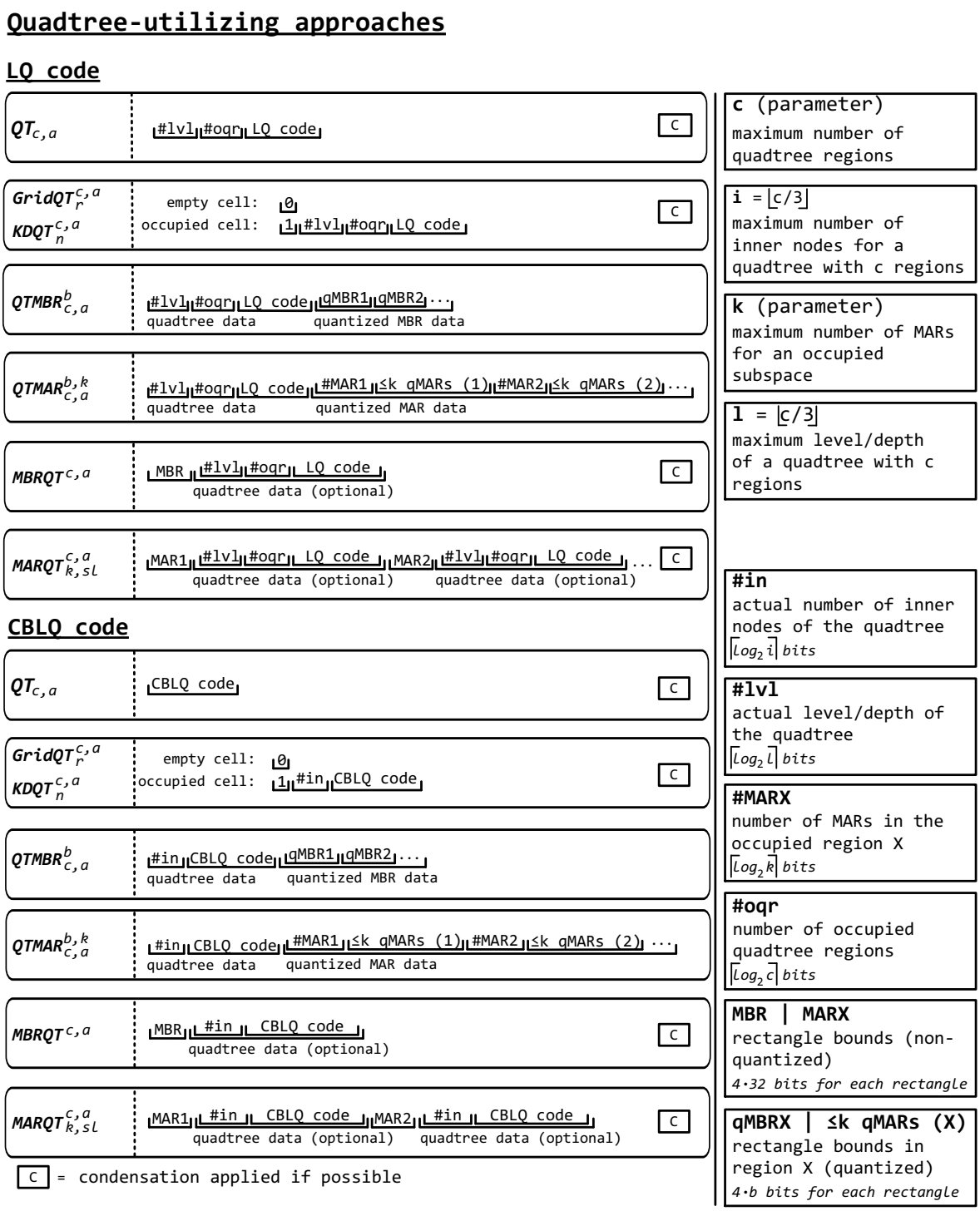


Fig. 27: Structure of the bit vectors representing the resource summaries for the different quadtree-utilizing approaches.

non-uniform grid). Generally, it has been shown in prior work ([Henrich and Blank 2010; Kufer 2012; Kufer et al. 2012; Kufer et al. 2013; Kufer and Henrich 2014; Blank et al. 2016; Kufer and Henrich 2017]) that dividing the data point set to describe into groups is the most suitable strategy. Within this strategy, working with axis-aligned rectangles (full-precision or quantized) or the utilization of space partitioning were found to be the most promising indexing concepts. The use of kmeans++ clustering has

been demonstrated to be inferior compared to $\text{RecMAR}_{k,sl}$ (see [Kufer et al. 2012]). In [Kufer 2012], the utilization of (eventually multiple) spheres as well as (single) convex hulls (both exact as well as uniformly and adaptively sampled) were shown to be less suitable than other approaches presented in section 4. Furthermore, it should not be forgotten that, for example, multidimensional data structures also pursue additional goals besides the most accurate indexing of areas, e.g. a certain memory utilization of the disk pages. As explained in section 2.4.3, e.g. the sliding-mid-point k-d-tree may move the splitting hyperplane towards the data points to avoid empty cells (and thus empty nodes). Such an approach is completely unsuitable for our summarization approaches as we require a spatial delineation of the data point clouds which is as accurate as possible. This is also the reason why for k-d-trees, data-dependent split strategies seem less suitable than the applied distribution-dependent split strategy assuming a uniform data point distribution. Furthermore, the concrete application of a data-dependent strategy would lead to the introduction of additional parameters, further increasing the complexity of the approaches.

5. RESOURCE SELECTION

With the summarization approaches presented in section 4, the problem of how the 'spatial content' of a resource can be represented (apart from the coordinates of the data points themselves) has been covered. The next task to solve is the resource selection problem. On the basis of the resource summaries[91], the resources administering the information to answer a concrete query must be determined in a resource selection process—which has to be oriented towards the query type. Assuming precise kNN queries for spatial point data in a distributed search system (i.e. the true k nearest data points with respect to a given query point have to be retrieved from the resource network), it is pivotal to prune as many irrelevant resources as possible to minimize the communication costs in the network. Therefore, an efficient resource selection process for kNN queries has to consist of two parts: First, an adequate resource ranking with respect to the given query point must be created. Based on this, the determination of the k nearest neighbors to the query point can take place.

In general, the ranking decides on in which order the resources are contacted for their relevant data. Usually, only few resources contribute to the result of a kNN query—k resources at a maximum, with k usually being small compared to the number of resources in the network. Often, the number of relevant resources is even considerably smaller than k. Ideally, the relevant resources are ranked highest while all the irrelevant resources are ranked below. This allows for the fastest possible termination of the kNN algorithm. Of course, such a perfect ranking requires perfect information—which is a utopian assumption. Consequently, the resources have to be ordered by their *estimated* relevance for a given query—which is derived from the information provided by the respective resource summaries. The more accurate the information provided by the resource summaries, the more precise of a resource ranking can be obtained and the more effective the pruning of irrelevant resources can be conducted.

Concretely, we are concerned with spatial data. Since in general, the resource summaries index areas in which the resource's data points are located, it is possible to determine a lower bound for each resource with respect to the query point—the smallest possible distance of a data point administered by the resource is the MINDIST (discussed in section 2.5) between resource summary and query point. These lower bound distances are used to prune resources in the kNN algorithm—in case the lower bound distance of a resource is greater than the distance of the (current) k-th nearest neighbor, the resource can be discarded from search. Furthermore, it is also very common to use these lower bound distances for the ranking—sorting the resources in ascending order by their MINDIST in a priority queue [Kriegel et al. 2007, p. 81]. An alternative for rectangle-based, non-

[91] Obviously, also the directly represented resources have to be considered for determining the kNN. Nevertheless, they are dealt with separately (see section 6.4.1).

quantizing summarization approaches is a ranking by the MINMAXDIST discussed in section 2.5.
The remainder of this section is organized as follows: In section 5.1, the algorithms for deciding on a resource ranking are described. In section 5.2, an algorithm for *exactly* determining the kNN of a given spatial query point on the basis of a resource ranking is described.

5.1. Conceptual Resource Ranking Algorithms

Generally, a ranking algorithm takes an unordered set of resources as input and returns these as an ordered list. The greater the relevance of a resource for a query, the higher the resource should be ranked. In our spatial domain, the MINDIST between a bounded region and a query point is an optimistic estimation of this relevance since it assumes that the data points located in this region are as close to the query point as possible. Accordingly, the MINMAXDIST for rectangular regions is a pessimistic estimation. For the determination of the ranking order, the optimistic estimation, the pessimistic estimation, or combinations of both may be applied [Böhm et al. 2001, p. 24]. We apply an optimistic ordering since in related domains, advantages for the optimistic ordering are reported (see [Roussopoulos et al. 1995, p. 78]) and because the MINMAXDIST is only applicable for minimum bounded rectangular regions (i.e. every face of a rectangle has to contain a data point). Consequently, it is inapplicable for $\text{UFS}_{n,cc}$, $\text{DFS}^{b}_{n,cc}$, and hybrid approaches relying on quantized rectangles.
Aside from both Voronoi-based approaches, all other summarization approaches index one or several rectangular regions where the data points of a resource are located in. Therefore, for all of them, the same ranking algorithm is applied which utilizes the MINDIST from the rectangular region(s) to the query point to define a total order on the resources. A corresponding, conceptual ranking algorithm is described verbally in the following and depicted more formally in algorithm 1.

> Each resource is represented as a list of rectangular regions which are reconstructed from the information encoded in the respective resource summary[92] (line(s) 2, 6-10). These representations can be cached to avoid having to rebuild them for each query since the regions depicted in the summaries are independent of the concrete query. Afterwards, the lists of `R-Entries` which represent the resources in the ranking process are created (line(s) 3, 11-16). For each rectangular region, an `R-Entry` is built. An `R-Entry` captures the MINDIST of the rectangular region to the query point q and the size of the region's surface area. The `R-Entries` of a resource are sorted in ascending order by a) MINDIST to q and b) minimum surface area (in case of equal MINDISTs; line 14).

[92] Plus the separately transmitted information about the underlying space partition for the approaches utilizing global space partitioning.

With each resource being represented by its sorted list of `R-Entries`, the actual resource ranking commences (line(s) 4, 17-37).
A total order on the resources is defined by a pairwise comparison of two resources res_a and res_b. In order to avoid unequal list sizes, the smaller list of `R-Entries` is filled with dummy entries until its size matches the size l of the larger list (lines 20-22).[93] To decide on the ranking between res_a and res_b, the respective `R-Entries` res_{a_i} and res_{b_i} at the same list index i, $0 \leq i < l$, are compared in succession (lines 23-32). If the MINDIST of res_{a_i} is smaller than the MINDIST of res_{b_i}, res_a is ranked higher. In case of equal MINDISTs, res_a is ranked higher if the surface area of res_{a_i} is smaller than the surface area of res_{b_i} (line 27). If no *decision1* can be made by comparing the `R-Entries` at index i, i is incremented and the `R-Entries` at the next index position are compared alike. If the comparison of the entire lists of `R-Entries` does not lead to a *decision1*, res_a is ranked higher than res_b if res_a administers more data points and therefore is 'bigger' than res_b (*decision2*, lines 33-36). If res_a and res_b are of the same size, a random ranking decision is made (line 37).

For the Voronoi-based approaches, two different ranking mechanisms are evaluated. The *spatial domain ranker* is an adaption of the ranking algorithm just described to polygon and sphere data ($\rightarrow$ covering radii information of $\text{DFS}^b_{n,cc}$). The *metric-domain-like* ranker conducts a resource ranking that would also be applicable in the metric domain because it is solely based on distances between the query object and the sites. Thus, we also take up the red thread from section 2.6.3 where a verification of the postulate that the *use of the existing coordinate information will lead to the best results in the low-dimensional spatial domain* has been announced.
(a) *Spatial domain ranker*. The polygons of the Voronoi cells can be constructed by knowledge of the sites' coordinates.[94] On the basis of the binary information about cell occupancy in a resource summary, the one or several polygonal regions containing the resource's data points (i.e. the resource's indexed areas) can be obtained. Aside from being based on polygonal areas rather than on rectangular areas, for $\text{UFS}_{n,cc}$, the corresponding ranking algorithm is the same as for the rectangular-based summarization approaches described above.
For $\text{DFS}^b_{n,cc}$, the spatial domain ranking is also based on an `REntry` for each indexed region. Nevertheless, the determination of its MINDIST to

[93] Note that this is only assumed for descriptive clarity in the verbal and formal depictions of the ranking algorithms. In the implementation, the lists are compared to each other as they are. The creation of suchlike dummy entries would lead to huge amounts of unnecessary object instantiations and significantly increase the ranking algorithms' runtimes.

[94] For building the polygons of the Voronoi cells, we use `JVoroTreemap` ([Nocaj and Brandes 2012]) which is a fast standalone `Java` library that can easily handle Voronoi diagrams of thousands of sites.

ALGORITHM 1: Conceptual algorithm for the spatial domain ranker, divided into several procedures.

Data:

q	the query point
L_r	list of resources, initially unranked
res	resource ($res \in L_r$)
$l_{res.reg}$	list of indexed regions in which res contains its data points
$l_{res.RE}$	list of `R-Entries` representing res in the ranking process
res_a	resource a to be compared
res_b	resource b to be compared
$l_{res_a.RE}$	list of `R-Entries` representing res_a in the ranking process
$l_{res_b.RE}$	list of `R-Entries` representing res_b in the ranking process
res_{a_i}	i-th entry in $l_{res_a.RE}$, representing the i-th closest region of res_a to q
res_{b_i}	i-th entry in $l_{res_b.RE}$, representing the i-th closest region of res_b to q

Output: L_r as a ranked list of resources

Algorithm `rankingProcess`(L_r, q)
  buildResourceRepresentations(L_r)
  buildREntriesForTheResources(L_r, q)
  rankResources(L_r)
  return L_r

Procedure `buildResourceRepresentations`(L_r)
  for res in L_r **do**
    $l_{res.reg}$ = reconstructRegionsFromResourceSummary(res)
    res.setRepresentation($l_{res.reg}$)
  end
  `// the resource representations can be cached beforehand since they are query-independent`

Procedure `buildREntriesForTheResources`(L_r, q)
  for res in L_r **do**
    $l_{res.reg}$ = res.getRepresentation()
    $l_{res.RE}$ = getSortedListOfREntries($l_{res.reg}$, q)
    res.setREntries($l_{res.RE}$)
  end

Procedure `rankResources`(L_r)
  `// define a total order on` L_r `by pairwise comparisons of two resources` res_a `and` res_b`; the depiction of the ranking is limited to a comparison of` res_a `and` res_b
  $l_{res_a.RE}$ = res_a.getREntries()
  $l_{res_b.RE}$ = res_b.getREntries()
  if the size of the two lists $l_{res_a.RE}$ and $l_{res_b.RE}$ differs **then**
    add dummy `R-Entries` to the smaller list until the list sizes are equal
  end
  $i = 0$
  while $i < |l_{res_a.RE}|$ **do**
    $res_{a_i} = l_{res_a}.RE$.get(i)
    $res_{b_i} = l_{res_b}.RE$.get(i)
    $decision_1$ = compareREntries(res_{a_i}, res_{b_i})
    if $decision_1$ can be made **then**
      report better ranked resource
    end
    i += 1
  end
  $decision_2$ = compareResourceSizes(res_a, res_b)
  if $decision_2$ can be made **then**
    report better ranked resource
  end
  report random resource to be ranked better

q as well as of its surface area must be adapted. This is because for each occupied Voronoi cell, besides the polygon, there is also the quantized covering radius ball centered at the cell's site—the 'cluster ball'—in which the cell's data points are located. Hence, the actual data-point-containing region is the intersection of the cell's polygon and the cluster ball—for which we require a lower bound distance and an (estimation) of the surface area size.[95] Generally, lower bounds should be as large as possible. Therefore, given the cell's polygon p, the cell's covering ball b, and the query point q, the region's MINDIST to q is calculated as $max(dist(p,q), dist(b,q))$. The region's surface area is simply estimated by the quantized covering radius.[96]

(b) *Metric-domain-like ranker*. The ranking is based on the sorted list of sites $L_{q,s}$, sorted in ascending order by their distance to q. Originally, this ranking scheme has been proposed by Eisenhardt et al. ([Eisenhardt et al. 2006]) for approximated kNN queries and distributed metric access methods. It only considers the cell occupancy information encoded in the resource summaries. This is done in the order specified by $L_{q,s}$.[97] Therefore, the computational costs are lower since only point-to-point distances and no surface area sizes must be calculated (instead of point-to-polygon distances and polygonal surface area sizes as for $\text{UFS}_{n,cc}$, or point-to-polygon and point-to-sphere distances as for $\text{DFS}^b_{n,cc}$; the sizes of the spherical surface areas for $\text{DFS}^b_{n,cc}$ are assessed by the covering radii information of the

[95] Since polygons and balls are based on different types of geometry (line-based versus curve-based), there is no practicable way of representing their intersection for joint distance and surface area calculations. Therefore, we conduct separate calculations or estimations.

[96] Note that the surface areas of the Voronoi cell polygons are not very expressive in this regard since it is a global space partition. Hence, the polygons are the same for all resources.

[97] Hence, the covering radius information provided by the $\text{DFS}^b_{n,cc}$ resource summaries is only used for pruning purposes in the kNN algorithm but not for the ranking process of the metric-domain-like ranker.

Note that we conducted experiments where only the (non-quantized) cluster balls of occupied Voronoi cells (and *not* the polygons or sites of the cells) have been used to determine a resource ranking in the same way as for the spatial domain ranker, i.e. based on a) the MINDIST between the cluster balls and q, and b) the size of the surface area of the cluster balls (given by the covering radius). Such a ranker is also applicable in the metric domain since it is solely based on distance information. However, it led to significantly worse results for $\text{DFS}^b_{n,cc}$ compared to a site-based ranking.

Blank discusses that the site-based ranker (called *stable* in [Blank 2015]) results in a resource ranking based on lower bound distances arising from the Voronoi-like space partition while the cluster-ball-based ranking (called *mindist* in [Blank 2015]) utilizes lower bound distances derived from the cluster balls—which possibly strongly overlap the borders of their Voronoi cells and therefore might be less precise [Blank 2015, p. 139]. The measurements in [Blank 2015, p. 137f]—although being for high-dimensional metric data rather than for low-dimensional spatial data—also resulted in that the site-based ranker is superior compared to the cluster-ball-based ranker.

corresponding cluster balls and are not calculated explicitly).[98] Being only based on point-to-point distances is what makes this ranker applicable in the metric domain. Conceptually, the location of a cell's site is the location which can be expected *on average* for a data point located in this cell. Therefore, a site-based ranking corresponds to a more pessimistic ordering compared to the 'optimistic' spatial domain ranker. A conceptual algorithm for the metric-domain-like ranker is described verbally in the following and depicted more formally in algorithm 2. Only the *cc* closest sites are considered for the ranking.

Before the actual ranking commences, the cell occupancy information has to be extracted from the resource summaries (line(s) 2, 6-10).[99] Similar to the spatial domain ranker, caching can be applied—the cell occupancy information is query-independent.
For the actual resource ranking, the list of sites $L_{q,s}$ is sorted in ascending order by distance to q (line 3). At the head of the list is the site which is closest to q (and therefore defines the cell which contains q). At the tail of the list is the site whose distance from q is the greatest.[100] A total order on the resources is defined by a pairwise comparison of two resources res_a and res_b, taking $L_{q,s}$ and cc as additional input (line(s) 4, 11-27). To decide on the ranking between res_a and res_b, the cell occupancies are considered sequentially in the order defined by the positions j of the corresponding sites in $L_{q,s}$, $0 \leq j < cc$ (lines 14-22). If res_a maintains at least one data point in the cell of site s_j while res_b does not, res_a is ranked higher. If both res_a and res_b do (not) maintain at least one data point in the cell of site s_j, the next site of $L_{q,s}$ is considered ($\rightarrow j \mathrel{+}= 1$). If cc sites $s_j \in L_{q,s}$ have been considered without leading to a *decision1*, res_a is ranked higher than res_b if res_a is bigger than res_b (*decision2*, lines 23-26). If res_a and res_b are of the same size, a random ranking decision is made (line 27).

5.2. A Conceptual kNN Algorithm

The next step in the resource selection process is to truly determine the k nearest neighbors to the query point q. This determination is based on the ordered list of resources returned by the ranking algorithms discussed in section 5.1. The precise kNN algorithm is implemented as an iterative range query for which the query radius q_{rad} is adjusted in every iteration.

[98] Note though that the polygonal distance and surface area values need to be calculated only once for each Voronoi cell in each ranking process since the space partition is global (i.e. the same for all resources) when utilizing Voronoi-based approaches.

[99] This is trivial for $\text{UFS}_{n,cc}$ since the corresponding bit vector depicts exactly this information. For $\text{DFS}^b_{n,cc}$, the bit vector has to be decoded considering the incorporated quantized covering radius information for occupied cells.

[100] Note that the i-th closest site to q does not necessarily define the i-th closest cell to q (also see Figure 28).

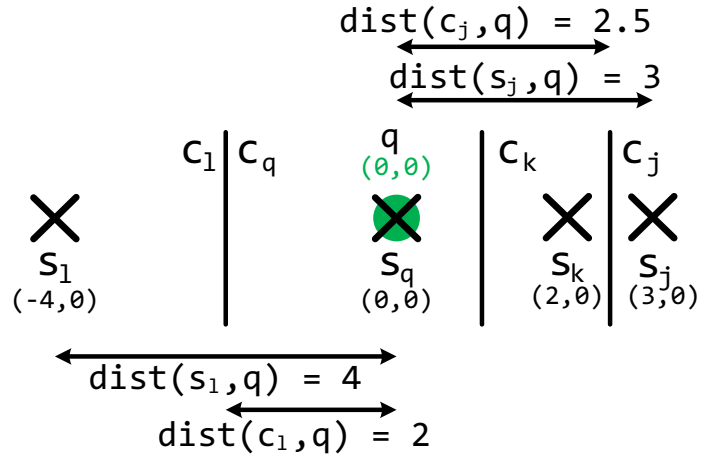


Fig. 28: Exemplary depiction of a situation where the i-th closest site to the query point q does not define the i-th closest Voronoi cell to q. Although the site s_l has a greater distance to q than s_j ($dist(s_l, q) = 4$; $dist(s_j, q) = 3$), the cell c_l defined by s_l is closer to q than the cell c_j defined by s_j ($dist(c_l, q) = 2$; $dist(c_j, q) = 2.5$).

Hereby, q_{rad} is dependent on the temporary result set of the k closest data points found so far.[101] In each round, q_{rad} is set to the distance of the k-th closest data point of this temporary result set. The parallelism of the search scenario can be exploited by contacting n_{rp} resources ($n_{rp} \geq 1$) per round. The basic principle behind the exact kNN algorithm is pruning. The pruning of resources is based on the lower bound distances derivable from the resource summary information, i.e. its MINDIST to q. The MINDIST of an entire resource to q is the minimal MINDIST of all the areas indexed by its summary. Resources whose MINDIST is greater than q_{rad} are pruned from search—there is no possibility that such a resource may administer data points which are part of the query result.[102] A conceptual algorithm for precise kNN search is described verbally in the following and depicted more formally in algorithm 3.

[101] This temporary result set might contain an arbitrary number of data points from the final result set, i.e. the true kNN.

[102] Note that also for the resources ranked by the metric-domain-like ranker, the pruning is based on all the spatial information that is available—since this allows for the fastest pruning possible. The correctness of the results could theoretically also be guaranteed by applying the *Double-Pivot Distance Constraint* (for both $UFS_{n,cc}$ and $DFS^{b}_{n,cc}$) and the *Range-Pivot Distance Constraint* (for $DFS^{b}_{n,cc}$) discussed in section 2.6.3. Nevertheless, results obviously would be worse since for these, cell boundaries are only lower-bounded based on the 'bilateral' relationship between the site of the query cell s_q and the other sites s_i, $q \neq i$. In contrast, the available spatial information allows for an exact determination of cell boundaries. Though it is out of scope for this thesis, note that both metric pruning rules could also be used for the filter-step in a multi-step kNN query processing approach (see for example [Kriegel et al. 2007, ch. 2]) since computationally, they are significantly cheaper than determining exact point-to-polygon distances.

ALGORITHM 2: Conceptual algorithm for the metric-domain-like ranker, divided into several procedures.

Data:

q	the query point
S	set of sites defining the Voronoi diagram
cc	number of sites to consider for the ranking
s_j	j-th closest site to q
$L_{q,s}$	list of sites, sorted in ascending order by their distance to q
L_r	list of resources, initially unranked
res	resource $\in L_r$
bv_{res}	bit vector containing the cell occupancy information for res
res_a	resource a to be compared
res_b	resource b to be compared
bv_{res_a}	bit vector containing the cell occupancy information for res_a
bv_{res_b}	bit vector containing the cell occupancy information for res_b

Output: L_r as a ranked list of resources

```
1  Algorithm rankingProcess(L_r, q, S, cc)
2      buildResourceRepresentations(L_r)
3      L_{q,s} = determineSortedListOfSites(q, S)
4      rankResources(L_r, L_{q,s}, cc)
5      return L_r
6  Procedure buildResourceRepresentations(L_r)
7      for res in L_r do
8          bv_res = extractCellOccupancyInfoFromSummary(res) // trivial for UFS_{n,cc}
9          res.setCellOccupancyInfo(bv_res)
10     end
       // the cell occupancy information can be cached beforehand since it is
          query-independent
11 Procedure rankResources(L_r, L_{q,s}, cc)
       // defines a total order on L_r by pairwise comparisons of two
          resources res_a and res_b; the depiction of the ranking is limited to
          a comparison of res_a and res_b
12     bv_{res_a} = res_a.getCellOccupancyInfo()
13     bv_{res_b} = res_b.getCellOccupancyInfo()
14     j = 0
15     while j < cc do
16         s_j = L_{q,s}.get(j)
17         decision_1 = compareCellOccupancyInfo(bv_{res_a}, bv_{res_b})
18         if decision_1 can be made then
19             report better ranked resource
20         end
21         j += 1
22     end
23     decision_2 = compareResourceSizes(res_a, res_b)
24     if decision_2 can be made then
25         report better ranked resource
26     end
27     report random resource to be ranked better
```

At start, q_{rad} is set to infinity and the `topk[]` array of the (temporal) k nearest neighbors is empty (line 1). The resources are ranked by the respective ranking algorithms discussed in section 5.1 (line 2). The sorted list L_r determines the order in which the resources are queried for their data points. The list L_r is processed until all resources from L_r have been queried or pruned—and thus L_r is empty (lines 3-14).

In each round, the resource summaries of the first n_{rp} resources of L_r are examined (line 5). For each resource res, the following applies: if its MINDIST to q is greater than q_{rad}, res is pruned from search (line

6). If res cannot be pruned, its $\texttt{topk}_{res}$[] array is requested which contains res's k closest data points to q.[103] Then, the global topk[] array is updated: the (with respect to q) most distant data points in topk[] are replaced with eventually existing closer data points of $\texttt{topk}_{res}$[] (line 8). Afterwards, topk[] is sorted in ascending order by distance from q (line 9).
After each round, the n_{rp} resources which have been examined (i.e. they have been contacted or pruned) are removed from L_r (line 12) and q_{rad} is set to the distance of the current k-th nearest neighbor for the next round (line 13). When L_r is empty and the algorithm ends, the k nearest neighbors have been determined and are stored in topk[]—which now represents the final query result and is returned (line 15).

Note that after the initial ranking, no re-ranking of resources is applied. Theoretically, a re-ranking of resources could be conducted after each round. Nevertheless, for the spatial domain ranking, a re-ranking would have no effect since it is solely based on the lower bound distances which are also used for pruning.
For the metric-domain-like ranker, it could have effects if and only if in advance of conducting the re-ranking, it would be determined which Voronoi cells c_i are in query range and if subsequently, only the corresponding Voronoi sites s_i would be considered for the re-ranking.[104] As an example, consider the situation depicted in Figure 28. Initially, in $L_{q,s}$, the site s_j is ranked better than s_l because it is closer to q. Therefore, a resource res_a administering data points in the cells c_q and c_j is ranked higher than a resource res_b administering data points in the cells c_q and c_l. If while querying q_{rad} is reduced such that $2 \leq q_{rad} < 2.5$, s_j can be considered irrelevant (since the query ball does not intersect c_j anymore)—while s_l must not (as c_l is still intersected). Therefore, res_b would now be ranked higher than res_a if a re-ranking by the metric-domain-like ranker would be conducted. However, changes in the ranking positions during re-rankings are rather seldom cases: Test measurements did not result in noteworthy differences between a sole initial ranking and an initial ranking with additional re-rankings after each round.[105] Hence, also for the metric-domain-like ranker, no re-ranking is applied. Depending on if and which caching mechanisms are applied, a re-ranking could also significantly increase the computational costs of the query processing.

[103] If res administers less than k data points, it simply returns all of its data points. Of course, in a concrete implementation, the contacted resource would only transmit its data points whose distances to q are smaller than q_{rad}. Nevertheless, for the sake of descriptive clarity, we assume the $\texttt{topk}_{res}$ array to be transmitted.

[104] The conceptual metric-domain-like ranking algorithm, as it is depicted, considers all of the cc closest sites to q. This is because an infinite initial query radius and only a single conduction of the ranking are assumed for the conceptual kNN algorithm.

[105] In fact, our test measurements resulted in slight advantages for the sole initial ranking (the greater n for $\text{UFS}_{n,cc}$, the bigger the advantage).

Instead of setting q_{rad} to infinity at the beginning of the kNN algorithm, q_{rad} can be strongly reduced by calculating an upper bound distance for the k-th nearest data point to q based on the information provided by the *resource descriptions*. This might result in a faster pruning of resources and reduced memory-requirements for the resource selection process. Details are discussed in section 6.4.2.

ALGORITHM 3: Conceptual algorithm for precise kNN queries.

Data:

k	the desired number of NNs
q	the query point
q_{rad}	the query radius
L_r	initially unranked list of resources
L'_r	list of resources which are examined in the current round
n_{rp}	number of resources queried in parallel
$topk[\,]$	global result array of the *temporal* k nearest neighbors to q
$topk_{res}[\,]$	local result array (of length $\leq k$) of a resource

Output: sorted array $topk[\,]$ of the *true* k nearest neighbors to q

```
1  q_rad = ∞; topk[] = ∅
2  L_r = rankResources(q, L_r)
3  while L_r ≠ ∅ do
4  |  L'_r = fetchNextN_rpResources(n_rp, L_r)
5  |  for res in L'_r do
6  |  |  if resourcePruningNotPossible(res, q, q_rad) then
7  |  |  |  topk_res[] = queryResource(res, k, q, q_rad)
8  |  |  |  topk[] = update(topk[], topk_res[])
9  |  |  |  topk[] = sort(topk[], q)
10 |  |  end
11 |  end
12 |  L_r = removeFirstN_rpResources(L_r, L'_r)
13 |  q_rad = updateQueryRadius(topk[])
   |  // no re-ranking is applied
14 end
15 return topk[]
```

6. GENERAL EVALUATION ENVIRONMENT

After the resource selection has been addressed, the next and final task is the result merging: The respective query results of the single resources have to be assessed with respect to their relevance for the query, and an overall result has to be created. In the spatial domain, the result merging is trivial—the 'relevance' of a data point is simply determined by its spatial distance to the query point. Hence, all three basic tasks outlined in section 1.3 are covered.

Therefore, the evaluation of the approaches can commence. In this section, the general 'environment' of the evaluation is discussed. It is organized as follows: we briefly outline how the results are assessed (section 6.1), introduce the data collections used for the evaluation (section 6.2), present the general experimental setup (section 6.3), and discuss the optimizations conducted to augment the capability of the distributed search system (section 6.4). The concrete evaluation of all approaches for the distributed application scenario is presented section 7. In section 8, the in-depth evaluation of selected approaches is conducted.

6.1. Assessment of the Results

For each approach and each of its tested parameterizations[106] (also see section 6.3), it has to be assessed how capable the resulting resource descriptions are. Two main criteria are assessed in the evaluation:

— The *average resource fraction contacted* (*rfc*) to retrieve the k nearest neighbors to the query point in the resource network for a set of spatial queries. The fewer resources need to be queried for their data points, the better—since the communication costs in the network are reduced. The lower the *rfc* value, the better the achieved *selectivity* or *effectiveness* of a technique.

— The *average resource description size* (*rds*) which results. A small *rds* is preferable since both the amount of resource description data to be transmitted in the network as well as the general storage space requirements are reduced. For the *rds* value, the number of bytes spent for the resource descriptions is captured and averaged over the number of resources. The lower the *rds* value, the better the achieved *efficiency* of a technique.

The *rfc* and *rds* criteria can also be seen as the two dimensions of our evaluation. The optimization of both is conflicting: Usually, the more storage space is spent, the greater the spatial accuracy of an approach's resource descriptions (though this is not guaranteed since for multi-parameter approaches, the 'utility value' of changing a specific parameter value may vary between the different parameters). The higher the spatial accuracy

[106] In the following, we may refer to a specific approach/parameter-combination as 'technique'. For example, KDMBR$_n^b$ is the general approach while KDMBR$_{32}^3$ is a technique with its parameters n being set to 32 and b being set to 3.

of the resource descriptions, the fewer resources need to be contacted since the ranking is more accurate and hence less irrelevant resources are queried.
Since all of the approaches aside from the MBR approach are parameterized, the results have to be considered on two levels: What is/are the preferable parameterization/-s for the single approaches (i.e. their best technique/-s)? And based on these, how do the different approaches compare to each other? In general, a variation of an approach's parameterization affects both the resulting *rfc* and *rds* values. It is hard to decide on which is *the* 'ideal' parameterization for an approach ($\rightarrow$ intra-approach comparison). Nevertheless, if a parameterization v results in requiring less resources to be contacted for solving a query ($\rightarrow$ lower *rfc* value) while simultaneously, the resource descriptions require less storage space on average ($\rightarrow$ lower *rds* value), both in comparison to a parameterization w, it is clear to choose parameterization v over parameterization w. Apart from that, an assessment of the trade-off between *rfc* and *rds* is strongly dependent on the concrete application. Furthermore, determining a single 'ideal' parameterization is not very practicable for a comparison *between* approaches ($\rightarrow$ inter-approach comparisons). It is also desirable to compare the suitability of different approaches in different *rds* ranges (for example 'low'/'medium'/'high' amounts of storage space spent).[107]
In general, a suitable means for intra- and inter-approach comparisons of parameterizable approaches is the Skyline operator [Börzsönyi et al. 2001]. Basically, it filters out a set of 'interesting' multidimensional data points from a potentially large set of multidimensional data points [Börzsönyi et al. 2001, p. 1]. It is applicable for assessing our results if we interpret the two criteria recorded for each tested parameterization of an approach as two-dimensional data point: *rds* being the value of the x-dimension, and *rfc* being the value of the y-dimension. This way, a set of two-dimensional data points results for each approach (one data point for each parameterization). For the Skyline operator, a point is 'interesting' if it is not dominated by any other point. A point dominates another point if it is as good or better in

[107] In previous works ([Kufer et al. 2013], [Kufer and Henrich 2014]), our comparison method was as follows: First, the 'best' parameterization of an approach was determined. For this intra-approach comparison, the parameterizations for which at least one other parameterization was better in both the *rfc*- *and* the *rds*-dimension were excluded from further considerations. Then, the remaining set was sorted in ascending order by the *rds* value, resulting in a sorted list of size l ($l \geq 1$). Initially, the parameterization at list index 0 was set to be the 'currently best' parameterization *bestparam*. Afterwards, the parameterizations at list indices i ($1 \leq i < l$) were sequentially compared with *bestparam* by contrasting the percentage *rds* growth with the percentage *rfc* reduction. If the percentage *rfc* reduction was greater than the percentage *rds* growth, the parameterization at the current list index i was set to be *bestparam*. Then, the procedure continued. After the list has been processed, the 'best' parameterization for the given approach has been determined. Finally, in the inter-approach comparison, the 'best' approach was determined analogously.

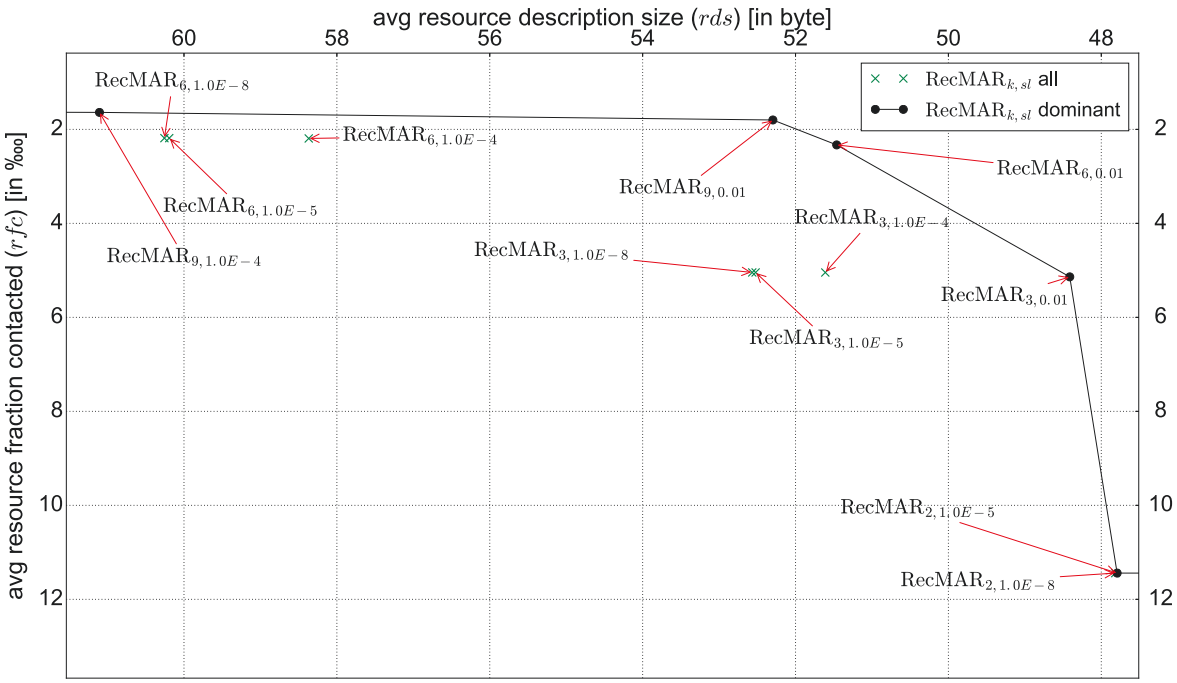


Fig. 29: Exemplary Skyline determination for RecMAR$_{k,sl}$ and the T1 collection. This is a snippet of the entire RecMAR$_{k,sl}$ Skyline. The black dots connected by the solid black lines forge the Skyline of the non-dominated parameterizations of RecMAR$_{k,sl}$.

all dimensions and better in at least one dimension [Börzsönyi et al. 2001, p. 1].
Thus, for a specific approach, a point p_x representing a parameterization x dominates a point p_y representing a parameterization y if ($p_x.rfc \leq p_y.rfc$ and $p_x.rds < p_y.rds$) or ($p_x.rfc < p_y.rfc$ and $p_x.rds \leq p_y.rds$). The set of non-dominated parameterizations builds the Skyline of an approach. Figure 29 illustrates an exemplary Skyline determination. Note that both axes are inverted in the Skyline diagrams, i.e. the more northeast-bound a point, the better the corresponding parameterization ($\rightarrow$ better selectivity, better efficiency). The Skylines determined for the respective approaches are used to compare the different approaches to each other.

6.2. Data Collections

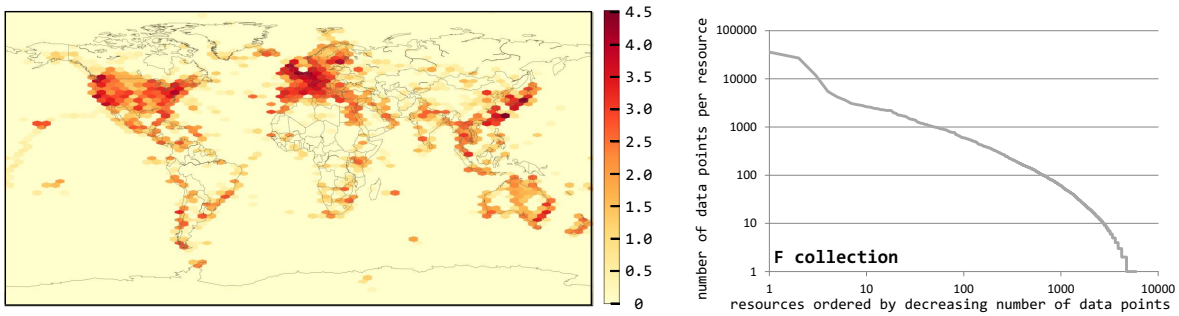
In the evaluation, three different data collections are used to test the different approaches for their robustness. These data collections are outlined in the following.
The F collection consists of 406,450 geotagged images which have been uploaded to Flickr by 5,951 different users. The images were crawled in 2008. For building resources and assigning data points to them, a resource is created for each user and the corresponding data points of his or her images are allocated to this resource. Hence, the assumption is that every user in the network operates a resource on his or her own. The distribution of the data points in the data space and the distribution of the amount of data points per resource for the F collection are depicted in Figure 30a. Gener-

ally, the spatial distribution of the data points is very uneven, i.e. regions which are very densely populated with data points vary with very sparsely populated regions. Particularly densely populated regions are discernible for parts of North America, Europe, and Japan. The distribution of data points per resource is heavily skewed: few resources administer a lot of data points and many resources administer only few data points.[108] The skewed distribution is typical for P2P data [Cuenca-Acuna et al. 2003, p. 247].

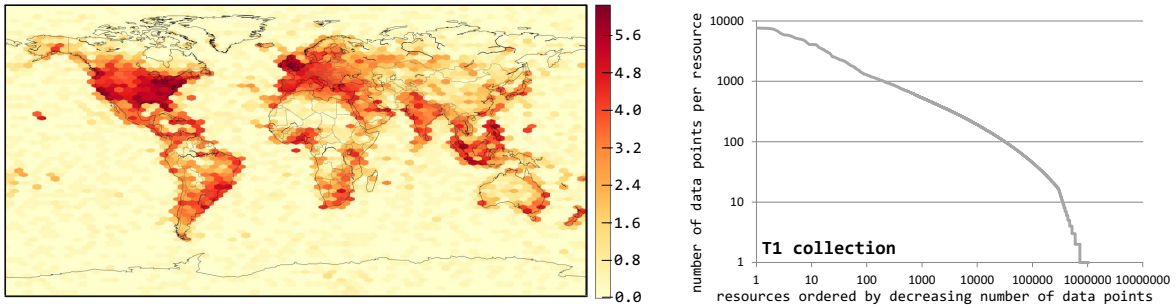
The other two collections originate from Twitter data. Between August 25th 2014 and September 1st 2014, we crawled a large amount of geotagged tweets. The tweets have been picked off in time slices of three minutes each and were restricted to the English language. The determination of a tweet's language was handled by the Twitter API. In total, 26,767,783 geotagged English tweets from 2,620,571 different users have been collected. Again, the spatial distribution of the corresponding data points is very uneven (see Figure 30b or Figure 30c, each on the left). Most data points are located in the USA and on the British Isles which is not too surprising due to the restriction to the English language. Nevertheless, there are several other regions holding a considerable amount of data points (such as continental Europe, India, parts of South East Asia, etc.). Altogether, the spatial distribution of the data points obtained from Twitter is more diverse than that of the data points obtained from Flickr. Due to the much greater number of available data points, the Twitter data (in contrast to the Flickr data) is subdivided into two dedicated data sets: a training data set and a test data set. The data points of the first ten percent of the time slices are assigned to the training data set, the data points of the remaining 90 percent are assigned to the test data set. The latter are employed as the set of data points administered in the resource network. After the division, 23,539,714 data points of 2,491,785 users and 3,019 time slices remain for the test data set. Out of this set, two collections are created, differing by the assignment of data points to resources:

— The `T1` collection. Similar to the `F` collection, the data points are assigned to resources via user ID. Again, this results in a skewed distribution of data points per resource (see Figure 30b on the right).
— The `T2` collection. The data points are assigned to resources on basis of the time slice in which they have been captured (i.e. a resource is created for each time slice). This results in a more even distribution of data points to resources (see Figure 30c on the right). As a consequence of the assignment procedure, a different application scenario has to be assumed for the `T2` collection: It is assumed that there is control over the assignment of data objects to resources but the spatial properties of the

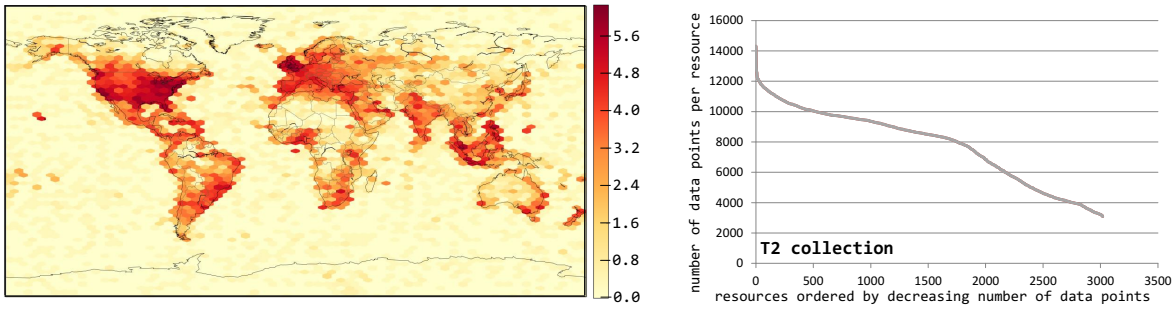
[108] Note that in the following, resources administering many (few) data points are referred to as 'big' ('small') resources. This terminology has already been used in conjunction with the ranking algorithms (see section 5.1).



(a) Distribution of the data points in the data space (left) and number of data points per resource (right) for the F collection. Both the coloring (left) and the axes (right) are log-scaled.



(b) Distribution of the data points in the data space (which is equivalent to Figure 30c, left) and number of data points per resource for the T1 collection. Both the coloring (left) and the axes (right) are log-scaled.



(c) Distribution of the data points in the data space (which is equivalent to Figure 30b, left) and number of data points per resource for the T2 collection. The coloring (left) is log-scaled. The axes (right) are linear-scaled.

Fig. 30: Distribution of the data points in the data space (respective left) and number of data points per resource (respective right) for the different collections. Note that the values on the respective left legend (i.e. the colorings) are $log_{10}(x+1)$ scaled. Therefore, the number x of data points per bin is calculated as $x = 10^n - 1$. For example, $n = 4.0$ results in $x = 9,999$.

data objects are not considered for the assignment, and that a more balanced distribution of data objects to resources is pursued. The changes with regard to the assumed application scenario are discussed in more detail in section 8.6 (where selected approaches are evaluated for the T2 collection).

Finally, Table 2 shows an overview of several key figures for the three different data collections. As already mentioned, for both the F and T1 collec-

tions, the amount of resources administering very few data points is very high. Hereby, this is even more the case for the T1 collection for which more than 50% of the resources administer only 1 or 2 data points.

Table 2: Overview of several key figures for the different data collections.

key figures→ collection↓	*# data points (dp)*	*# resources (res)*	*biggest res (# dp)*	*smallest res (# dp)*	*# res ≤ 10 dp*	*# res ≤ 2 dp*	*# res = 1 dp*
F	406,450	5,951	35,893	1	3,261 (54.8%)	1,740 (29.2%)	1,231 (20.7%)
T1	23,539,714	2,491,785	7,654	1	2,049,384 (82.2%)	1,304,038 (52.3%)	916,376 (36.8%)
T2	23,539,714	3,019	14,308	3,083	- (0%)	- (0%)	- (0%)

6.3. Experimental Setup

As previously specified, we conduct kNN queries in our evaluation—the *exact* k nearest data points with respect to a query point are to be retrieved from the network. The results are aggregated over an amount of 50 kNN queries. Hereby, k is also set 50. We assume that $k = 50$ is a reasonable number for the given application scenario as it is imputed that users search for images or short text messages associated with a certain location. For these types of documents, users are able to assess the relevance of a result document much easier and faster compared to for example larger text documents in traditional information retrieval search scenarios (where k is usually set to lower values).
The selection of the $k = 50$ query points differs for the collections originating from Twitter respectively Flickr:

— For the two collections originating from Twitter (T1 and T2), 50 data points are randomly selected from the training data set. No duplicates are allowed, i.e. no two or more query points have the same coordinates. The query points are the same for both collections. See Figure 31 (top) for a visualization of the query points' locations.

  For our application scenario, this means that external users[109] search the network for the spatially closest tweets to a location from which they issued a tweet themselves.

— For the F collection originating from Flickr, no dedicated training data set is available from which the query points could be selected. Instead, data points from the data collection itself are selected for the query points. This is done as follows: First, a random resource is selected—the probability of being selected is the same for all resources (regardless

[109]With external users we mean users which do not contribute data objects within the network.

of their respective size). Then, a random data point of the resource is selected as a query point. This procedure is repeated 50 times.[110] Again, no duplicate query points are allowed.
In the context of our application scenario, this simulates that users of the network search for the spatially closest images to an image of their own.[111] Thereby, every user has the same probability of issuing a query, there is no bias for users with big media archives. As a consequence of the selection procedure, for each query issued for the F collection, it is guaranteed that there is a NN with a distance of 0—the data point used as query point itself. See Figure 31 (bottom) for a visualization of the query points' locations.

For the F collection and the 50 selected query points, 7.74 resources contribute to the query result on average. This results in a *rfc* of $7.74/5,951 = 0.0013$ or 0.13% for the ideal selectivity—only contacting the resources contributing to the final query result. The corresponding numbers for the other two collections are as follows:

— T1: 12.92 resources contribute to the query result on average. Hence, the ideal *rfc* value is $12.92/2,491,785 = 5.185E-6$ or ~0.52 ‰.[112]
— T2: 44.56 resources contribute to the query result on average. Hence, the ideal *rfc* value is $44.56/3,019 = 0.0148$ or ~1.48%.

The evaluated parameter values of the different approaches are listed in Figure 32. It is not possible to set the parameter ranges such way that for each approach, the Skylines begin and end at identical, predetermined *rds* values. For example, for MBRQT^{c,a} and MARQT$^{c,a}_{k,sl}$, it is not possible to set the parameters such that *rds* values below the *rds* value of the MBR approach result. Other approaches are not infinitely scalable due to their computational costs (for example, the MAR computation is a very expensive process) or the limited precision of the IEEE single precision floating-point number format (for example, when building a local quadtree space partition for resources administering just one data point, the precision limits of the number format are quickly reached). In general, the parameter values are set such that

[110] In previous works ([Henrich and Blank 2010], [Kufer 2012], and [Kufer et al. 2012]), it was shown that such a selection of query points results in higher *rfc* values compared to just randomly selecting data points from the data collection as query points. This is because for the latter, it is likely to select a data point from a big resource. Big resources are preferred by the ranking algorithms (see section 5.1) and are therefore likely to be ranked higher. Also, the number of relevant resources is lower. The simple random selection also assumes a slightly different application scenario. This issue is discussed in more detail in section 8.5 of the in-depth evaluation.

[111] Note that such a specification of the query adheres to the 'query-by-example' paradigm where a user specifies a query object and the system retrieves the database objects which are similar to the query object [Theng 2009, p. 447].

[112] Note that ‰ is the 'per myriad' sign and means 'per ten thousand'.

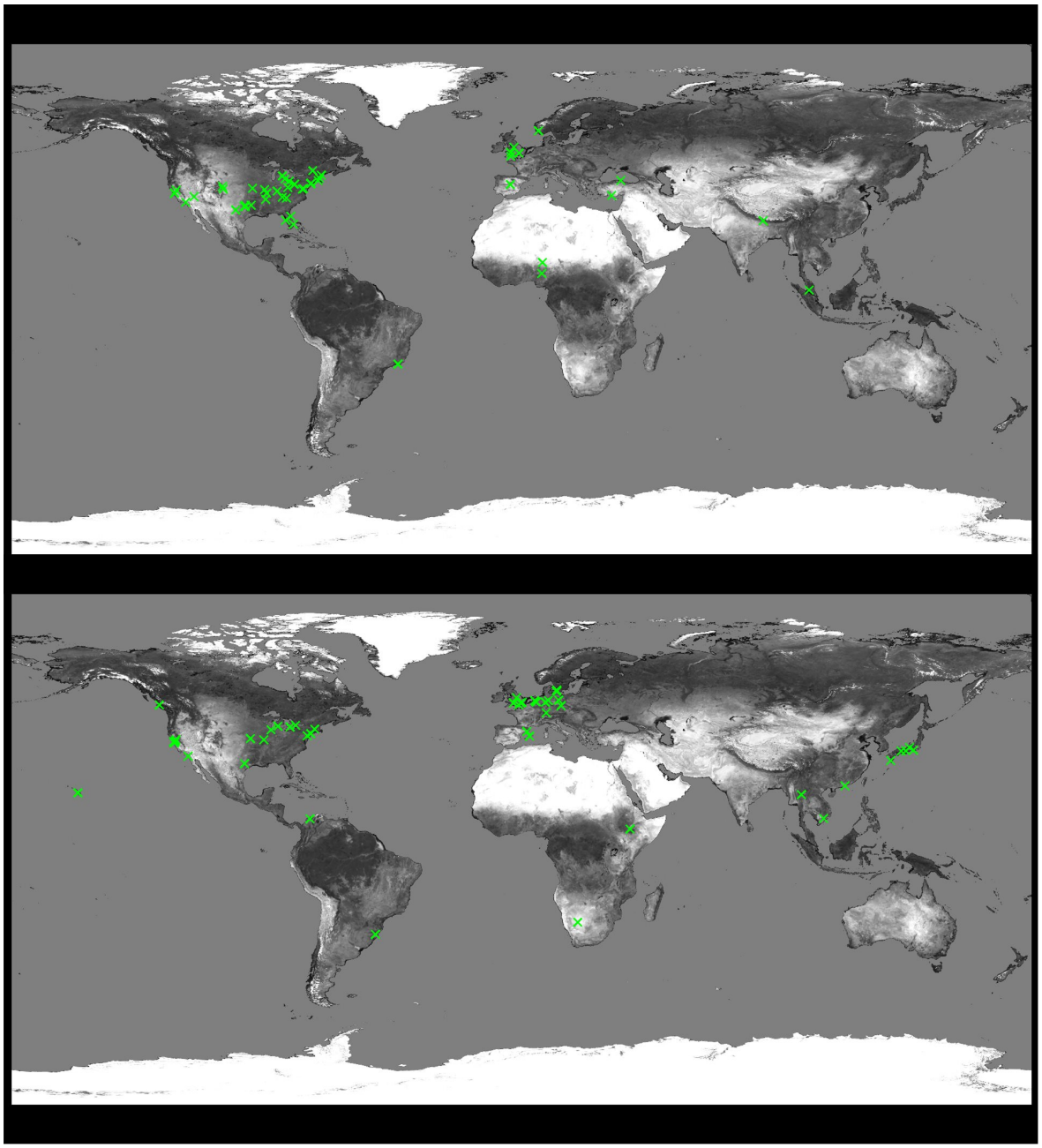


Fig. 31: Locations of the query points of the T1 and T2 collection (top) as well as the F collection (bottom). The green crosses depict the locations.

a) for each approach, *rds* values well below (if possible) and above the *rds* value of the MBR approach result,

[113] As a practicable approach for coping with the limited precision of the floating-point number format, the technical limits of the GPS (Global Positioning System) accuracy are taken into account. This is because our data originates from devices with GPS sensors. The accuracy of GPS is roughly 10 meters in outdoor environments (see [Rana et al. 2010, p. 108] and [Djuknic and Richton 2001, p. 123]). In their most accurate parameterizations, the cells of the quadtree-utilizing approaches are constructed up to a threshold surface area of $a = 1.0E{-}8$ 'square data space units' (dsu^2) in our spatial coordinate system. For a quadratic cell, this would result in a width of 0.0001 dsu. This corresponds to 11.44 meters on the equator, 8 meters when *lat* is 45°, and 7.0E-16 meters on the poles [Veness 2016].

b) the computational costs for the resource summary creation are kept at a reasonable level, and
c) the limited accuracy of the single precision floating-point number format does not cause problems.[113]

For the approaches and their parameter values listed in Figure 32, all possible combinations are tested. For example, for $\text{QTMAR}_{c,a}^{b,k}$, this results in $|c| \cdot |a| \cdot |b| \cdot |k| = 6 \cdot 5 \cdot 3 \cdot 3 = 270$ different parameterizations for which the corresponding resource descriptions are computed and the 50 kNN queries (with $k = 50$) are run.[114] The listed parameter values are tested for all evaluations involving the F and T1 collections. The T2 collection is a separate case which is discussed in section 8.6.
Some approaches include random components. This applies to the Voronoi-based approaches (selection of the sites, see section 6.4.3), the k-d-based approaches (selection of the training data points for learning the k-d space partition, also see section 6.4.3), and the quadtree-based approaches (selection of the next quadtree region to split when there are several occupied quadtree regions of equal size). For these approaches and the F collection, four runs with different seeds are conducted to minimize the effect of outliers for each parameterization (the same set of query points is used in each run). The results presented for the evaluations of the F collection (in section 8.2, section 8.4, and section 8.5) are averaged over these four runs. For all remaining evaluations of the T1 collection and the T2 collection, only one run is conducted due to the significantly larger data collections ($\rightarrow$ greater computational costs) and since generally, only marginal differences arise between the single runs for the F collection.

6.4. Adjustments for Simulating a 'Real-World-System'

For experimental results as realistic as possible, we conduct several adjustments to our implementation simulating a search system for a resource network. These adjustments are either necessarily required for the operation of a 'real-world-system' or would be conducted for its optimization.
As outlined in section 1.2, we assume the distributed search system to be implemented as a Java application—which affects the transmission of resource descriptions in the network. Furthermore, the storage space requirements for the resource descriptions can be optimized. All the adjustments concerning the resource descriptions, their associated data alongside its transmission in the network, as well as the resulting consequences for the query processing and the evaluation are outlined in section 6.4.1. An optimization of the initial query radius of the kNN algorithm is described in section 6.4.2. In section 6.4.3, the appropriate selection of sites

[114] Note that non-feasible combinations are excluded. For example, testing the $\text{UFS}_{n,cc}$ approach with $n = 32$ (number of sites) and $cc = 256$ (number of sites considered in the ranking) does not make sense. Consequently, $\text{UFS}_{32,256}$ is not tested.

MBR		no parameters
$RecMAR_{k,sl}$	k	2|3|6|9
	sl	10.0|1.0|0.01|1.0E-4|1.0E-5|1.0E-8
$UFS_{n,cc}$	n	32|64|128|256|512|2048|8192|16384
	cc	16|64|256|n
KD_n	n	32|64|256|512|2048|8192|16384
$QT_{c,a}$	c	32|64|128|256|512|1024|2048|8192
	a	1.0|0.1|0.001|1.0E-5|1.0E-7|1.0E-8
$KDMBR^{b}_{n}$	n	32|64|256|512|2048|8192
	b	3|4|6|8
$KDMAR^{b,k}_{n}$	n	32|64|256|512|2048|8192
	b	3|4|6
	k	2|3|4
$DFS^{b}_{n,cc}$	n	32|64|128|256|512|2048|8192|16384
	cc	16|64|256|all
	b	3|4|6
$GridQT^{c,a}_{r}$	r	4|8|16|32|64
	c	16|32|64|256|512
	a	1.0|0.1|0.001|1.0E-5|1.0E-7|1.0E-8
$KDQT^{c,a}_{n}$	n	32|64|128|256|512|2048|8192
	c	16|32|64|256|512
	a	1.0|0.1|0.001|1.0E-5|1.0E-7|1.0E-8
$QTMBR^{b}_{c,a}$	c	16|32|64|256|512|1024
	a	1.0|0.1|0.05|0.01|0.001
	b	3|4|6|8
$QTMAR^{b,k}_{c,a}$	c	16|32|64|256|512|1024
	a	1.0|0.1|0.05|0.01|0.001
	b	3|4|6
	k	2|3|4
$MBRQT^{c,a}$	c	16|64|256|512|1024|2048|8192
	a	1.0|0.1|0.001|1.0E-5|1.0E-7|1.0E-8
$MARQT^{c,a}_{k,sl}$	k	2|3|4
	sl	10|1.0|0.1|0.01
	c	16|64|256|512|1024|2048
	a	1.0|0.1|0.001|1.0E-5|1.0E-7|1.0E-8

a = threshold surface **a**rea

b = # of **b**its

c = max. # of **c**ells of a quadtree

cc = # of **c**entroids **c**onsidered

k = max. # of MARs

n = **n**umber of subspaces

r = # of **r**ows

sl = thre**s**ho**l**d distance rectangle center to associated data points

Note that for the hybrid techniques, the parameters of the foundation are always subscript whereas the parameters for the refinement are always superscript.

Fig. 32: Listing of the parameter values tested in the evaluations of the F and T1 collections.

respectively training data for the Voronoi-based respectively k-d-based approaches is discussed.

6.4.1. ADJUSTMENTS CONCERNING THE RESOURCE DESCRIPTION DATA (AND THEIR CONSEQUENCES). The resource descriptions themselves (i.e. the coordinates of the data points or the summaries whose encoding schemes have been presented in Figure 26 on page 96 and Figure 27 on page 97) do not depict all the data that has to be transmitted in the resource network for a real-world-system implemented in Java. There is plenty of additionally required data: The specification of the utilized resource description approach and its parameters, requests for data point coordinates or media objects from resources during the query processing, data required for the preservation of the P2P overlay network, and so on.

As a consequence, it is necessary that it can be distinguished what type of data has been received. In the Java world, an easy and efficient way towards such a distinction is to transmit the data as appropriate Java objects across the network. Java objects are sent between a sender and a recipient via sockets which require writing the corresponding data to an `ObjectOutputStream`. Java objects which are written to an `OutputStream` are serialized, i.e. the memory data structure is encoded into a serial sequence of bytes which can be inflated again. The serialization induces a storage space overhead of 27 bytes which is included in all the measurements of section 7 and section 8.[115] Obviously, this serialization overhead increases the overall amount of storage space required. Nevertheless, we also apply two optimizations to reduce the storage space spent for the resource description data itself.

As discussed first at the beginning of section 2, for 'small' resources—and depending on the approach and its concrete parameterization—summarizing a resource's spatial footprint by indexing areas might require more storage space than storing the floating-point numbers of the coordinates of the resource's data points directly. In such cases—which we deemed to be inadequate when defining the requirements for resource summaries at the beginning of section 4—the bit vector of the resource description contains the binary representation of the data point coordinates instead of a summary. Besides saving storage space, this 'direct representation' is technically also a more accurate description of the spatial footprint since the summary information is always 'only' an aggregated depiction of where a resource's data points are located at. Additionally, including the option of a direct representation is also the most suitable frame for comparing the different resource description approaches: Some approaches cope better with describing small resources, for example because of inherent properties of an approach itself or the applicability of compression due to arising low-entropy resource summary data. This is especially important if the distribution of data points per resource is very skewed such that a large portion of resources is very small—as for both the `F` and the `T1` collection (see Table 2 on page 114).

Second, all resource description information—whether directly or summary-represented— is encoded as a bit vector, in the end. Bit vectors are often compressible which can be exploited to reduce the amounts of storage space spent [Beskers and Fischer 2014, p. 608]. We require only one operation on the compressed data—its unpacking. Therefore,

[115] Note that alongside the resource descriptions, for example also resource IDs have to be transmitted in the network. Nevertheless, for such data, we assume that the network overlay software handles the appropriate transmission.

we discard specialized bit vector compression schemes[116] and resort to entropy compression, applying Java's `gzip` implementation with default parameters. The `gzip` file format is based on the DEFLATE algorithm [Deutsch 1996] which is a combination of LZ77 and Huffman coding. For each resource, the bit vector(s) containing the resource summary information as well as the bit vector containing the coordinate information are `gzip`-compressed. In the end, the option requiring the least amount of storage space is selected as a resource's description. Hence, in case a compression can reduce the storage space requirement, the compressed data is used.
All in all, for non-quadtree-utilizing approaches, there are four possibilities of how the spatial footprint of a resource is stored in the bit vector (the resource *description*), depending on if the content of a resource is described by indexed areas (the resource *summary*) or 'directly' by the coordinates of its data points (the *direct representation*), and whether the data is compressed or not. The corresponding resource description types are:

— `sum-z`: indexed areas, zipped (compressed) data,
— `sum-nz`: indexed areas, non-zipped (non-compressed) data,
— `dr-z`: coordinates of the data points, zipped data, and
— `dr-nz`: coordinates of the data points, non-zipped data.

For quadtree-utilizing *summaries*, it must additionally be differentiated if the quadtrees defining the space partitions are LQ-encoded or CBLQ-encoded, resulting in six different resource description types:

— `lq-z`: indexed areas, utilizing LQ-encoded quadtrees, zipped data,
— `lq-nz`: indexed areas, utilizing LQ-encoded quadtrees, non-zipped data,
— `dr-z`: coordinates of the data points, zipped data,
— `dr-nz`: coordinates of the data points, non-zipped data,
— `cblq-z`: indexed areas, utilizing CBLQ-encoded quadtrees, zipped data, and
— `cblq-nz`: indexed areas, utilizing CBLQ-encoded quadtrees, non-zipped data.

The information on the resource description type has to be incorporated as metadata into the Java object (which is a byte array) sent accross the network such that the associated payload of the resource description can be decoded unambiguously. For the sake of uniformity, we equally use 3 bits for encoding the resource description type for both quadtree-utilizing as well as non-quadtree-utilizing approaches. In the metadata, the (quantized) resource size (which is used in the ranking algorithms) is additionally included using 5 bits. Overall, this results in 1 extra byte in addition

[116]The specialized bit vector compression schemes—which allow the processing of a bit vector without unpacking—could be applied for global space partitioning approaches which only capture binary cell occupancy information (i.e. $\text{UFS}_{n,cc}$ and KD_n) in case packing and unpacking the bit vectors would be too costly in terms of runtime.

to the serialization overhead and the resource description payload data itself. Hence, the total data transmitted for a resource is comprised of three parts (all included in the measurements of section 7 and section 8) which are once again depicted in Table 3.
A final (and somewhat unplanned) measure to reduce the resource description sizes results from a property of the Java class `BitSet`, which is used to represent the bit vectors in our implementation. A `BitSet` instance does not have an explicit size but encodes a bit vector that 'grows as needed'. If such a `BitSet` is handed to a Java `OutputStream`, the `BitSet` (which is just as large as it is required to represent the rightmost bit) is converted into an array of bytes. As a consequence, the logical bit vector of a summary is physically clipped in a byte-aligned fashion after the rightmost bit of the logical bit vector which is set to 1. For example, an $\mathrm{UFS}_{256,cc}$ resource summary logically consists of 256 bit. If in this summary, only the first bit is set (because the resource administers only data points in the Voronoi cell with $ID = 0$), the summary bit vector is physically clipped after the first byte. This bit vector clipping is implicitly applied for all resource descriptions as they all are represented as bit vectors. However, for some approaches, it is applied relatively often whereas for others, it does not occur at all since the data is of too high entropy. This is discussed in more detail at the appropriate stages of the evaluation.

Table 3: Overview of the composition of the total amount of data transmitted for a resource description.

serialization overhead	resource description type + resource size	actual resource description information (might be compressed and clipped)
27 byte	1 byte	(total number of bytes -28 byte)

The possibility of the direct representation has consequences for the resource ranking: Resources for which the coordinates of the data points are transmitted instead of a resource summary are not ranked since the exact locations of their data points are already known. This also affects the query processing and the determination of the *rfc* values. A query is now processed in two stages:

— In the first stage, the k closest data points to the query point are determined. The locations of the data points whose coordinates are directly transmitted are known. Therefore, the corresponding resources do not have to be contacted in this stage for assessing their data points as candidates for the query result. In contrast, each resource which potentially might administer candidates for the query result (derived by the information encoded in the respective resource summaries) must be contacted for its data points in the first stage.

— In the second stage, each resource administering one of the top k data points must be contacted for the media objects associated with these data points. The result-contributing resources can be both directly as well as summary-represented resources.

This means that resources which transmit summary data *and* contribute to the final query result are contacted twice in the two-stage query processing—once for their data points, once for the media object(s) associated with the data points of the final query result. Consequently, the *rfc* value of a specific query arises as the result from:

$$(\#res_{can,sum} + \#res_{top,dr} + \#res_{top,sum}) \,/\, \#res \tag{1}$$

, with:

— $\#res_{can,sum}$: number of summary-represented resources which are contacted while processing the kNN algorithm,
— $\#res_{top,dr}$: number of result-contributing resources which transmitted the coordinates of their data points as a resource description,
— $\#res_{top,sum}$: number of result-contributing resources which transmitted their summary as a resource description, and
— $\#res$: number of resources in the resource network (cardinality of the resource set).

Such a calculation of the *rfc* value assumes an application scenario where the media objects associated with the data points are so big that it makes no sense to transmit them alongside the data points when a contact between the query processor and a resource administering *candidates* for the final query result is made. Typical media objects in such a scenario could be images or videos.
For very small media objects, it is conceivable that the media objects are transmitted alongside their associated data points when a contact between query processor and resource is made. In this case, the media objects could for example be *short* text messages. The *rfc* value would then be calculated as:

$$(\#res_{can,sum} + \#res_{top,dr}) \,/\, \#res \tag{2}$$

For the evaluation, we employ the *rfc* calculation as listed in Equation 1 since this is the more suitable approach in the given distributed application scenario (which assumes a 'multi-aspect' resource description application in which geotagged media items are administered in a P2P system, see section 1.2).
Nevertheless, it has to be noted that such an assessment of the *rfc* value potentially disadvantages approaches for which very storage-space-efficient summaries can be built. As discussed above, a resource is described by the option requiring the least amount of storage space—either a sum-

mary or the direct representation. For some approaches, it is possible to build very 'small' summaries which require less storage space than the coordinates of a *single* data point. As a consequence of the low storage space consumption, the indexed areas are typically rather coarse. This complicates the distinction between relevant and irrelevant resources, especially in comparison to knowing the exact location of the single data point. As a result, the $res_{can,sum}$ number is potentially increased. Furthermore, a summary-represented resource contributing to the final query result is contacted twice—the more resources transmit summaries, the more summary-represented resources tend to contribute to the final query result. Both aspects result in an increased overall *rfc* value. All in all, the trade-off between slightly reduced resource description sizes on the one hand and the drawbacks with respect to the *rfc* value on the other hand are unlikely to be favorable for the approaches for which a high share of summaries is transmitted. Nevertheless, it is hard to assess and also strongly application-dependent how a conceivable trade-off between the storage space requirements and the spatial accuracy of a resource description could look like. Therefore, we stick to the storage-space-driven selection of a resource's description type and the computation of the *rfc* value according to Equation 1.

6.4.2. REDUCING THE INITIAL QUERY RADIUS. In the conceptual kNN algorithm presented in section 5.2, the query radius is initially set to infinity. This initial query radius can be significantly reduced by exploiting two kinds of available information:

— From the set of directly transmitted data points, an *initial* set of k nearest neighbors can be determined.[117] The distance of the k-th nearest neighbor of this set to the query point is an upper bound distance for the *final* query result.
— It is also possible to determine an upper bound distance by the information contained in the resource summaries. The summaries index areas, each containing at least one data point. Thus, for each area, it is guaranteed that at least one data point is in MAXDIST—the distance between the query point and the farthest possible point of the indexed area [Böhm et al. 2001, p. 22]—of the query point. A summary-based upper bound for the k-th nearest neighbor can therefore be determined as follows: For each area indexed by a summary, the MAXDIST between the query point and the area is determined. The distances are then sorted in ascending order. The k-th smallest distance is the sought-for upper

[117] The (sorted) initial set of k nearest neighbors is then also used as initial `topk[]` array (which is taken as input for the kNN algorithm). To efficiently support the determination of the k nearest neighbors from the set of directly transmitted data points, a centralized multidimensional data structure like the R-tree or other approaches discussed in section 2.4.3 can be used. This multidimensional data structure has to be maintained by the query processor.

bound distance—it is guaranteed that at least k data points administered by summary-represented resources are as close or closer to the query point.[118]

For both ways of determining an upper bound, an own list of k smallest distances is maintained. Then, these lists are merged into a mutual top-k-list of smallest distances. The k-th smallest distance of the mutual list is the initial radius for the kNN algorithm. It is the smallest possible upper bound that can be determined a priori.
Theoretically, the minimization of the initial query radius has two merits: a) the memory requirements for the resource selection process (i.e. for the ranking procedure as well as the kNN determination) are reduced, and b) possibly better results for the *rfc* values.
With an infinite initial query radius, the representations of all summary-represented resources must be kept in memory for the ranking procedure. Its result (i.e. the sorted list of resources) is passed to the kNN algorithm. With a minimal initial query radius derived from upper bound distances (i.e. MAXDISTs), resources whose MINDIST is greater than this initial query radius can be pruned before ranking.
With regard to improved *rfc* values, it is as follows: When utilizing the spatial domain ranker, improved *rfc* values only occur in a rather theoretical case: If all the k nearest neighbors are from directly represented resources *and* the MINDISTS of all the summary-represented resources are greater than the initial query range. Otherwise, due to the MINDIST-based ranking, there is not much potential for the premature pruning of resources. For the metric-domain-like ranker, *rfc* improvements might occur more often—in circumstances similar to the situation depicted in Figure 28 on page 105 where the cell c_j of a higher ranked site s_j can be pruned prematurely while the cell c_l of a lower ranked site s_l cannot. For example, resources only administering data points in c_j are then eventually pruned sooner. Nevertheless, improved *rfc* values should be rare for both ranking schemes. Consequently, the main benefit is the reduced memory requirement for the resource selection process.

6.4.3. SELECTION OF THE SITES AND THE TRAINING DATA. In section 4.2, we mentioned that the capability of both Voronoi-based and k-d-based approaches depends on an 'appropriate selection' of sites respectively training data.
Before coming to the selection process, let us quickly discuss some key aspects of these approaches since the idea behind global space partitioning is fundamentally different compared to using bounding volumes or local space partitioning (which both try to minimize the indexed areas and do

[118] For $\text{DFS}^b_{n,cc}$, the MAXDIST of an indexed area (delineated by the polygon p_i and the cluster ball b_i of the Voronoi cell c_i) is calculated as follows: *min*(*MAXDIST*(q, p_i), *MAXDIST*(q, b_i))—with q being the query point. The *min* operator is used to utilize the smaller of the two MAXDISTs. This is because upper bounds should be as low as possible.

not take information into account apart from the single resource's data points): Ultimately, the pure global space partitioning approaches rely on building a large-enough amount of subspaces which allow for a high selectivity and a storage-space-efficient description of their cell occupancies. In their summaries, single bits represent the cell occupancies. Depending on the amount of global subspaces, the raw bit vectors might consume considerably large amounts of storage space. Thus, the storage space efficiency is obtained by compressing the resource summaries: Usually, a resource only administers data points in very few subspaces of the space partition. Consequently, long zero runs arise in the summaries which can be easily entropy-compressed. This also applies to hybrid approaches based on global space partitioning: The blocks of refinement information in the summary bit vector are usually of high entropy but the low entropy blocks of the basic space partition are still easily compressible.

Prerequisite for also achieving a high selectivity on the basis of global space partitioning is a space partition which is adjusted to the spatial distribution of the entire collection's data points: For densely populated regions, many fine-grained subspaces are required. For sparsely populated regions, coarse-grained subspaces are acceptable. Ideally, the global number of data points which are located in a subspace is the same for each subspace. Due to the operating principle of the ranking algorithms and the kNN algorithm, the worst case for a global space partitioning approach is a space partition where the query subspace is populated with a very large number of data points (which are then likely administered by many different resources). For hybrid approaches based on a global space partitioning, such situations can potentially be mitigated by the refinement—but a well-adjusted foundation is preferable, too.

With regard to the concrete selection, it is as follows: For the `T1` and `T2` collections, a training data set is available (see section 6.2) from which sites

[119] In a cooperative real-world system, resources can be asked for a sample of their data points to build an appropriate training data set.

[120] Note that in order to match our simple selection strategy in a real-world-application, greater (smaller) sample sets would have to be requested from bigger (smaller) resources.

[121] It has been shown by Blank and Henrich ([Henrich and Blank 2010]) that for Voronoi-based approaches, a selection of sites right from the data collection itself is superior to a selection from external sources which feature data point distributions that approximate the data collection's distribution. This is intuitively clear. In [Kufer 2012], [Kufer et al. 2012], and [Kufer et al. 2013], it has been shown that the same applies to the selection of training data points to learn k-d space partitions. For the selection of sites, we also experimented with two pivot selection techniques suggested in the context of nearest neighbor search in metric spaces—the selection of N random groups and the incremental selection [Bustos et al. 2003]—but both were inferior to the simple random selection. This is because basically, these pivot selection techniques try to maximize inter-site distances which is not what we require (since in regions with a very high global data point density, we require very fine-grained cells and thus sites being very close to each other). The pivot selection methods are rather suited for application in situations in which only rather few sites are required whereas we require a considerable amount of sites.

and training data can be drawn. For the F collection, no training data set is available. Since we assume a cooperative environment (see section 1.3), sites and training data are drawn from the data collection itself. Hence, for all three data collections, a cooperative environment is assumed.[119] From the respective underlying sets, the training data for learning the k-d space partition is simply selected at random. For the selection of the sites for the Voronoi diagrams, the procedure is almost the same—the only constraint is that no two sites with exactly the same coordinates are allowed. These simple strategies[120] result in a very suitable adjustment of the respective space partitions to the spatial distribution of the data collection.[121] For the k-d space partition, the assignment of IDs to the subspaces is simply based on the order in which they were created. This means we do not proactively assign IDs to subspaces according to space-filling curves or the like. For the Voronoi space partition, we sort the sites in ascending order by a) their *long* value and b) (in case of equal long values) by their *lat* value.[122] The IDs are assigned to the Voronoi cells based on this order.

[122]This allows for determining the Voronoi cell of a data point by considering significantly less than all of the n sites. Hence, it is an optimization to speed up the summary calculation for Voronoi-based summaries.

7. EXPERIMENTAL RESULTS—ANALYSIS OF THE T1 COLLECTION

In this section, the evaluation starts. As outlined in section 1.5, the evaluation is split into two parts: First, all 14 *summarization approaches* presented in section 4 are evaluated as *resource description approaches* for the T1 collection. In the second part, selected approaches are evaluated in-depth. The reason for the division is that due to the sheer amount of approaches, their parameterizations, available data collections, and so on, there cannot be an extensive in-depth evaluation of all approaches for all the data collections. Instead, the main focus of the first part of the evaluation is on the T1 collection since it features the long tail distribution of data points to resources which fits the assumed distributed application scenario. Furthermore, it is much bigger than the F collection (which features a similar distribution).

The remainder of section 7 is organized as follows: In section 7.1.1, a general, joint overview of the results for the T1 collection and all the approaches is given. Afterwards, the approaches are divided into groups based on their characteristics—similar approaches are placed into the same group—for a more detailed and specific comparison within these groups (section 7.2 to section 7.6). The analyses in these sections are mostly the discussions of observations and interesting phenomena. Out of each of the groups, the most promising and/or interesting approaches are selected as representatives for the in-depth evaluation in section 8.

7.1. General Overview of the Results

In the following, a general overview of the results for the T1 collection is presented. The primary tool for comparing the different approaches is the Skyline operator. Therefore, the Skylines of the approaches are contrasted first (see section 7.1.1). Further assessments involve the comparison of different aggregated key figures for the resource description sizes as well as the transmission of the resource descriptions in the network (see section 7.1.2). Finally, the *rfc* values are compared which result if for each approach, approximately as much storage space is used on average as for the MBR approach (i.e. the *rds* values are similar). See section 7.1.3. All these comparisons involve all the 14 evaluated approaches—the division into groups is applied in the subsequent sections.

[123] With regard to the specification of storage space sizes in byte (B), the following applies: If it is referred to storage space sizes of individual resource description instances or a set of instances that have a particular property in common, no decimal places are specified since the byte sizes can only be integers, anyway. For example, “for approach *X*, no resource descriptions below a size of 33 B exist”. If, on the other hand, the given storage space sizes are average values over a set of instances or if storage space ranges are specified, *up to* two decimal places are provided. This also applies to ranges of integer values. For example, “a zoomed depiction of the Skylines for *rds* values between 32.0 B and 50.0 B is shown in Figure *Y*”.

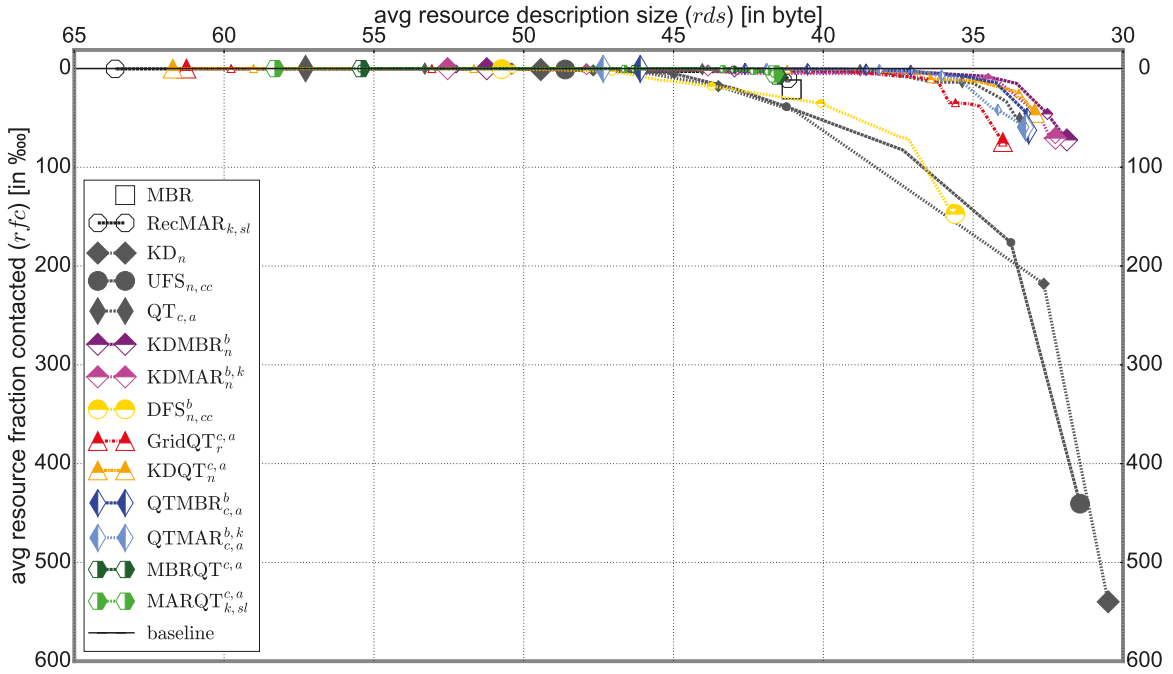


Fig. 33: Full Skylines of all the 14 approaches evaluated for the T1 collection alongside the baseline which represents the theoretical optimum for the *rfc* value. A zoomed in version for the *rds* range between 32.0 B and 50.0 B as well as *rfc* values of 0.005 (or 50.0 ‰) and lower can be found in Figure 34.

Note that if several parameterizations of an approach result in exactly the same *rfc* and *rds* values, we select the parameterization which in theory should result in the *coarsest* resource descriptions to be part of the approach's Skyline. For example, the results for KDQT$^{16,0.001}_{32}$, KDQT$^{16,1.0E-5}_{32}$, KDQT$^{16,1.0E-7}_{32}$, and KDQT$^{16,1.0E-8}_{32}$ are all *rds* = 33.51 byte (B)[123] and *rfc* = 23.84 ‰[124]. Since KDQT$^{16,0.001}_{32}$ is the parameterization which in theory should result in the coarsest resource descriptions, it is selected to be part of the Skyline. The other three parameterizations are discarded. Generally, for the T1 collection, suchlike selections are required for KDQT$^{c,a}_n$, GridQT$^{c,a}_r$, and MARQT$^{c,a}_{k,sl}$.

7.1.1. ANALYSIS OF THE SKYLINES. At first, the resulting Skylines are assessed. In Figure 33, the full Skylines of all the 14 approaches evaluated for the T1 collection are shown. The markers on the respective Skylines represent the non-dominated parameterizations building the Skyline (the 'Skyline parameterizations'). They are sampled for a better visibility of the Skyline pathways, i.e. not every single Skyline parameterization is depicted by its own marker. The start and end points of the respective Skylines are depicted by bigger markers. Note that both the x-axis (depicting the average resource description sizes, *rds*, in byte) as well as the y-axis (depicting the average resource fraction contacted, *rfc*, in ‰) are inverted,

[124]Note that '‰' is the 'permyriad'-sign, meaning 'per ten thousand'. Therefore, an *rfc* value of 1 ‰ means that 1 of 10,000 resources is contacted for solving a query on average.

i.e. the more north-east-bound a parameterization or whole Skyline, the better.
In general, for all approaches, the Skylines are roughly shaped like asymptotic curves that gradually approach the baseline which represents the theoretical optimum selectivity (*rfc* = 5.19E-6 or ~0.052 ‰, corresponding to 12.92 contacted resources on average). For each Skyline, in the early stage (i.e. at the bottom right end of the respective curves), a slight *rds* increase results in heavy *rfc* improvements, while in the final stage (i.e. at the top left end of the curves), even a heavy *rds* increase results in only minor *rfc* improvements.
Despite a similar general shape, the individual Skylines of the diverse approaches differ in particular properties:

— At what *rds* values does the Skyline start/end (*rds* range of the Skyline)?
— In which *rfc* ranges is the Skyline?
— How big are the *rfc* improvements for small *rds* investments in the early stage (steepness of the asymptotic curve), how much *rds* must be invested for minor *rfc* improvements in the final stage (flatness of the asymptotic curve)?
— At what *rds* values does the transition—the 'elbow' or 'knee' of the curve, where the *rds* investments required for additional *rfc* improvements start to grow disproportionately—between the early stage and the final stage commence? How long does the transition take?
— How close does the Skyline approach the baseline?

Altogether, these properties contribute to the general suitability of an approach for the given distributed application scenario and the T1 collection. The *rds* range of the different Skylines is from a minimum of 30.5 B (for KD_{32}) to a maximum of 63.6 B ($RecMAR_{9,1.0E-5}$). For the 2,491,785 resources of the T1 collection, this results in an amount of 76.0 MB to 158.5 MB of resource description data to be transmitted to each node in the network which shall be capable of issuing queries. The non-compressed transmission of all data point coordinates would account for 255.6 MB.[125] The raw spatial data to be stored on query-capable nodes—i.e. without the serialization overhead and without considering any auxiliary data structures for indexing the data or associating the data points with their resources—is at least the transmitted data without the serialization overhead, i.e. ranging from 8.7 MB (KD_{32}) to 91.2 MB ($RecMAR_{9,1.0E-5}$) when utilizing resource descriptions while requiring 188.3 MB for the non-compressed data point coordinates. Consequently, by using resource descriptions, the amount of data to be transmitted to each query-capable node is reduced

[125] The 23,539,714 data points of the T1 collection require 8 B each. Additionally, for each of the 2,491,785 resources, the 27 B serialization overhead for transmitting the data in the network have to be taken into account, resulting in 255.6 MB overall. Note that the conversions rest upon a *Base-10*-definition, i.e. 1000 is used as a conversion factor for converting e.g. B to KB.

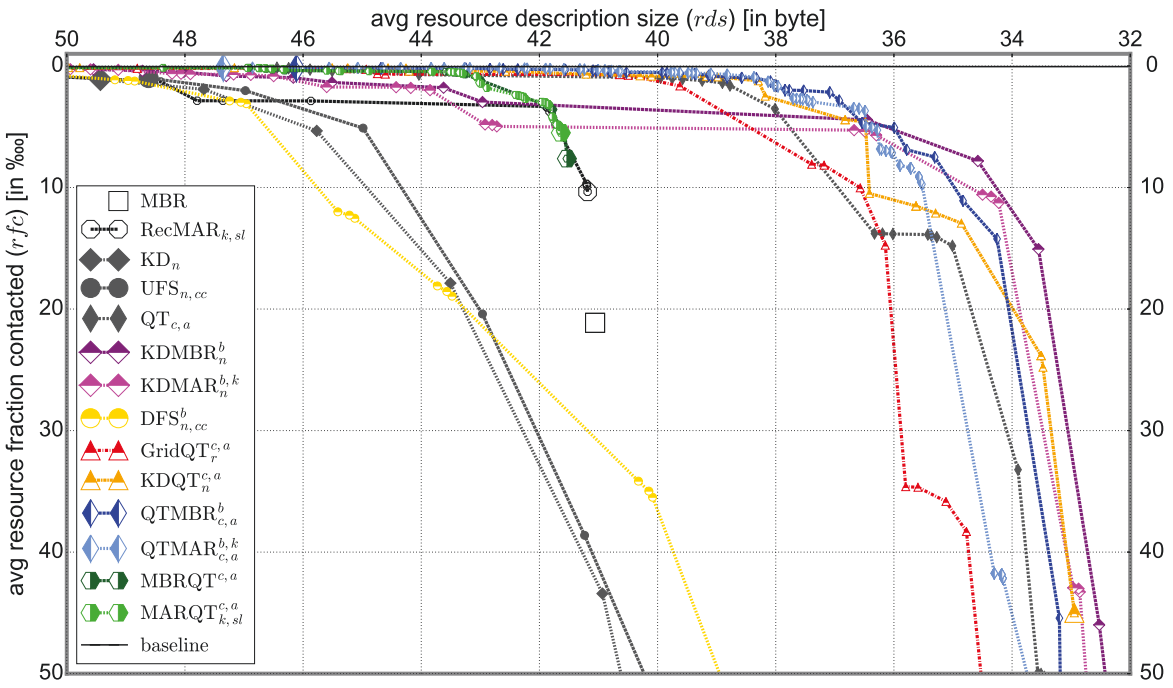


Fig. 34: Zoomed in version of Figure 33 for a better comparability of the different approaches for high selectivity (i.e. low *rfc* values).

by between 38.0% (RecMAR$_{9,1.0E-5}$) and 70.3% (KD$_{32}$). The raw data to be stored is reduced by between 51.6% (RecMAR$_{9,1.0E-5}$) and 95.4% (KD$_{32}$). Note that the T1 collection with its large amount of resources administering only few data points is not well-suited for reducing the total amount of data by the use of summaries: for many parameterizations of each approach, a direct representation is chosen as resource description for a large share of resources.

The *rfc* range of the different Skylines is from a maximum of 539.7 ‰ (corresponding to ~134,382 contacted resources, KD$_{32}$) to a minimum of 0.095 ‰ (corresponding to ~23.6 contacted resources, KDQT$^{512,1.0E-8}_{8192}$)—which is very close to the theoretical optimum of ~12.92 resources (especially when considering that summary-represented resources which contribute to the top 50 data points are considered to be contacted twice, see section 6.4.1).

Looking at the concrete Skylines, it can be observed that the Skylines of KD$_n$, UFS$_{n,cc}$, and DFS$^b_{n,cc}$ are far worse than those of the other approaches. At comparable *rds* values, they even perform worse than the very basic MBR approach. Only at already relatively large *rds* values of ~48.0 B, they converge to the Skylines of some of the poorer other approaches (also already see Figure 34 for a more detailed depiction).

For the full-precision rectangle-*based* approaches (RecMAR$_{k,sl}$, MBRQT^{c,a}, and MARQT$^{c,a}_{k,sl}$), the Skylines begin only at *rds* values beyond that of the MBR approach since in principle, they are storage-space-lower-bounded by the MBR—which is the starting base of these three approaches before refining the summaries in different ways. Therefore, *rds* values below the one

of the MBR approach are not possible for these approaches. Consequently, the Skylines of RecMAR$_{k,sl}$, MBRQT^{c,a}, and MARQT$^{c,a}_{k,sl}$ can be thought of to be extendable to the MBR 'Skyline'.[126] At their early stages, the Skylines of these three approaches are significantly below those of the other parameterizable approaches (except KD$_n$, UFS$_{n,cc}$, and DFS$^b_{n,cc}$), but at least the Skylines of MBRQT^{c,a} and MARQT$^{c,a}_{k,sl}$ quickly converge to those of the other quadtree-utilizing approaches afterwards (at the latest when ~43.0 B *rds* are reached).
The Skylines of the approaches not explicitly mentioned so far are all relatively close together for the whole *rds* range. Nevertheless, there are still noticeable differences and changes in relative performance for varying *rds* values. For example, KDMBR^b_n and KDMAR$^{b,k}_n$ are best for very small *rds* values but fall off for higher *rds* values.
Figure 34 shows a zoomed in depiction of the Skylines such that the performance differences between the single approaches become more apparent for the range between 32.0 B *rds* and 50.0 B *rds*. The Skyline markers are no longer sampled, i.e. each Skyline parameterization is represented by a corresponding marker. At ~50.0 B *rds*, all approaches provide an *rfc* of 1.1 ‰ (corresponding to ~274 contacted resources) or better. In general, for a basic set of 2,491,785 resources from which ~12.92 resources contribute to the query result on average, contacting 270 resources for retrieving the *exact* top 50 data points of the network appears to be fairly selective. For comparison, the MBR approach requires contacting ~5,265 resources while spending an *rds* of 41.1 B. This shows that for all parameterizable approaches, it is possible to achieve an acceptable selectivity while simultaneously offering a sufficiently small storage space footprint: 50.0 B *rds* results in 124.6 MB to be transmitted or 57.3 MB raw data to be stored, corresponding to a storage space reduction of ~51.3 % for the data to be transmitted respectively ~69.6 % for the raw data to be stored. Nevertheless, some approaches achieve a higher selectivity with considerably less storage space spent on average. In Figure 34, it can be seen that especially quadtree-utilizing approaches all in all seem to fit the `T1` collection very well.

7.1.2. ANALYSIS OF THE KEY FIGURES FOR THE SKYLINE PARAMETERIZATIONS. In the following, some key figures (mostly aggregated ones) concerning the set of non-dominated Skyline parameterizations for each approach are presented and analyzed. These key figures are displayed in Table 4. Note though that the aggregated key figures are fairly coarse: For each approach, they are aggregated over the entire set of Skyline parameterizations, and the characteristics of an approach may vary greatly depending on the chosen parameterization. Thus, they are more suited for developing a basic feeling for the different approaches rather than for com-

[126] Since the MBR approach is not parameterized, the corresponding MBR 'Skyline' consists of only one anchor point.

Table 4: Key figures related to the Skyline parameterizations of all 14 approaches evaluated for the T1 collection. The key figures are grouped into the number of Skyline parameterizations (row 2), *rds*-related key figures (rows 3 to 8), and transmission-method-related key figures (rows 9 to 14). For the latter two groups, the given values are aggregated over all the respective Skyline parameterizations.

approach↑ key figure→	MBR	RecMAR$_{k,sl}$	KD$_n$	UFS$_{n,cc}$	QT$_{c,a}$	KDMBR$_n^b$	KDMAR$_n^{b,k}$	DFS$_{n,cc}^b$	GridQT$_r^{c,a}$	KDQT$_n^{c,a}$	QTMBR$_{c,a}^b$	QTMAR$_{c,a}^{b,k}$	MBRQT^{c,a}	MARQT$_{k,sl}^{c,a}$
non-dominated parameterizations	1 (100%)	16 (66.7%)	7 (100%)	8 (100%)	27 (56.3%)	15 (62.5%)	34 (63.0%)	24 (100%)	28 (18.7%)	43 (20.5%)	38 (31.7%)	64 (33.7%)	26 (61.9%)	87 (20.1%)
avg *rds* [in byte]	41.06	41.2 - 63.6	30.5 - 49.4	31.4 - 48.6	33.3 - 57.3	31.9 - 51.2	32.3 - 52.5	36.0 - 50.7	34.0 - 61.3	32.9 - 61.7	33.1 - 46.1	33.3 - 47.3	41.5 - 55.4	41.6 - 58.3
min *rds* [in byte]	36	36 - 36	29 - 29	29 - 29	29 - 36	30 - 32	30 - 32	33 - 33	30 - 36	29 - 35	33 - 36	33 - 36	36 - 36	36 - 36
25%-quant. *rds* [in byte]	36	36 - 36	29 - 36	31 - 36	33 - 36	31 - 36	31 - 36	35 - 36	33 - 36	32 - 36	33 - 36	33 - 36	36 - 36	36 - 36
median *rds* [in byte]	44	44 - 44	30 - 44	32 - 44	33 - 44	32 - 44	32 - 44	36 - 44	34 - 44	33 - 44	33 - 40	33 - 39	44 - 44	44 - 44
75%-quant. *rds* [in byte]	44	44 - 76	31 - 60	32 - 60	33 - 64	33 - 60	33 - 62	36 - 64	34 - 76	34 - 73	33 - 49	33 - 51	45 - 60	45 - 63
max *rds* [in byte]	44	60 - 172	33 - 1,617	33 - 1,376	38 - 2,105	75 - 10,612	107 - 14,992	48 - 2,234	194 - 12,232	160 - 26,901	58 - 2,798	83 - 6,255	50 - 2,199	71 - 2,106
sum-z/lq-z [in %]	0	2.3 (0.002- 15.4)	20.8 (0- 36.3)	18.7 (0- 37.1)	0.02 (0- 0.4)	18.3 (0- 34.4)	16.5 (0- 34.1)	17.4 (0- 34.5)	5.8 (0- 24.0)	4.5 (0- 21.0)	0	0	0.003 (0- 0.04)	< 0.001 (0- 0.01)
sum-nz/lq-nz [in %]	47.7	37.6 (3.5- 47.4)	37.9 (0.6- 100)	38.8 (0.7- 100)	72.6 (15.6- 98.6)	40.3 (1.1- 100)	43.8 (1.1- 100)	31.3 (0.6- 72.7)	41.7 (0- 99.3)	44.1 (0.3- 99.1)	65.1 (44.4- 99.9)	71.8 (44.3- 99.9)	41.1 (22.2- 47.1)	40.4 (15.9- 46.9)
dr-z [in %]	0	0.2 (0.002- 1.6)	0.06 (0- 0.3)	0.09 (0- 0.5)	$\ll 0.001$ (0- 0.001)	0.08 (0- 0.4)	0.06 (0- 0.4)	0.2 (0- 0.9)	0.4 (0- 1.5)	0.2 (0- 1.2)	$\ll 0.001$ (0- < 0.001)	$\ll 0.001$ (0- < 0.001)	< 0.001 (0- 0.1)	0.01 (0- 0.3)
dr-nz [in %]	52.3	60.0 (52.5- 79.5)	41.2 (0- 65.2)	42.4 (0- 65.3)	17.7 (0- 61.3)	41.4 (0- 67.0)	39.7 (0- 67.6)	51.1 (27.2- 68.7)	44.5 (0- 76.7)	43.6 (0- 75.4)	30.5 (0- 45.8)	25.4 (0- 45.7)	53.7 (52.3- 62.1)	54.2 (52.6- 67.1)
cblq-z [in %]	-	-	-	-	0.2 (0- 2.8)	-	-	-	4.3 (0- 15.4)	2.7 (0- 11.6)	0.001 (0- 0.004)	< 0.001 (0- 0.003)	0.04 (0- 0.6)	0.02 (0- 0.5)
cblq-nz [in %]	-	-	-	-	9.5 (1.4- 20.3)	-	-	-	3.3 (0- 9.7)	4.9 (0.2- 12.3)	4.5 (0.1- 10.4)	2.7 (0.1- 10.4)	5.1 (0.6- 15.2)	5.4 (0.6- 16.5)

pletely understanding their characteristics and properties. The key figures for the Skyline parameterizations are displayed in Table 4.

How to read Table 4. Since the aggregated key figures may not be easily comprehensible without further clarification, we first explain how the figures displayed in Table 4 are determined and how they are to be interpreted.

The ‘non-dominated parameterizations’-row (row 2) shows how many Skyline parameterizations exist for each approach (corresponding to the number of ‘anchor points’ on the respective Skylines) and how great their share is with respect to all tested parameterizations for the single approaches. For e.g. RecMAR$_{k,sl}$, 16 out of the 24 tested parameterizations are part of the Skyline.
All the other rows display resource-description-size-related (*rds*-related) or transmission-method-related key figures of the respective Skyline parameterizations. The key figures are aggregated over the entire set of Skyline parameterizations for each approach, i.e. they depict the characteristics of the different approaches on a very high and general level.
Consider RecMAR$_{k,sl}$ (column 3) as an example. The *rds*-rows (rows 3 to 8, i.e. ‘avg rds’-row to ‘max rds’-row) *summarize* the descriptive statistic *rds* values[127] (min *rds*, median *rds*, etc.) for the Skyline parameterizations of RecMAR$_{k,sl}$. In a single cell, the range of occurring values is displayed. The base data for these aggregations (which are displayed in the RecMAR$_{k,sl}$-column in Table 4) is shown in Figure 35. For example, the average resource description sizes of the 16 Skyline parameterizations of RecMAR$_{k,sl}$ range from 41.2 B (for RecMAR$_{2,10.0}$) to 63.6 B (RecMAR$_{9,1.0E-5}$). The average resource description sizes of the other RecMAR$_{k,sl}$ Skyline parameterizations are all somewhere in between. In the ‘avg rds’-row of column 3 in Table 4, this range of occurring average *rds* values is depicted (as ‘41.2 - 63.6’). As another example, the maximum resource description sizes which occur for single parameterizations of RecMAR$_{k,sl}$ range from 60 B (*minimal* maximum resource description size, occuring for e.g. RecMAR$_{2,10.0}$, RecMAR$_{2,1.0}$, etc.[128]) to 172 B (*maximal* maximum resource description size, for e.g. RecMAR$_{9,10.0}$, RecMAR$_{9,1.0E-5}$, etc.). The occurring maximum resource description sizes for the other RecMAR$_{k,sl}$ Skyline parameterizations are all somewhere in between.[129] The summarized information on this is displayed in the ‘max rds’-row of column 3 in Table 4 (as ‘60 - 172’). Generally, the *rds*-rows (rows 3 to 8) of the RecMAR$_{k,sl}$-column in Table 4 summarize the boxplots of the single RecMAR$_{k,sl}$ Skyline parameterizations depicted in Figure 35.

[127] Note that for determining descriptive statistic values, we utilize the Apache Commons Math library (version 3.4.1): http://commons.apache.org/proper/commons-math/, last visit: 21.11.2018.

[128] This means that for e.g. RecMAR$_{2,10.0}$, the most storage-space-intensive description of any resource of the T1 collection requires 60 B.

[129] Note that for RecMAR$_{k,sl}$, the maximum resource description sizes are only dependent on the maximum number of rectangles built to summarize a resource—which is determined by parameter k. Therefore, there are several RecMAR$_{k,sl}$ Skyline parameterizations featuring the *minimal* maximum resource description size (smallest maximum boxplot tick in Figure 35 $\rightarrow$ parameterizations with $k = 2$) as well as the *maximal* maximum resource description sizes (biggest maximum boxplot tick in Figure 35 $\rightarrow$ parameterizations with $k = 9$).

In rows 9 to 14 (i.e. 'sum-z/lq-z'-row to 'cblq-nz'-row), the way the resources transmit their description data is summarized. For example, for the MBR approach, 47.7% of the resources transmit their descriptions as non-zipped summaries ('sum-z/lq-z'-row), while 52.3% transmit the non-zipped coordinates of their data points as a direct representation ('dr-nz'-row). For the parameterizable approaches, the single rows depict how many percent of the resources—averaged over all the Skyline parameterizations—use the corresponding transmission method to transmit their data (big number at the top of the cell) as well as what are the minimally and the maximally occuring shares of this transmission method over all the Skyline parameterizations (bracketed small numbers in the middle—minimum share—and at the bottom—maximum share—of the cell).
As an example, consider again RecMAR$_{k,sl}$. In Figure 35, the bars of the barchart plot display the shares of the single transmission methods for its Skyline parameterizations. For example, for the non-zipped transmission of summary data ('sum-nz'-option), the shares range from 3.5% (RecMAR$_{9,1.0E-5}$) to 47.4% (RecMAR$_{2,10.0}$). For all other RecMAR$_{k,sl}$ Skyline parameterizations, the share of resources transmitting non-zipped summary data is somewhere in between. Averaged over all the Skyline parameterizations of RecMAR$_{k,sl}$, the share of resources transmitting non-zipped summary data is 37.6%. In Table 4, these three numbers are depicted in the 'sum-nz/lq-nz'-row of column 3 as '37.6% (3.5% - 47.4%)'. Generally, the rows 9 to 14 of the RecMAR$_{k,sl}$-column in Table 4 summarize the bars of the single RecMAR$_{k,sl}$ Skyline parameterizations in Figure 35.
Hence, in Table 4, from rows 3 to 8, the descriptive statistic *rds* values of the single approaches' Skyline parameterizations are aggregated (i.e. the boxplots exemplarily depicted for RecMAR$_{k,sl}$ in Figure 35). From rows 9 to 14, the transmission method shares are aggregated and averaged (i.e. the bars exemplarily depicted for RecMAR$_{k,sl}$ in Figure 35).

Explanatory Power of Table 4. Due to the high level of aggregation, the key figures displayed in Table 4 are not particularly useful for a concrete and detailed analysis of the different approaches: The characteristics of the results for a specific approach can vary greatly depending on its parameterization and, for example, whether 'small' or 'big' average resource description sizes result. Nevertheless, it is possible to observe some diverging characteristics of the various approaches. For example: For which approaches, a tendency of transmitting zipped summary data instead of non-zipped summary data exists? Which approaches tend to build very storage-intensive resource descriptions in extreme cases? And so on. Therefore, it facilitates to develop a feeling for the different inherent properties of the single approaches, and the differences between them.

Analysis of the Different Approaches' Share of Skyline Parameterizations. As first key figure from Table 4, the respective share of Skyline parameterizations of the different approaches is evaluated ('non-

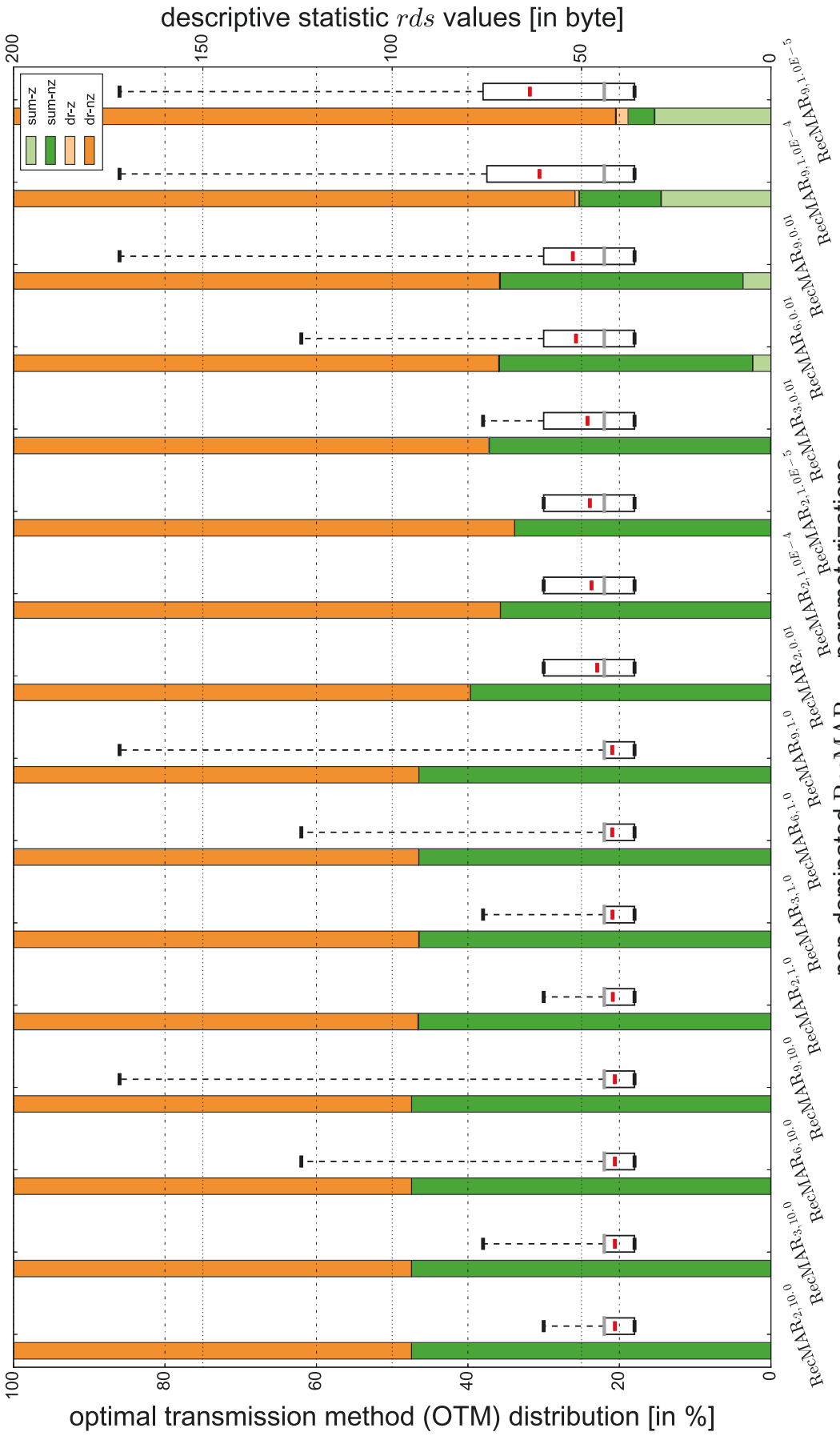


Fig. 35: Exemplary depiction of the data aggregated into a single column of Table 4. The figure depicts the Skyline parameterizations of $\mathrm{RecMAR}_{k,sl}$. For each parameterization, the shares of how the resource descriptions are transmitted in the network (bars on the respective left) and the descriptive statistic values for the resulting resource description sizes (boxplots on the respective right) are depicted.

dominated parameterizations'-row). Regarding these, it shows that if an approach has only a small number (1 or 2) of parameters *determining the spatial accuracy* of the resource summaries[130], the share of Skyline parameterizations is generally high (i.e. most of the tested parameterizations are part of the approach's Skyline). The more of these accuracy-determining parameters exist (and therefore the greater the total number of tested parameterizations), the smaller the share of an approach's parameterizations which are part of the corresponding Skyline.[131] For example, for $\text{RecMAR}_{k,sl}$ (two accuracy-determining parameters), the share of Skyline parameterizations is 66.7% while for $\text{MARQT}^{c,a}_{k,sl}$ (four accuracy-determining parameters), it is only 20.1%.

The more parameters, the more it also proves particularly beneficial to raise *specific* parameters or parameter combinations—potentially excluding large chunks of other parameter combinations from being part of the Skyline.

For e.g. $\text{KDMAR}^{b,k}_{n}$, out of 34 parameterizations forging the Skyline, only 3 feature $b = 3$, 13 feature $b = 4$, whereas all of the 18 parameterizations with $b = 6$ are present. For the parameters n and k, the single tested values are much more evenly represented in the Skyline. Therefore, for $\text{KDMAR}^{b,k}_{n}$, it seems that parameter b (specifying the accuracy of the cell-interior rectangles) is especially important for the performance.

In contrast, for e.g. KD_n and $\text{UFS}_{n,cc}$, there is only one accuracy-determining parameter (n). Thus, there is no 'competition' between different accuracy-determining parameters. Expectably, more subspaces n result in more selective and more storage-space-intensive resource descriptions and therefore, all KD_n and $\text{UFS}_{n,cc}$ parameterizations are part of the corresponding Skylines. Another reason that a large share of Skyline parameterizations might result for an approach is the dominance of a single parameter with regard to both the description accuracy and the resulting *rds* values. For e.g. $\text{DFS}^{b}_{n,cc}$, the variation of parameter n (number of subspaces) completely dominates the variation of parameter b (number of bits for the quantized r^{out} distances of occupied cells). For example, 35.6 B *rds* result for $\text{DFS}^{3}_{32,cc}$. For $\text{DFS}^{6}_{32,cc}$, the *rds* increases to only 35.9 B while for $\text{DFS}^{3}_{64,cc}$, the *rds* increases to 37.1 B. As a consequence of this dominance, the share of Skyline parameterizations of $\text{DFS}^{b}_{n,cc}$ is 100%.

[130] Note that the cc parameter of $\text{UFS}_{n,cc}$ and $\text{DFS}^{b}_{n,cc}$ is a an example for a parameter *not* determining the spatial accuracy of resource summaries (as it only influences the ranking process of the metric-domain-like ranker but not the summary creation).

[131] Note that for approaches with a low number of accuracy-determining parameters, the total number of parameterizations is rather low in comparison to approaches featuring more parameters determining the accuracy of the resource description. For example, for $\text{RecMAR}_{k,sl}$, there are 24 different parameterizations, while for $\text{MARQT}^{c,a}_{k,sl}$, there are 432 tested parameterizations.

Analysis of the Aggregated *rds* Values for the Different Approaches. In the following, we take a look at the *rds*-related key figures of Table 4 (rows 3 to 8, i.e. the 'avg rds'-row to the 'max rds'-row).
The 'avg rds'-column of Table 4 once again shows that the Skylines of the approaches start and end at fairly different *rds* values. The starting points range from 30.5 B (KD_{32}) to 41.6 B ($MBRQT^{16,1.0}_{2,10.0}$), whereas the end points range from 46.1 B ($QTMBR^{8}_{1024,0.001}$) to 63.6 B ($RecMAR_{9,1.0E-5}$). It highlights the difficulty of determining and setting the parameters of the different approaches such that the corresponding Skylines cover similar *rds* ranges. On the other hand, the comparability of the approaches is not really affected by the differing Skyline ranges. This is because all Skylines asymptotically approach the baseline in the final stage (where high additional *rds* investments only result in marginal *rfc* improvements) while in the early stage, all Skylines rise steeply (since low additional *rds* investments result in big *rfc* improvements)—even though the absolute flatness respectively absolute steepness of the Skyline vary from one approach to another. Furthermore, towards the starting points of the Skylines, for low *rds* values and all approaches except the full-precision-MBR-based approaches (MBR approach, $RecMAR_{k,sl}$, $MBRQT^{c,a}$, and $MARQT^{c,a}_{k,sl}$), selectivity at some point becomes rapidly (much) worse than MBR selectivity—and such a selectivity is of no real interest. Hence, the comparability of the approaches is not restricted by the diverging Skyline ranges. In principle, the Skyline of each parameterizable approach could be extended to the left as desired by setting the parameters accordingly—an infinite scaling is hampered by the arising computational costs, though. Late starting points occur for the respective Skylines of $RecMAR_{k,sl}$, $MBRQT^{c,a}$, and $MARQT^{c,a}_{k,sl}$ since they are storage-space-lower-bounded by their full-precision rectangle(s) (i.e. they cannot have lower *rds* values than the MBR approach). For all the other parameterizable approaches, the Skylines start below 36.0 B *rds*.
From the 'min rds'-row, it is evident that for most approaches, it is possible to create summaries which consume less storage space than the coordinates of a single data point (which account for 36 B as a resource description or 8 B raw payload). For all approaches except again those being based on full-precision MBRs, minimum resource description sizes below 36 B occur. For some approaches, it is only achievable for 'low' (i.e. non-storage-space-intensive) parameterizations (which typically result in spatially coarse resource summaries), though. For e.g. $QTMBR^{b}_{c,a}$, the minimum resource description sizes range from *minimally* 33 B (for $QTMBR^{3}_{16,1.0}$) to *maximally* 36 B.[132] Hence, for 'higher' (i.e more storage-space-intensive) parameterizations of $QTMBR^{b}_{c,a}$ (such as $QTMBR^{6}_{1024,0.1}$), it is not possible to build summaries which require less storage space

[132]The *minimal* (*maximal*) minimum resource description size is the smallest (biggest) minimum resource description size occuring for any Skyline parameterization of a specific approach.

than a single data point. For local-space-partitioning-based approaches and GridQT$_r^{c,a}$, it is dependent on the parameterization whether summary sizes below data-point-size occur. In contrast, for KDMBR$_n^b$, the minimum resource description sizes range from *minimally* 30 B to *maximally* 32 B, i.e. even for its highest Skyline parameterization (KDMBR$_{8192}^8$), summary sizes below data-point-size occur. In general, for all approaches based on global space partitioning (i.e. Voronoi- and k-d-based pure space partitioning and hybrid approaches), minimum summary sizes below data-point-size can occur for *all* of their Skyline parameterizations. The reason for this is that the bit vectors of the resource descriptions are generally clipped in a byte-aligned fashion after the rightmost bit which is set to '1'. If for e.g. KD$_{32}$, a resource contains its single data point in the cell with $ID = 0$ (which is represented by the leftmost bit in the corresponding summary bit vector), the raw summary data is clipped after the first byte (i.e. 8 bit) of its actually 32 bit. The clipped information is sufficient since by knowledge of the parameterization, it can be reconstructed that only the cell with $ID = 0$ contains data points while all the other cells are empty. The clipping is not 'reliable' for the approaches based on global space partitioning and involves a lot of randomness: It works well for resources which administer their data points in subspaces which are represented early in the summary bit vector (i.e. subspaces with a low ID) but not when the subspaces are represented far back in the bit vector. Generally, the byte-aligned bit vector clipping is attempted for all approaches but is never applicable for MBR, RecMAR$_{k,sl}$, QT$_{c,a}$, QTMBR$_{c,a}^b$, QTMAR$_{c,a}^{b,k}$, MBRQT^{c,a}, and MARQT$_{k,sl}^{c,a}$.
The 'max rds'-row reveals that the maximum scaling of the resource description sizes varies greatly for the different approaches. For data partitioning approaches (MBR approach, RecMAR$_{k,sl}$), the maximum resource description sizes are always kept within certain, specifiable bounds—even when utilizing the respective most storage-space-intensive parameterizations—since they are storage-space-upper-bounded by the number of bounding volumes utilized. For e.g. RecMAR$_{k,sl}$, no resource description sizes above 172 B occur since the maximum amount of bounding MARs is $k = 9$ ($27\,B + 1\,B + (9 \cdot 16\,B) = 172\,B$). For pure global space partitioning approaches, the *occuring* maximum resource description size is not predictable due to the bit vector clipping. For e.g. KD$_{16384}$, the occuring maximum bit vector size is 1,617 B while for UFS$_{16384,cc}$, it is 1,376 B—the *theoretical* maximum resource description size is $27\,B + 1\,B + (16,384/8)\,B = 2,076\,B$ for both. Nevertheless, the maximum resource description size which *theoretically* can occur is still easily predictable for pure global space partitioning approaches. This prediciton is at least not trivially possible for QT$_{c,a}$ since the quadtree structure can vary greatly and additionally, two different linear quadtree encoding schemes are available for its codification. Interestingly, the *maximally* occuring maximum resource description size for the QT$_{c,a}$ approach is 2,105 B which is about in the same range as the theoretical maximum sizes for

both global space partitioning approaches. For the hybrid approaches, it is as follows: For hybrid approaches utilizing quadtree structures, a maximum limit is difficult to assess. For the others (KDMBR_n^b, $\text{KDMAR}_n^{b,k}$, and $\text{DFS}_{n,cc}^b$), it is easily specifiable. For e.g. KDMBR_{8192}^6, the theoretical maximum is $27\,B + 1\,B + (((n = 8192) \cdot (4 \cdot (b = 6)))/8)\,B = 24,604\,B$. Its *maximally* occurring maximum resource description accounts for 'only' 8,222 B, though. In general, in comparison to the pure global space partitioning approaches, far larger maximum resource description sizes can occur for hybrid approaches—both with respect to the *minimal* maximum resource description sizes as well as the *maximal* maximum resource description sizes.[133] For e.g. $\text{KDQT}_n^{c,a}$, the range for the maximum resource description sizes is from *minimally* 160 B ($\text{KDQT}_{32}^{16,1.0}$) to *maximally* 26,189 B ($\text{KDQT}_{8192}^{512,1.0E-8}$). Hence, even for the lowest Skyline parameterization of $\text{KDQT}_n^{c,a}$ ($\text{KDQT}_{16}^{16,1.0}$), at least one resource is described by 160 B. In contrast, for the lowest Skyline parameterizations of e.g. KD_n (KD_{32}) or $\text{UFS}_{n,cc}$ ($\text{UFS}_{32,cc}$), it can be limited to 33 B (meaning for these, there are parameterizations for which *exclusively* summaries are transmitted for the entire resource set since even the biggest summary of a resource requires less storage space than the coordinates of a data point).[134] For the hybrid approaches, especially the k-d-based approaches (KDMBR_n^b, $\text{KDMAR}_n^{b,k}$, and $\text{KDQT}_n^{c,a}$) as well as $\text{GridQT}_r^{c,a}$ stand out with very large *minimal* and *maximal* maximum resource description sizes. For scenarios requiring strictly limitable maximum resource description sizes, the harsh scaling of these approaches in extreme cases (i.e. for resources whose data points are spatially very spread) must be kept in mind.

In our scenario, in the end, only the average resource description sizes are of interest. Therefore, the adaptability to different circumstances as well as the scalability of an approach are beneficial properties: if possible, use little storage space to describe a resource with a certain degree of spatial accuracy but spend a lot of storage space if required (i.e. for spatially spread resources). This adaptive scalability is particularly noticeable for the hybrid approaches. For e.g. $\text{KDQT}_{8192}^{512,1.0E-8}$, the resources of the `T1` collection

[133] The *minimal* (*maximal*) maximum resource description size is the smallest (biggest) occuring maximum resource description size for any Skyline parameterization of a specific approach.

[134] The 33 B maximum resource description sizes for KD_{32} and $\text{UFS}_{32,cc}$ occur due to the specifics of the Java classes used in our implementation. Generally, one would expect 32 B to be the maximum resource description size for both approaches: 27 B serialization overhead plus 1 B metadata plus 32 bit or 4 B for the occupancy information of the 32 cells of the global space partition. The class `java.util.BitSet` is used to represent bit vectors. However, since it does not support sign extension, it must be guaranteed that the rightmost bit is not a '1' when a `BitSet` instance is converted to a byte array (`byte[]`). Hence, for e.g. $\text{UFS}_{32,cc}$ summaries for which the 32th bit is set to '1', a byte array of length 5 (instead of length 4) results. In general, this possible extension by 1 B applies for all approaches where the summaries are represented by a `BitSet` prior to the final conversion into a byte array (i.e. all approaches except MBR and $\text{RecMAR}_{k,sl}$).

are described with between 35 B and 26,189 B, illustrating the adaptiveness of $\text{KDQT}_n^{c,a}$.
By nature, the 'max rds'-values are outliers and do not necessarily coincide with the average resource description size values ('avg rds'-row). In this respect, the 25%-quantile ('25%-quant. rds'-row), median ('median rds'-row), 75%-quantile ('75%-quant. rds'-row), and to a smaller extent even the minimum resource description size values ('min rds'-row) are more representative and correlate much better with the 'avg rds'-values. Notably, for $\text{QTMBR}_{c,a}^{b}$ and $\text{QTMAR}_{c,a}^{b,k}$, the *maximal* median resource description size value is below the resource description size of two data points (44 B). Hence, even with the highest parameterizations of $\text{QTMBR}_{c,a}^{b}$ and $\text{QTMAR}_{c,a}^{b,k}$, at least 50% of the resource descriptions only require 40 B ($\text{QTMBR}_{c,a}^{b}$) respectively 39 B ($\text{QTMAR}_{c,a}^{b,k}$) or even less. This also explains why the Skylines of both approaches end at relatively low *rds* values ($\text{QTMBR}_{c,a}^{b}$: 46.1 B, $\text{QTMAR}_{c,a}^{b,k}$: 47.3 B)—the majority of resources can be described at the targeted degree of spatial accuracy with low storage space requirements (remember that the construction of the basic quadtrees is accuracy-limited by parameter a).

Analysis of the Aggregated Transmission Method Values for the Different Approaches. In the following, it is evaluated how the resource description data is preferably transmitted for the different approaches, i.e. the transmission-method-related key figures from Table 4 are analyzed.
For the transmission method values in rows 9 to 14, it has to be considered that the big numbers at the top of the single cells are mean values averaged over all the Skyline parameterizations of a specific approach. Consequently, these values provide only a rough overview and are not suited for a detailed analysis since the values can be biased depending on whether the Skyline parameterizations preferably occur in the early or the final stage of the respective Skyline.
As an example, consider $\text{QTMBR}_{c,a}^{b}$ and $\text{QTMAR}_{c,a}^{b,k}$. In general, the non-aggregated transmission method values for the $\text{QTMBR}_{c,a}^{b}$ and the $\text{QTMAR}_{c,a}^{b,k}$ Skyline parameterizations[135] are very similar when similar *rds* values result. Also, the ranges of the *minimally* (small, bracketed values in the middle of the cells from row 9 to 14) and *maximally* (small, bracketed values at the bottom) occurring shares are almost identical for both approaches. Consider the 'dr-nz'-row (i.e. the share of resources transmitting non-zipped data point coordinates as resource descriptions) as an example: for $\text{QTMBR}_{c,a}^{b}$, the range is between minimally 0% and maximally 45.8%. For $\text{QTMAR}_{c,a}^{b,k}$, it is between 0% and 45.7%. Nevertheless, the averaged 'dr-nz'-value for the $\text{QTMBR}_{c,a}^{b}$ Skyline parameterizations is 30.5% while for the $\text{QTMAR}_{c,a}^{b,k}$ Skyline parameterizations, it is 25.4%—even though for

[135]Note that these non-aggregated, single values are neither depicted in Table 4 nor anywhere else in the work due to reasons of space.

similar resulting *rds* values, the respective 'dr-nz'-shares are always much closer to each other than these given average numbers. For example, for $\text{QTMBR}^{6}_{1024,0.001}$ (*rds*: 44.68 B), the 'dr-nz'-option is selected for 45.61% of the resources while for $\text{QTMAR}^{4,4}_{1024,0.001}$ (*rds*: 44.65 B), it is selected for 45.23%. The reason for this discrepancy is that for $\text{QTMAR}^{b,k}_{c,a}$, there are much more parameterizations located at the early stage of the Skyline—for which the 'dr-nz'-values are typically low. Consequently, the average 'dr-nz'-value for $\text{QTMAR}^{b,k}_{c,a}$ is significantly lower than for $\text{QTMBR}^{b}_{c,a}$. Thus, the expressiveness of the averaged transmission method values is not sufficient for a detailed analysis, and at least the occurring *minimal* and *maximal* shares of the transmission methods also have to be considered when assessing these rows in detail.

Overall, the numbers show that by tendency, a greater share of resource descriptions is 'non-zipped', i.e. the resource description data is often not compressible. In general, the averaged, the maximal, and also the minimal values for almost all the '-z'-rows are smaller than for their '-nz'-counterparts.[136] Thus, for many approaches, it can as well occur that for certain of their parameterizations, the zipped transmission option is chosen more often. For e.g. KD_{16384}, the zipped summary ('sum-z') share is 33.8% while the non-zipped summary ('sum-nz') share is only 0.6%. As a general rule, the higher the parameterization, the more data is contained in a resource description and the more often the zipping option is chosen (and vice versa).

Pure data partitioning approaches and all quadtree-*utilizing* approaches except $\text{GridQT}^{c,a}_{r}$ and $\text{KDQT}^{c,a}_{n}$ show a low tendency of transmitting zipped *summaries*. For the former, this was to be expected since the summaries only contain few float values (respectively their binary representation) which offer little potential for entropy compression. For the latter, the linear quadtree codes typically possess a high entropy and are therefore usually also not compressible. In fact, the zipped summary options (i.e. 'lq-z'- and 'cblq-z'-options combined, which for e.g. $\text{MBRQT}^{c,a}$ result in 0.043%) are chosen even less often for the quadtree-utilizing approaches except $\text{GridQT}^{c,a}_{r}$ and $\text{KDQT}^{c,a}_{n}$ than the zipped summary option ('sum-z', 2.3%) is chosen for $\text{RecMAR}_{k,sl}$. This means that for $\text{RecMAR}_{k,sl}$, more resources tend to transmit zipped summaries. Pure space partitioning approaches as well as hybrid approaches being based on global space partitioning tend to have a higher share of zipped summaries. The binary cell occupancy information of the summaries is more often compressible, especially when there are few (many) populated cells such that long zero (one) runs occur in the bit vector. Generally, the refinement of occupied cells of a global space partition with quantized rectangles or quantized cluster balls raises the entropy of the corresponding summary data. Therefore, the share of zipped summaries (values in the 'sum-z/lq-z'-row) for e.g. KDMBR^{b}_{n} and

[136] The only exception are the 'cblq-z'-/'cblq-nz'-rows (4.3% versus 3.3%) for $\text{GridQT}^{c,a,}_{r}$.

$\text{KDMAR}_n^{b,k}$ is tendencially lower than for KD_n. The same goes for $\text{DFS}_{n,cc}^b$ in comparison to $\text{UFS}_{n,cc}$. With respect to the shares of zipped summaries, somewhere in between the quadtree-*based* approaches on the one hand and the global-space-partitioning-based approaches on the other hand are $\text{GridQT}_r^{c,a}$ and $\text{KDQT}_n^{c,a}$. Both are based on global space partitioning but utilize quadtrees for refinement. Their average zipped summary option shares are about in the middle between the corresponding values for the other quadtree-utilizing approaches and the global space partitioning approaches.[137] Hence, it can be suspected that in the $\text{GridQT}_r^{c,a}$ and $\text{KDQT}_n^{c,a}$ summaries, the quadtree data has a great weight, i.e. possibly rather few, coarse global subspaces are built which are then refined in greater detail by the subspace-interior quadtrees. On the other hand, the average shares of directly represented resources ('dr-nz'-row) for $\text{GridQT}_r^{c,a}$ and $\text{KDQT}_n^{c,a}$ are much closer to the shares of e.g. KDMBR_n^b and $\text{KDMAR}_n^{b,k}$ than of $\text{QTMBR}_{c,a}^b$. Hence, the quadtree-typical property of building very storage-space-efficient summaries for many resources of the `T1` collection does not seem to be preserved by $\text{GridQT}_r^{c,a}$ and $\text{KDQT}_n^{c,a}$. We return to this in later parts of the evaluation.

For the quadtree-*utilizing* approaches, it is evident that the LQ code is chosen more often than the CBLQ code for encoding the quadtree data: on average, the LQ code is chosen by between 40.4% ($\text{MARQT}_{k,sl}^{c,a}$) to 80.2% ($\text{QTMAR}_{c,a}^{b,k}$)[138] of the resources whereas the CBLQ code is chosen by between 2.0% ($\text{QTMAR}_{c,a}^{b,k}$) and 9.7% ($\text{QT}_{c,a}$)[139].

The share of the 'dr-z'-option (i.e. transmitting zipped coordinates of the data points as resource descriptions) is generally very low. This is as expected since usually, only for resources administering few data points, the direct representation is chosen over a summary as the resource's description. Hence, the direct representations are typically very few float values—which can rarely be entropy-compressed.

The 'dr-nz'-row shows that the share of the non-zipped transmission of data point coordinates is generally high—which is no surprise due to the `T1` collection's large amount of resources administering only one or two data points (see Table 2 on page 114). In a comparison of the approaches, for the quadtree-*based* approaches ($\text{QT}_{c,a}$, $\text{QTMBR}_{c,a}^b$, $\text{QTMAR}_{c,a}^{b,k}$), the 'dr-nz'-shares are lowest for both the averaged shares (big numbers at the top of the cells) as well as the *maximally* occurring shares (small, bracketed numbers at the bottom of the cells). For the rectangle-based approaches (MBR

[137] The average share of resources transmitting zipped summary data is 5.8% ('sum-z'-/'lq-z'-row) $+ 4.3\%$ ('cblq-z'-row) $= 10.1\%$ for $\text{GridQT}_r^{c,a}$, and $4.2\% + 2.5\% = 6.7\%$ for $\text{KDQT}_n^{c,a}$. For both $\text{QTMBR}_{c,a}^b$ and $\text{QTMAR}_{c,a}^{b,k}$, basically 0% of the resources transmit zipped summaries. For KDMBR_n^b ($\text{KDMAR}_n^{b,k}$), it is 18.3% (16.5%).

[138] The given values result from summing up the 'lq-z'- and the 'lq-nz'-values of the corresponding columns of Table 4.

[139] The given values result from adding up the 'cblq-z'- and the 'cblq-nz'-values of the corresponding columns of Table 4.

approach, RecMAR$_{k,sl}$, MBRQT^{c,a}, MARQT$^{c,a}_{k,sl}$), invariably at least 52.3% of the resources chose the 'dr-nz'-option because 52.3% of the resources administer only one or two data points (also see Table 2): Since a single rectangle already requires two data points for its specification (lower left and upper right corner), summaries are never chosen in these cases. For all other approaches except DFS$^{b}_{n,cc}$[140], it is possible that 0% of the resources chose the 'dr-nz'-option when low parameterizations are utilized ($\rightarrow$ the *minimal* 'dr-nz'-value—small, bracketed number in the middle of the respective cells—is 0% for these approaches). Notably, the *maximally* occuring share of 'dr-nz'-choosing resources is greater 60% for all approaches except QTMBR$^{b}_{c,a}$ (45.8%) and QTMAR$^{b,k}_{c,a}$ (45.7%). In total, the latter two can be considered to be the approaches with the greatest tendency of describing resources with summaries—which was assumed to be rather disadvantageous in section 6.4.1. Nevertheless, the Skylines of both QTMBR$^{b}_{c,a}$ and QTMAR$^{b,k}_{c,a}$ are certainly among the best for the T1 collection.

In general, for high parameterizations (and therefore high resulting *rds* values), the share of directly represented resources ('dr-z'- + 'dr-nz'-values) is very high (up to 81.1% for RecMAR$_{9,1.0E-5}$). This might be the explanation for the strong convergence of the approaches for high *rds* values: The ranking algorithm operates with very accurate summaries for the few summary-represented resources and additionally, the exact coordinate information is known for all these directly represented resources—ultimately resulting in a very clear distinction between relevant and irrelevant resources. For very low *rds* values—i.e. when the resources are almost exclusively described by (very coarse) summaries—there are tremendous differences between the single approaches. The Skylines reveal that pure global space partitioning approaches are especially unsuitable for very low *rds* values in the given environment. Pure global space partitioning approaches can obviously not succeed for the T1 collection and low *rds* values: For example, for 32 global subspaces without further refinement, even if each of the 2,491,785 summary-described resources would occupy only one

[140] DFS$^{b}_{n,cc}$ is a bit of a special case since its summaries also encode the respective resource's *maximal* r^{out} value to quantize against as a single precision floating-point number—which accounts for 4 B. With $n = 32$, 4 B must additionally be spent for the binary cell occupancy information. Since at least one cell must be occupied with a data point, at least b bits have to be spent for the quantized r^{out} radius of this cell. In total, in its lowest parameterization ($n = 32$, $b = 3$), at least $4\,B + 4\,B + 3$ *bit* of raw data must be spent—which is more than the 8 B for a single data point. Nevertheless, the 'dr-nz'-share of the *technique* DFS$^{3}_{32,cc}$ is 27.2% (which constitutes the *minimal* occuring 'dr-nz'-value of the *approach* DFS$^{b}_{n,cc}$)—even though 36.8% of the resources administer only one data point. The reason for this is the bit vector clipping. Therefore, if for DFS$^{3}_{32,cc}$, a resource contains its data points in cells with a low enough ID, the bit vector can be clipped before the theoretical minimum of $8\,B + 3$ *bit* of raw data are reached. This is the reason why not all of the resources which administer one data point (36.8% in total) transmit the coordinates of their data point.

subspace—which by no means is the case—about 78,000 resources would share their subspace with each other on average.
Still, let us emphasize once again that the high level of aggregation of the transmission-method-related results allows only very general assertions—which can be contradicted by single parameterizations of the single approaches. For e.g. $\text{KDQT}_{8192}^{512,1.0E-8}$, there are ~22.8% zipped summaries and ~0.5% non-zipped summaries. This is totally contradictory to the aggregated values displayed in Table 4 (where for $\text{KDQT}_{n}^{c,a}$, zipped summaries generally have much lower shares than non-zipped summaries).

7.1.3. COMPARISON OF THE APPROACHES AT 'MBRSIZE'. In the following, we assess the results for *concrete parameterizations* of the approaches. In Table 5, a comparison of the 14 approaches is shown. Each approach is represented by the one Skyline parameterization whose resulting *rds* value is closest to the MBR approach's *rds* value.[141] The grey-shaded rows highlight the two primary key figures we assess with regard to the quality of a resource description approach, the *rds* values (i.e. the resulting average resource description sizes, in row 2) and the corresponding *rfc* values (i.e. the resulting average resource fractions contacted for the 50 queries, in row 8). Both are accompanied by the corresponding descriptive statistic values: From rows 3 to 7 (i.e. the 'min rds'-row to the 'max rds'-row), the descriptive statistic of the resource description sizes is outlined. From rows 9 to 13 (i.e. the 'min rfc'-row to the 'max rfc'-row), the descriptive statistic of the resource fractions contacted is outlined. The rows 14 to 19 (i.e. the 'sum-z/lq-z'-row to the 'cblq-nz'-row) outline the transmission method shares.[142]
The top bracket of techniques at 'MBRsize' consists exclusively of quadtree-*utilizing* approaches: $\text{QT}_{8192,0.001}$, $\text{GridQT}_{4}^{512,0.001}$, $\text{KDQT}_{32}^{64,1.0E-5}$, $\text{QTMBR}_{512,0.05}^{6}$, and $\text{QTMAR}_{256,0.1}^{6,2}$ are clearly ahead of the rest with regard to the *rfc* value ($\rightarrow$ 'avg rfc'-row; also evident from Figure 34). For $\text{GridQT}_{4}^{512,0.001}$ and $\text{KDQT}_{32}^{64,1.0E-5}$, it is remarkable that in either case, only 32 global subspaces are built but each occupied subspace is refined by up to 64 ($\text{KDQT}_{32}^{64,1.0E-5}$) or even 512 ($\text{GridQT}_{4}^{512,0.001}$) quadtree cells. Of course, occupied global subspaces are only very rarely (if ever) refined by such an amount of quadtree cells since the quadtree construction can be ended prematurely by the threshold surface area parameter a. Nevertheless, the strong preference for the quadtree component of the summary in these parameterizations is noteworthy—all the more since in addition, the quadtree

[141] Note that in the following, the Skyline parameterization of an approach whose *rds* value is closest to the *rds* value resulting for the MBR approach is entitled to be the Skyline parameterization at 'MBRsize' of this approach.

[142] Thus, the structure of Table 5 is very similar to the structure of Table 4 (with the addition of the descriptive statistic for the resource fractions contacted). Nevertheless, it is important to note that for Table 4, the values are aggregated over the whole set of Skyline parameterizations for an approach while for Table 5, values for a single, specific Skyline parameterization of an approach are depicted.

Table 5: Listing comparing the different approaches in their parameterizations at 'MBRsize'. For each approach, the descriptive statistic values for the resulting *rds* values (rows 2 to 7) and the resulting *rfc* values (rows 8 to 13) are given alongside the shares of the resource description transmission methods (rows 14 to 19).

technique↑ key figure →	**MBR**	**RecMAR**$_{2,10.0}$	**KD**$_{256}$	**UFS**$_{256,cc}$	**QT**$_{8192,0.001}$	**KDMBR**$^{6}_{256}$	**KDMAR**$^{4,2}_{256}$	**DFS**$^{6}_{128,cc}$	**GridQT**$^{512,0.001}_{4}$	**KDQT**$^{64,1.0E-5}_{32}$	**QTMBR**$^{6}_{512,0.05}$	**QTMAR**$^{6,2}_{256,0.1}$	**MBRQT**$^{16,1.0}$	**MARQT**$^{16,1.0}_{2,10.0}$
avg *rds* [in byte]	41.06	41.19	40.93	41.24	39.95	42.96	42.72	40.32	40.64	41.21	41.04	41.01	41.53	41.64
min *rds* [in byte]	36	36	29	29	34	31	31	33	34	32	36	36	36	36
25%-quant. *rds* [in byte]	36	36	36	36	36	36	36	36	36	36	36	36	36	36
median *rds* [in byte]	44	44	37	38	37	40	39	39	37	38	38	38	44	44
75%-quant. *rds* [in byte]	44	44	47	49	41	51	50	44	42	45	45	44	45	45
max *rds* [in byte]	44	60	61	61	1,948	565	717	123	2,235	486	1,154	1,303	50	71
avg *rfc* [in ‰]	21.13	10.35	43.41	38.62	0.92	2.97	4.97	34.33	0.72	0.65	0.39	0.31	7.62	5.50
min *rfc* [in ‰]	1.41	0.30	6.22	2.25	0.04	0.09	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
25%-quant. *rfc* [in ‰]	14.61	4.58	27.23	20.38	0.37	1.03	1.23	11.75	0.28	0.29	0.16	0.15	2.64	1.54
median *rfc* [in ‰]	20.55	9.25	38.39	34.69	0.73	1.56	2.58	30.13	0.46	0.50	0.25	0.23	6.03	3.73
75%-quant. *rfc* [in ‰]	30.23	14.40	49.24	53.41	1.12	3.02	3.93	47.50	0.73	0.99	0.51	0.44	9.74	7.37
max *rfc* [in ‰]	44.39	31.89	202.8	154.4	4.36	24.36	35.06	175.0	3.37	2.69	2.07	1.80	31.01	26.30
sum-z/lq-z [in %]	0	≪0.01	12.8	16.8	0	11.8	12.0	0	≪0.01	0	0	0	0	0
sum-nz/lq-nz [in %]	47.7	47.5	44.1	34.3	50.3	40.0	41.0	55.4	53.9	55.9	52.4	56.1	47.1	46.9
dr-z [in %]	0	≪0.01	0	0.04	0	≪0.01	≪0.01	0	0	0	≪0.01	0	0	≪0.01
dr-nz [in %]	52.3	52.6	43.2	48.8	36.2	48.2	47.0	44.6	36.8	34.2	42.0	40.5	52.3	52.6
cblq-z [in %]	-	-	-	-	0.01	-	-	-	0.01	≪0.01	≪0.01	≪0.01	0	0
cblq-nz [in %]	-	-	-	-	13.5	-	-	-	9.3	9.9	5.7	3.4	0.6	0.6

[143]Note that within the top bracket, it is evident that for both QTMBR$^{6}_{512,0.05}$ and QTMAR$^{6,2}_{256,0.1}$, no summaries below a size of 36 B can be built (see 'min rds'-column of Table 5). Hence, resources administering only one data point transmit the data point's coordinates instead of a QTMBR$^{6}_{512,0.05}$ respectively QTMAR$^{6,2}_{256,0.1}$ summary as a resource description. For QT$_{8192,0.001}$, GridQT$^{512,0.001}_{4}$, and KDQT$^{64,1.0E-5}_{32}$, minimum resource description sizes below 36 B exist—so at least some 'one-data-point-resources' transmit summaries. As a consequence, the 'dr-nz'-value is between 4% and 8% higher for QTMBR$^{6}_{512,0.05}$ respectively QTMAR$^{6,2}_{256,0.1}$ in comparison to the other approaches of the top bracket.

component is also the more expensive one storage-wise (as the cell occupancy of the global space partition is simply depicted by binary information and easily compressible—in contrast to the quadtree data). For the top bracket approaches, two additional aspects stand out:

— The shares of directly represented resources for the top bracket approaches—which are between 34.2% ($\text{KDQT}_{32}^{64,1.0E-5}$) and 42.0% ($\text{QTMBR}_{512,0.05}^{6}$)—are lower than for all the other approaches—being between 43.2% (KD_{256}) and 52.6% ($\text{RecMAR}_{2,10.0}$ and $\text{MARQT}_{2,10.0}^{16,1.0}$).[143]
— The summaries of the top bracket approaches are almost never zipped—equally applying for both LQ-encoded summaries as well as CBLQ-encoded summaries:
  — Solely 13 resources transmit a zipped LQ-encoded summary for $\text{GridQT}_{4}^{512,0.001}$—which is the only top bracket approach that ever transmits zipped LQ-encoded summaries.
  — For all the top bracket approaches, the total share of zipped CBLQ-encoded summaries is 0.01% or much lower.

  Hence, the total share of zipped summaries is 0.01% or lower for all the top bracket approaches. In contrast, for KD_{256}, $\text{UFS}_{256,cc}$, KDMBR_{256}^{6}, and $\text{KDMAR}_{256}^{4,2}$, zipped summaries have a share of at least 10%.

Generally, utilizing quadtree structures (no matter if as the description base or as a refinement) seems to be unconquerable for the T1 collection when aiming at 'MBRsize'.[144] The (much) better selectivity of the top bracket approaches despite the smaller share of directly represented resources and the virtually non-existent zipping underlines this fact. Additionally, the listed 75%-quantile resource description sizes ($\rightarrow$ values in the '75%-quant. rds'-row) reveal that for the top bracket approaches (at least by tendency), the majority of resource descriptions require similar or even less storage space compared to the resource descriptions of the other approaches: The values are between 41 B ($\text{QT}_{8192,0.001}$) and 45 B ($\text{KDQT}_{32}^{64,1.0E-5}$ and $\text{QTMBR}_{512,0.05}^{6}$) for the top bracket approaches while for the remaining approaches, it is between 44 B (MBR approach, $\text{RecMAR}_{2,10.0}$, and $\text{DFS}_{128,cc}^{6}$) and 51 B (KDMBR_{256}^{6}). Nevertheless, the maximum resource description sizes (values in the 'max rds'-row of Table 5) for particularly $\text{GridQT}_{4}^{512,0.001}$ and $\text{QT}_{8192,0.001}$—but also for $\text{QTMBR}_{512,0.05}^{6}$ and $\text{QTMAR}_{256,0.1}^{6,2}$—can be very large for the given parameterizations. This shows that quadtree-utilizing approaches are much more adaptive than the other approaches for the targeted 'MBRsize'.
With regard to the selectivity, the approaches can be categorized into three groups:

[144]Note that for both $\text{MBRQT}^{16,1.0}$ and $\text{MARQT}_{2,10.0}^{16,1.0}$, the quadtree structures are of very limited impact here: if any, only very small quadtrees are built, and the overwhelming part of the summaries' storage space is devoted to encoding the basic full-precision rectangles.

— The top bracket approaches offer at least 20 times better selectivity than the MBR approach.
— The middle bracket approaches (RecMAR$_{2,10.0}$, KDMBR$^{6}_{256}$, KDMAR$^{4,2}_{256}$, MBRQT$^{16,1.0}$, and MARQT$^{16,1.0}_{2,10.0}$) offer an 2 to 10 times better selectivity than the MBR approach. Notably, the suitability of utilizing quadtree structures shows once again when considering MBRQT$^{16,1.0}$ and MARQT$^{16,1.0}_{2,10.0}$: The selectivity is improved by a factor of ~64% (~74%) by spending only 0.49 B *rds* (0.58 B *rds*) extra.
— The bottom bracket approaches (KD$_{256}$, UFS$_{256,cc}$, and DFS$^{6}_{128,cc}$) are ~1.6 to ~2.1 times worse than the MBR approach.

The 'avg rfc'-values correspond to between ~77 contacted resources (for QTMAR$^{6,2}_{256,0.1}$) and ~10,817 contacted resources (for KD$_{256}$), revealing tremendous differences in the suitability of the different approaches. Generally, the 'avg rfc'-values correlate strongly with the descriptive statistic *rfc* values (rows 9 to 13 in Table 5). If a ranking of approaches for the values in each of these rows is built, each ranking is almost the same as the ranking of approaches for the 'avg rfc'-row values. Hence, it can be claimed that for none of the approaches, a distorted 'avg rfc'-value resulted because of outliers—the general performance of the approaches is very well represented by the 'avg rfc'-values.

With regard to the linear quadtree encoding schemes, the general presumption is that the LQ code (depicting a separate code for each black node) is better suited for quadtrees with few black nodes—irrespective of the quadtrees' depths. In contrast, the CBLQ code (depicting the quadtree structure as a whole) is better suited for quadtrees with many black nodes. This is because with many black nodes, the LQ code is very redundant as the first few literals of the black nodes' codes are often the same. See Figure 36 for an illustrative example quadtree. The quadtree there has only one black cell. Assuming that the maximum depth of a quadtree is greater than 2, the corresponding LQ code would be 11X whereas the corresponding CBLQ code is 0300-0100. Since a literal in the LQ code requires 3 bits and a literal in the CBLQ code requires 2 bits, the codes would account for 9 bits (LQ code) respectively 16 bits (CBLQ code) in total. For any additional level introduced in the given quadtree structure, the LQ code would require 3 additional bits (as one additional literal of 3 bits has to be added to code) while the CBLQ code would require 8 additional bits (as the entire added level must be encoded, i.e. four additional literals of 2 bits have to be added to the code). Hence, the deeper the structures of quadtrees with few black cells, the more advantageous the LQ code should be in theory. However, if there would be many black cells, it is obvious that the LQ code quickly becomes disadvantageous.

In the results, for all the quadtree-utilizing approaches, the share of LQ-encoded summaries is greater than the share of CBLQ-encoded summaries. Most resources of the T1 collection administer few data points and

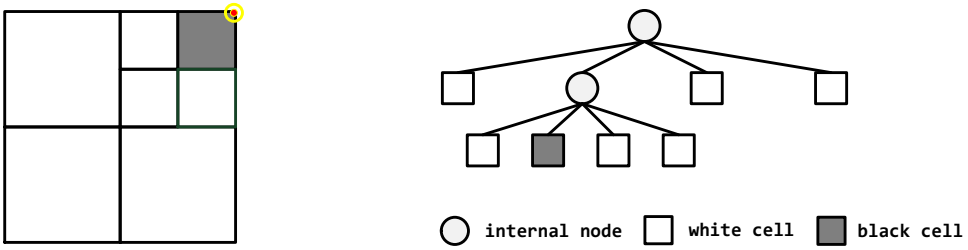


Fig. 36: Example quadtree illustrating the LQ code's advantageousness over the CBLQ code for quadtrees with few black cells.

are spatially narrow, i.e. most quadtrees should only feature one or few black nodes—which favors the LQ code. Hence, the results confirm the general presumption. The CBLQ share is highest for $\text{QT}_{8192,0.001}$ and lowest for $\text{MBRQT}^{16,1.0}$/$\text{MARQT}^{16,1.0}_{2,10.0}$ which are plausible results considering the respective parameterizations. For $\text{MBRQT}^{c,a}$ and $\text{MARQT}^{c,a}_{k,sl}$, in general, it must be borne in mind that in case the basic rectangles have a surface area below a, no refining quadtrees are built. Such cases are registered as LQ-encoded summaries in our data.[145] Hence, these values have to be treated with caution as they are definitely skewed. We do not further pursue the extent of this skewing at this point. However, we return to this issue in later parts of the work (see section 13.2.1).
In the following, the approaches (excluding the MBR approach) are divided into groups based on their characteristics, and each group is examined separately.

7.2. Results for the MBR-based Approaches

As first group, the approaches based on full-precision rectangles[146] are assessed: $\text{RecMAR}_{k,sl}$, $\text{MBRQT}^{c,a}$, and $\text{MARQT}^{c,a}_{k,sl}$. See Figure 37 for a visualization of the respective Skylines.
Generally, the Skylines of the three approaches (which only start at *rds* values of ~41.2 B to ~41.6 B) show very similar characteristics up to an *rds* value of ~42.0 B. Afterwards, the $\text{RecMAR}_{k,sl}$ Skyline clearly drops off as at this point, it exhibits a sharp bend (i.e. a sudden and sharp flattening of the curve) which does not conform to the typical, rather smooth course of the asymptotic curves. The initially involved parameterization is $\text{RecMAR}_{9,1.0}$ (*rds*: 41.88 B, *rfc*: 3.33 ‰) which is followed by $\text{RecMAR}_{2,0.01}$ (*rds*: 45.87 B, *rfc*: 2.88 ‰) as the next anchor point on the Skyline. By these numbers, it is evident that the actual number of MARs built (and thus the resulting *rds* value) is more dependent on parameter sl (threshold distance between rectangle center point and associated data points after which the construction of additional MARs is stopped) than on parame-

[145] This is because then, the bits for the resource description type (see Table 3 on page 121) are all set to *false* which corresponds to the code of the 'lq-nz'-option.
[146] Note that for convenience, we refer to this group as 'MBR-based approaches' even though technically, it ought to be 'full-precision-rectangle-based approaches'.

ter k (maximum possible number of rectangles), and that sl is generally the more important parameter. This is plausible since in principle, sl determines if—for the given data collection as a whole—the MARs are too coarse, or whether too many MARs are built in a narrow space. For the RecMAR$_{k,sl}$ Skyline, the variation of $sl = 1.0$ (*dsu*) to $sl = 0.01$ is obviously too large for a smooth course of the asymptotic curve. With interim values such as $sl = 0.1$, the curve could most likely be smoothed out, leading to a better overall result for RecMAR$_{k,sl}$. Generally, it would be interesting to parameterize RecMAR$_{k,sl}$ solely with the sl parameter (i.e. building as many MARs as are necessary for the targeted spatial accuracy) in order to increase the adaptiveness of RecMAR$_{k,sl}$. Restricting k to 9 at a maximum may also be the reason why the maximum RecMAR$_{k,sl}$ selectivity (with 0.41‰ *rfc* or ~101 contacted resources for RecMAR$_{9,1.0E-5}$) does not quite reach the maximum selectivity of other approaches. Nevertheless, even with adjusted parameterizations, it would be unlikely that the same performance as for MBRQT^{c,a} or MARQT$^{c,a}_{k,sl}$ can be achieved since the latter are able to index additional areas with much less additional storage space: for each additional area, RecMAR$_{k,sl}$ requires the $4 \cdot 32$ bits of a full-precision rectangle. Furthermore, the MAR calculation is computationally much more expensive than a quadtree refinement—which becomes even worse with regard to a theoretically unbounded k as suggested above.

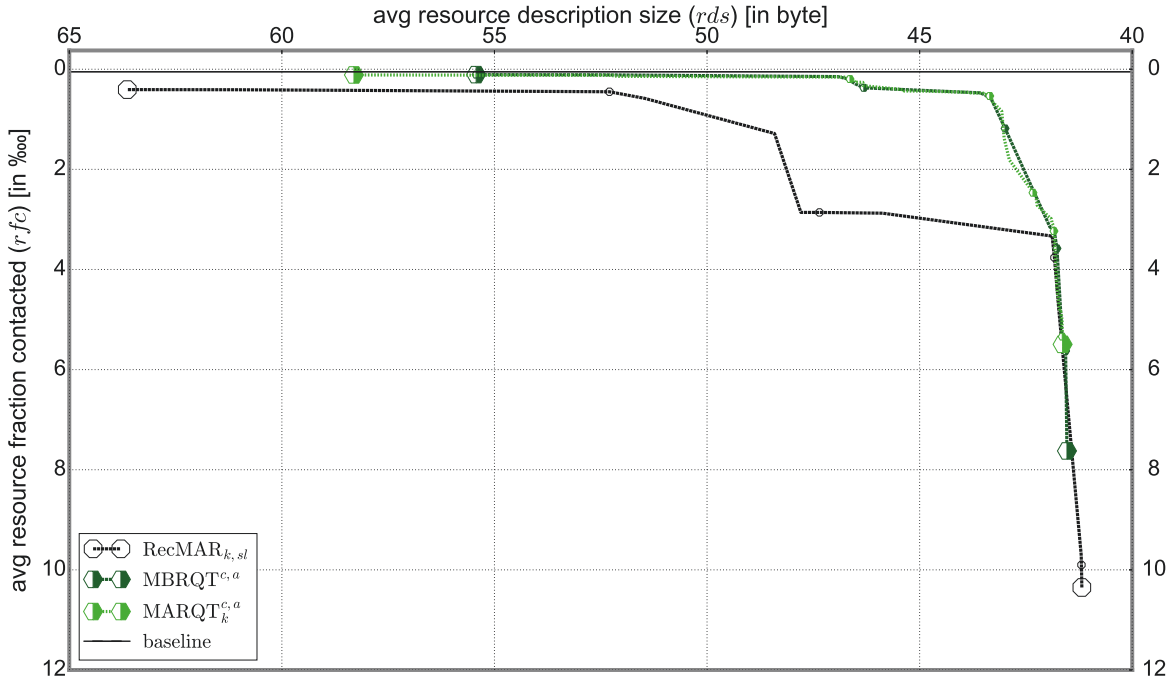


Fig. 37: Full Skylines of the three approaches being *based* on full-precision rectangles.

For MBRQT^{c,a} and MARQT$^{c,a}_{k,sl}$, the Skylines are almost identical. The first suspicion is that for MARQT$^{c,a}_{k,sl}$, it must be beneficial to set the parameters in a way that the resulting descriptions are as similar to the MBRQT^{c,a} descriptions as possible. As just analyzed for RecMAR$_{k,sl}$, the

number of MARs built is primarily dependent on parameter sl. Hence, if the MARQT$^{c,a}_{k,sl}$ resource descriptions shall be as similar to the MBRQT^{c,a} resource descriptions as possible, the Skyline parameterizations should mostly feature high sl values since then, only one MAR is built by tendency. The presumption can be confirmed by the experimental results: Out of the 87 MARQT$^{c,a}_{k,sl}$ Skyline parameterizations,

— 59 parameterizations feature $sl = 10.0$,
— 19 parameterizations feature $sl = 1.0$,
— 6 parameterizations feature $sl = 0.1$, and
— 3 parameterizations feature $sl = 0.01$.

Thus, the reason for the great similarity of the MBRQT^{c,a} and MARQT$^{c,a}_{k,sl}$ Skylines has been revealed. Additionally, it demonstrates once again that additional storage space is usually better invested into refining quadtree structures than into additional full-precision rectangles.
The MBRQT^{c,a} Skyline exhibits a bend at ~43.6 *rds*. The parameterization at the start of the bend is MBRQT$^{8192,0.001}$ (*rds*: 43.58 B, *rfc*: 0.47‰), followed by MBRQT$^{256,1.0E-5}$ (*rds*: 46.31 B, *rfc*: 0.37‰) at the end. Similar to the parameters sl and k for RecMAR$_{k,sl}$, this shows that the accuracy and the storage space requirements of quadtree structures are primarily determined by the accuracy-limiting parameter a and not by c defining the maximum possible number of quadtree cells. Even though the bend is not as pronounced as for RecMAR$_{k,sl}$, a smoother course of the Skyline would most likely be achievable by using more fine-grained variations of parameter a.
Altogether, the Skylines evidence that utilizing multiple full-precision rectangles is not suitable for the given data collection.
Table 6 depicts the numbers of the respective parameterizations at 'MBR-size' once again. Since all three approaches rely on a full-precision MBR as their description base and not much refinement can have happened yet, the values are very similar for all techniques. For MBRQT$^{16,1.0}$ and MARQT$^{16,1.0}_{2,10.0}$, the depicted *rds* values ('avg rds'-row) are slightly greater than for RecMAR$_{2,10.0}$ and as a consequence, their *rfc* values ('avg rfc'-row) are slightly better. Nevertheless, in this scope, there are generally almost no differences between the approaches (also see Figure 37). A more profound insight is that the MBR base can be greatly improved by only few additional expenses. For example, for RecMAR$_{2,10.0}$, only 0.14 B *rds* more are spent for the resource descriptions—but the selectivity is more than twice as good (10.35 ‰ versus 21.13 ‰ for the MBR approach).
For the further evaluation, MBRQT^{c,a} is selected to represent the MBR-based approaches. It clearly outperforms RecMAR$_{k,sl}$ and in addition, it is computationally much cheaper. Furthermore, it offers basically the same general performance as MARQT$^{c,a}_{k,sl}$ which for its part

— is computationally considerably more complex due to the (possible) MAR computation,

— requires four parameters to be specified instead of only two, and
— whose Skyline parameterizations indicate that $\text{MARQT}_{k,sl}^{c,a}$ summaries which are very similar the $\text{MBRQT}^{c,a}$ summaries are most advantageous, anyway.

Table 6: Comparison of the descriptive statistic *rds* and *rfc* values as well as of the shares for the resource description transmission methods at 'MBR-size' for the approaches being *based* on full-precision MBRs (excerpt from Table 5).

technique → key figure ↓	$\text{RecMAR}_{2,10.0}$	$\text{MBRQT}^{16,1.0}$	$\text{MARQT}_{2,10.0}^{16,1.0}$
avg *rds* [in byte]	41.19	41.53	41.64
min *rds* [in byte]	36	36	36
25%-quant. *rds* [in byte]	36	36	36
median *rds* [in byte]	44	44	44
75%-quant. *rds* [in byte]	44	45	45
max *rds* [in byte]	60	50	71
avg *rfc* [in ‰]	10.35	7.62	5.50
min *rfc* [in ‰]	0.30	0.04	0.04
25%-quant. *rfc* [in ‰]	4.58	2.64	1.54
median *rfc* [in ‰]	9.25	6.03	3.73
75%-quant. *rfc* [in ‰]	14.40	9.74	7.37
max *rfc* [in ‰]	31.89	31.01	26.30
sum-z/lq-z [in %]	$\ll 0.01$	0	0
sum-nz/lq-nz [in %]	47.45	47.10	46.89
dr-z [in %]	$\ll 0.01$	0	$\ll 0.01$
dr-nz [in %]	52.55	52.33	52.55
cblq-z [in %]	-	0	0
cblq-nz [in %]	-	0.57	0.56

7.3. Results for the k-d-based Approaches

In the following, the k-d-based approaches (KD_n, KDMBR_n^b, $\text{KDMAR}_n^{b,k}$, and $\text{KDQT}_n^{b,k}$) are compared against each other. The Skylines in Figure 38 show that KD_n (pure global space partitioning, no refinement) is far behind over almost the entire *rds* range. Only in the course of the general convergence of the Skylines for high *rds* values, it gets closer to the hybrid approaches. But even then, the KD_n Skyline still lags a little behind: for KD_{16384} (*rds*: 49.4 B), the *rfc* is ~1.10‰ (corresponding to ~274 contacted resources) while for e.g. KDMBR_{8192}^4 (*rds*: 49.5 B), the *rfc* is ~0.27‰ (corresponding to ~67 contacted resources). In total, this shows that pure global space partitioning approaches—even with a very large

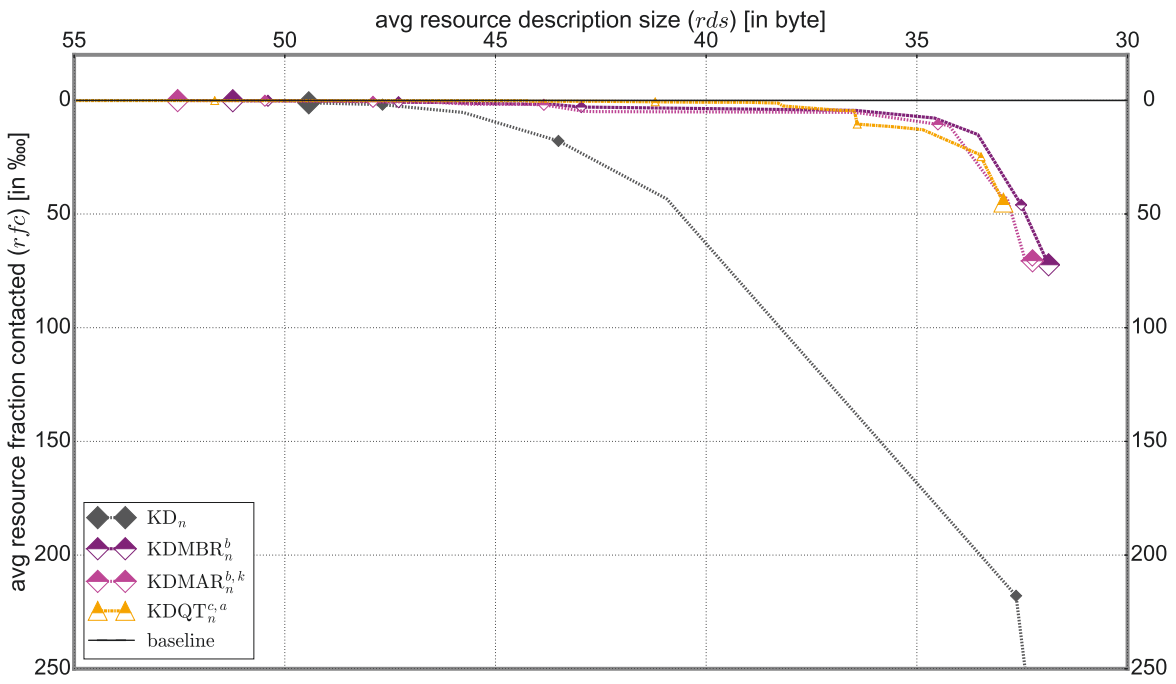


Fig. 38: Skylines of the four approaches being *based* on a global k-d space partition.

number of subspaces such as 16,384—are not the best fit for data collections with that many resources: Obviously, too many resources share the single cells of the global space partition with each other. A certain degree of 'distinctiveness' of the areas indexed in the resources' summaries is indispensable, obviously. For a global space partitioning base, this can be attained by refining the occupied subspaces. Regarding these refinements, it shows that quantized rectangles (KDMBR^b_n, KDMAR$^{b,k}_n$) are slightly superior to subspace-internal quadtrees (KDQT$^{c,a}_n$) for very small *rds* values up to ~36.5 B. Afterwards, KDQT$^{c,a}_n$ is superior to KDMBR^b_n and KDMAR$^{b,k}_n$ for the remainders of the respective Skylines.
In a comparison of KDMBR^b_n and KDMAR$^{b,k}_n$, the KDMBR^b_n Skyline is slightly better for almost the entire *rds* range (only between ~47.0 B and ~48.0 B, the KDMAR$^{b,k}_n$ Skyline is minimally ahead). For both approaches, in general, it shows that low values for parameter b (number of quantization bits per bound) are not beneficial. Looking at the 15 Skyline parameterizations of KDMBR^b_n, only 1 features $b = 3$, 3 feature $b = 4$, 5 feature $b = 6$, and all 6 of the $b = 8$ parameterizations are present.[147] For the 34 KDMAR$^{b,k}_n$ Skyline parameterizations, it is as follows:

— 3 feature $b = 3$, 13 feature $b = 4$, and all 18 $b = 6$ parameterizations are present.
— Regarding parameter k, 11 feature $k = 2$, 11 feature $k = 3$, and 12 feature $k = 4$.

[147]Note that due to space restrictions, it is not possible to list the Skyline parameterizations for the approaches explicitly (especially since there are also approaches with more than 60 Skyline parameterizations, see Table 4 on page 132).

For both KDMBR$_n^b$ and KDMAR$_n^{b,k}$, the tested values for the number of subspaces n (defining the description base) are fairly equally present. Only $n = 64$ is underrepresented with only three occurences. The Skyline parameterizations of both approaches show that for a very low number of global subspaces (such as 32), it is more beneficial to invest additional storage space into more accurately quantized rectangles than into a (bit-wise) equivalent number of additional global subspaces. Still, also for higher values of n, an accurate quantization of the internal rectangles is very important: the non-Skyline parameterizations lagging especially far behind all feature $b = 3$.[148] The importance of parameter b is a bit surprising since an additional cell of the k-d space partition (increasing the accuracy of the description base which is shared by *all* resources) only accounts for 1 bit in a resource summary. In contrast, the $4 \cdot b$ bits for refining a single occupied k-d cell (increasing the *individual* refinement accuracy) appear to be comparatively expensive. Nevertheless, they are obviously a more efficient way of adding more 'distinctiveness' to a resource's description.[149]
For both KDMBR$_n^b$ and KDMAR$_n^{b,k}$, the Skylines take a fairly sharp bend at ~36.5 B *rds* (where they then fall behind the KDQT$_n^{c,a}$ Skyline). At this point, the Skyline parameterizations of KDMBR$_n^b$ and KDMAR$_n^{b,k}$ change from $n = 64$ to $n = 256$. In general, such bends are always observable when parameter n is increased. Softer raises of n would most likely smoothen out the curves (leading to slightly better overall results). Another option to further smoothen out the curves could possibly be to utilize greater values than 8 (KDMBR$_n^b$) respectively 6 (for KDMAR$_n^{b,k}$) for parameter b. Nevertheless, this has to be carefully assessed because at some stage, one also gets to a point where too many bits are used per bound—which has a negative impact on the efficiency achieved. For similar resulting *rds* values, the values for parameter n are always the same for the corresponding KDMBR$_n^b$ and KDMAR$_n^{b,k}$ Skyline parameterizations. Thus, due to using the same seeding in the training phase for 'learning' the global k-d space partition, both their description bases are the same, and the occuring differences in the results are solely caused by the varying refinement of occupied cells. In these situations, KDMBR$_n^b$ features higher b values while for KDMAR$_n^{b,k}$, b is lower but more than one rectangle can be utilized for refinement. In total, it is evident that the MAR computation alongside the rectangle number reduction procedure lead to roughly the same overall result as the simple further increase of b for a quantized MBR, even slightly favoring KDMBR$_n^b$. This is a bit of a disappointment

[148] Note that in this thesis, the only figure also showing non-Skyline parameterizations is Figure 29 on page 111 where the determination of the RecMAR$_{k,sl}$ Skyline is exemplarily depicted. In Figure 29, it can be seen that for example RecMAR$_{3,1.0E-5}$ is especially far away from being part of the Skyline.

[149] The $4 \cdot b$ bits apply in the case of KDMBR$_n^b$. For KDMAR$_n^{b,k}$, it can be up to $4 \cdot b \cdot k$ bits per occupied cell.

with regard to the very complex and computationally very costly MAR refinement.
Concerning KDQT$_n^{c,a}$, it is evident that preferably, parameterizations forge the Skyline which build a low number of global subspaces (parameter n) but refine occupied subspaces with detailed cell-interior quadtrees (which are primarily dependent on parameter a). For example, 13 Skyline parameterizations feature $n = 32$, but only 2 feature $n = 2048$. In contrast, only 1 Skyline parameterization features $a = 1.0$ (*dsu*^{2}) but 12 feature $a = 1.0E-7$, and only 4 Skyline parameterizations feature $c = 16$ while 15 feature $c = 512$.[150] Therefore, it is now clear why for KDQT$_n^{c,a}$, there is a much lower tendency of zipping summaries compared to the other k-d-based approaches (see the values in the 'sum-z/lq-z'-row and the 'cblq-z'-row of Table 4): Most of the data contained in KDQT$_n^{c,a}$ summaries is quadtree data which is generally unsuitable for compression due to its high entropy.

Table 7: Comparison of the descriptive statistic *rds* and *rfc* values as well as of the shares for the resource description transmission methods at 'MBR-size' for the approaches being *based* on a global k-d space partition (excerpt from Table 5).

technique → key figure ↓	KD$_{256}$	KDMBR$_{256}^{6}$	KDMAR$_{256}^{4,2}$	KDQT$_{32}^{64,1.0E-5}$
avg *rds* [in byte]	40.93	42.96	42.72	41.21
min *rds* [in byte]	29	31	31	32
25%-quant. *rds* [in byte]	36	36	36	36
median *rds* [in byte]	37	40	39	38
75%-quant. *rds* [in byte]	47	51	50	45
max *rds* [in byte]	61	565	717	486
avg *rfc* [in ‰]	43.41	2.97	4.97	0.67
min *rfc* [in ‰]	6.22	0.09	0.04	0.04
25%-quant. *rfc* [in ‰]	27.23	1.03	1.23	0.32
median *rfc* [in ‰]	38.39	1.56	2.58	0.50
75%-quant. *rfc* [in ‰]	49.24	3.02	3.93	1.00
max *rfc* [in ‰]	202.8	24.36	35.06	2.69
sum-z/lq-z [in %]	12.76	11.81	11.99	0
sum-nz/lq-nz [in %]	44.09	39.95	41.00	55.91
dr-z [in %]	0	$\ll 0.01$	$\ll 0.01$	0
dr-nz [in %]	43.15	48.24	47.00	34.21
cblq-z [in %]	-	-	-	$\ll 0.01$
cblq-nz [in %]	-	-	-	9.88

[150] Even though parameter a is generally more important for the number of actually resulting quadtree cells than parameter c—as already discussed in section 7.2 in association with the bend of the MBRQT^{c,a} Skyline—also the c values of the KDQT$_n^{c,a}$ Skyline parameterizations illustrate that preferably, detailed quadtrees are built for refinement.

Table 7 shows the approaches for their Skyline parameterizations resulting in *rds* values at 'MBRsize'. Note that for KDMBR_n^b and $\text{KDMAR}_n^{b,k}$, there is no Skyline parameterization resulting in *rds* values *really* close to 'MBRsize' (41.06 B *rds*). Hence, the *rds* values for these two are a little greater (KDMBR_{256}^6: 42.96 B *rds*; $\text{KDMAR}_{256}^{4,2}$: 42.72 B *rds*).[151] Despite the greater *rds* value, both KDMBR_{256}^6 (*rfc*: 2.97‰, corresponding to ~740 contacted resources on average) and $\text{KDMAR}_{256}^{4,2}$ (~1,238 contacted resources) are significantly worse than $\text{KDQT}_{32}^{64,1.0E-5}$ (~161 contacted resources). For $\text{KDQT}_{32}^{64,1.0E-5}$, the quadtree data is obviously the dominant part since only $n = 32$ global subspaces are built which are refined by rather detailed quadtrees ($c = 64$, $a = 1.0E-5$) in case of being occupied. KD_{256} is far behind the other three approaches: the *rfc* value of e.g. $\text{KDQT}_{32}^{64,1.0E-5}$ is ~67 times better. Generally, the results of the descriptive statistic *rfc* values (rows 9 to 13) once again coincide with the *rfc* value (row 8), i.e. the approaches which are superior on average are also consistently better.
Since the *rds* value for $\text{KDQT}_{32}^{64,1.0E-5}$ is lower than for both KDMBR_{256}^6 and $\text{KDMAR}_{256}^{4,2}$, it is no surprise that the majority of resource descriptions is smaller for $\text{KDQT}_{32}^{64,1.0E-5}$ (see the 'median rds'- and '75%-quant. rds'-rows of Table 7). The occuring maximum resource description size ('max rds'-row) is also smaller for $\text{KDQT}_{32}^{64,1.0E-5}$. Furthermore, significantly less resources are directly represented (see the 'dr-z'- and 'dr-nz'-rows of Table 7), and almost no zipping is applied: There are only 21 'cblq-z'-represented resources for $\text{KDQT}_{32}^{64,1.0E-5}$ while for both KDMBR_{256}^6 and $\text{KDMAR}_{256}^{4,2}$, about 12% (or ~300,000) of the resource summaries are zipped. Finally, the share of directly represented resources and zipped summaries for $\text{KDQT}_{32}^{64,1.0E-5}$ is also significantly lower than for KD_{256}. All in all, this shows that very storage-space-efficient yet accurate summaries are built for $\text{KDQT}_{32}^{64,1.0E-5}$ which clearly outperform the comparable KD_{256}, KDMBR_{256}^6, and $\text{KDMAR}_{256}^{4,2}$ techniques–despite being at disadvantage with regard to the amount of directly represented resources and the frequency in which the zipping is applied.
In a comparison of KDMBR_{256}^6 and $\text{KDMAR}_{256}^{4,2}$, the numbers in Table 7 indicate that often, occupied subspaces are only refined by a single MAR for $\text{KDMAR}_{256}^{4,2}$: Both KDMBR_{256}^6 and $\text{KDMAR}_{256}^{4,2}$ have the same amount of global subspaces ($n = 256$). Hence, due to the seeding in the training phase, the k-d space partition is exactly the same for both. Consequently, the differences between the techniques must be exclusively due to the differing refinement of the occupied subspaces. The maximally usable amount of storage space for refining an occupied subspace is greater for $\text{KDMAR}_{256}^{4,2}$ ($4 \cdot (b = 4) \cdot (k = 2) = 32$ bits) than for KDMBR_{256}^6 ($4 \cdot (b = 6) = 24$ bits). Thus, if most of the times, $k = 2$ MARs would be used for refinement, the

[151] However, the comparability of the techniques is not affected by this circumstance since both the KDMBR_n^b and the $\text{KDMAR}_n^{b,k}$ Skyline are very flat between 36.5 B *rds* and 43.0 B *rds*.

rds value of $\text{KDMAR}^{4,2}_{256}$ would have to be greater. Nevertheless, the opposite is the case: the *rds* value of KDMBR^{6}_{256} is slightly greater (42.96 B versus 42.72 B). This can also be deduced from the values in the 'median rds'- and the '75%-quant. rds'-rows which are slightly greater for KDMBR^{6}_{256}, too. Therefore, it can be concluded that there is either mostly only one data point in an occupied cell, or the 'rectangle number reduction procedure' often reduces the number of MARs for refining an occupied cell to 1. Either way, for $\text{KDMAR}^{4,2}_{256}$, it results in a less accurate and less storage-space-intensive refinement since only $b = 4$ are used for quantizing a bound of the one remaining refinement-rectangle instead of $b = 6$ bits as for KDMBR^{6}_{256}. With regard to the further evaluation, both KDMBR^{b}_{n} as well as $\text{KDQT}^{c,a}_{n}$ are selected for continued consideration. KDMBR^{b}_{n} is far better than the unrefined KD_{n} approach and for the greatest part of the *rds* range, it is also better than $\text{KDMAR}^{b,k}_{n}$. Additionally, due to the MAR computation and the rectangle number reduction procedure of $\text{KDMAR}^{b,k}_{n}$, KDMBR^{b}_{n} is significantly cheaper from a computational point of view. On top, it requires only two parameters instead of three. Hence, out of these three approaches (KD_{n}, KDMBR^{b}_{n}, and $\text{KDMAR}^{b,k}_{n}$), KDMBR^{b}_{n} is certainly the most suitable approach for a further evaluation. $\text{KDQT}^{c,a}_{n}$ significantly outperforms KDMBR^{b}_{n} after about 36.5 B *rds* have been reached—which is the more interesting *rds* range compared to very low *rds* values. Therefore, in principle, it would be the most interesting approach for further investigation. Nevertheless, the parameter values of the Skyline parameterizations indicate that the way in which $\text{KDQT}^{c,a}_{n}$ achieves its performance differs noticeably from the other k-d-based approaches. Due to this difference, both $\text{KDQT}^{c,a}_{n}$ as well as KDMBR^{b}_{n} are selected for continued consideration to also further clarify on in which situations the refinement with quantized MBRs is better suited than the refinement with quadtrees (and vice versa).

7.4. Results for the Quadtree-based Approaches

In the following, the quadtree-*based* approaches ($\text{QT}_{c,a}$, $\text{QTMBR}^{b}_{c,a}$, and $\text{QTMAR}^{b,k}_{c,a}$) and $\text{GridQT}^{c,a}_{r}$ are assessed. In total, $\text{QTMBR}^{b}_{c,a}$ offers the best Skyline of this group (see Figure 39). It is clearly superior to the other three approaches for low *rds* values (smaller than 36.0 B) and is basically on par with the Skyline for greater *rds* values. In comparison with its 'description base', the unrefined $\text{QT}_{c,a}$, it is superior for the entire *rds* range. It is not too surprising that a hybrid approach outperforms its non-hybrid description base. On the other hand, this is also not necessarily the only conceivable outcome, especially since $\text{QT}_{c,a}$ is already very competitive (see e.g. Figure 34 on page 130). In principle, both $\text{QT}_{c,a}$ as well as $\text{QTMBR}^{b}_{c,a}$ describe rectangular areas and adapt to the data point locations of the individual resource: For $\text{QTMBR}^{b}_{c,a}$, the summary is a combination of a basic quadtree structure with quantized MBRs for occupied cells while for $\text{QT}_{c,a}$, it is 'solely' a quadtree structure which is more detailed for com-

pensation. Hence, the quantized MBRs and the more detailed quadtree structure are conceptually very similar and thus interchangeable means of summarization. In the end, the question is simply which summarization is the more efficient one. Thus, it is nowhere near stringently expectable that the $\text{QTMBR}^{b}_{c,a}$ Skyline is ahead of the $\text{QT}_{c,a}$ Skyline for the entire *rds* range.

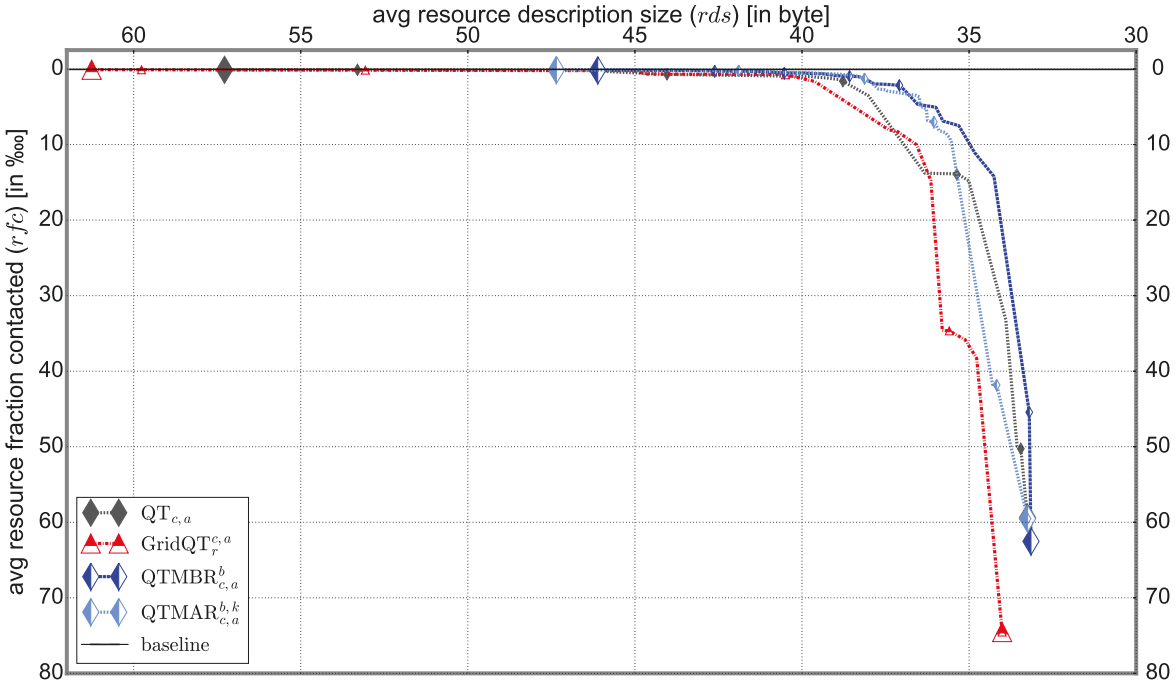


Fig. 39: Full Skylines of the three approaches being *based* on a local quadtree space partition as well as $\text{GridQT}^{c,a}_{r}$.

$\text{QTMAR}^{b,k}_{c,a}$ lags behind for very low *rds* values but closes up to $\text{QTMBR}^{b}_{c,a}$ no later than at ~36.5 B *rds*. Since $\text{QTMAR}^{b,k}_{c,a}$ is worse than $\text{QT}_{c,a}$ for very low *rds* values, it seems that here, using a single quantized rectangle is beneficial for refinement but using multiple quantized MARs is not as efficient (both in comparison to simply adding more quadtree cells). $\text{GridQT}^{c,a}_{r}$ is the worst approach, only briefly overtaking $\text{QT}_{c,a}$ when the $\text{QT}_{c,a}$ Skyline flattens sharply—which happens several times and suggests suboptimal tested parameter combinations or not fine enough gradations of parameter values for $\text{QT}_{c,a}$.
For example, the $\text{QT}_{c,a}$ Skyline bends between ~35.0 B *rds* and ~36.3 B *rds*. A look at the corresponding parameterizations shows that during the flattening, the spatial accuracy of $\text{QT}_{c,a}$ is limited by parameter a being stuck at 0.1 (dsu^2), beginning at $\text{QT}_{64,0.1}$ (*rds*: 35.0 B, *rfc*: 14.8 ‰) and ending at $\text{QT}_{8192,0.1}$ (*rds*: 36.3 B, *rfc*: 13.7 ‰). After switching to $a = 0.001$, the Skyline steepens again—with $\text{QT}_{64,0.001}$ as next anchor point (*rds*: 38.0 B, *rfc*: 3.5 ‰). The $\text{QT}_{c,a}$ Skyline only falls behind the $\text{GridQT}^{c,a}_{r}$ Skyline in similar cases. Hence, $\text{QT}_{c,a}$ must generally be regarded as superior to $\text{GridQT}^{c,a}_{r}$. For $\text{QTMBR}^{b}_{c,a}$ and $\text{QTMAR}^{b,k}_{c,a}$, there are no similar sharp bends in the respective Skylines. For both, smaller performance boosts (i.e. a

steepening of the Skyline) occur when especially the parameters b and a are switched to values resulting in more accurate resource summaries (i.e. the values for b are raised while the values for a are decreased). In case parameter c is stuck at a very low value ($c \leq 32$), also its raise results in a small performance boost. For GridQT$_r^{c,a}$, the performance boosts (similarly to QT$_{c,a}$) coincide with setting parameter a to a smaller value—as long as parameter r is fixed to 4. On the other hand, the GridQT$_r^{c,a}$ Skyline flattens sharply after r is raised to 8. Here, even by consuming ~4.0 B *rds* more, only minimal selectivity improvements can be achieved from the start of the bend (GridQT$_4^{512,0.001}$, *rds*: 40.64 B, *rfc*: 0.719 ‰) until the end of the bend (GridQT$_8^{256,0.001}$, *rds*: 44.61 B, *rfc*: 0.674 ‰)—corresponding to contacting ~11 resources less on average. Since the majority of resources have a narrow spatial footprint, increasing the number of cells of the basic (regular and global) grid mostly results in storage space overhead but basically has no effect on the spatial accuracy of the GridQT$_r^{c,a}$ description, obviously.

Looking at the Skyline parameterizations of QTMBR$_{c,a}^b$, it is once again evident that parameter a is much more decisive for the resulting *rds* (and *rfc*) values than parameter c. For example, QTMBR$_{1024,1.0}^6$ results in 37.84 B *rds* (*rfc*: 1.97 ‰) whereas QTMBR$_{512,0.01}^6$ results in 42.61 B *rds* (*rfc*: 0.26 ‰). For the 38 QTMBR$_{c,a}^b$ Skyline parameterizations, the tested values for the parameters a and b are represented as follows:

— a: 12 parameterizations feature $a = 1.0$, 10 feature $a = 0.1$, 5 feature $a = 0.05$, 7 feature $a = 0.01$, and 4 feature $a = 0.001$.
— b: 11 parameterizations feature $b = 3$, 12 feature $b = 4$, 13 feature $b = 6$, and 2 feature $b = 8$.

Obviously, setting parameter b to values greater than 6 is not very efficient and additional storage space is better invested into more detailed quadtree structures. This is a difference compared to KDMBR$_n^b$—which also refines occupied cells with quantized MBRs but utilizes a global space partition as a description base (in contrast to the local space partition of QTMBR$_{c,a}^b$)—where 8 is the most represented value for parameter b. Regarding parameter a, predominantly greater values (resulting in less detailed basic quadtrees) are present—it is plausible that an additional refinement of an occupied cell is more efficient if the cell is rather coarse than when it is already very accurate.

For QTMAR$_{c,a}^{b,k}$, parameter k is not of great importance since due to the (usually low) number of data points in an occupied cell as well as the rectangle number reduction procedure, often less than k MARs refine an occupied cell. Therefore, the tested values for parameter k are fairly equally present in the 64 QTMAR$_{c,a}^{b,k}$ Skyline parameterizations. For the parameters a and b, it is as follows:

— a: 37 parameterizations feature $a = 1.0$, 12 feature $a = 0.1$, 8 feature $a = 0.05$, 3 feature $a = 0.01$, and 4 feature $a = 0.001$.
— b: 12 parameterizations feature $b = 3$, 21 feature $b = 4$, and 31 feature $b = 6$.

Hence, regarding parameter a, results are similar to $\text{QTMBR}^{b}_{c,a}$. The $\text{QTMAR}^{b,k}_{c,a}$ Skyline parameterizations resulting in a worse performance than comparable $\text{QT}_{c,a}$ Skyline parameterizations (which occur for very low *rds* values) all exhibit a low maximum number of quadtree cells (i.e. $c = 16$) and very coarsely quantized MARs (i.e. $b = 3$ or $b = 4$). For k, all three tested values are present in the corresponding Skyline parameterizations that result in very low *rds* values. In comparison, the $\text{QTMBR}^{b}_{c,a}$ Skyline parameterizations with similarly low *rds* values feature greater values for b and are better than both their $\text{QT}_{c,a}$ as well as $\text{QTMAR}^{b,k}_{c,a}$ counterparts. Hence, when there is only little storage space available and given a rather coarse quadtree structure as description base, additional expenses are best invested into accurately quantized single MBRs before adding additional cells to the quadtree structure. Utilizing several, rather coarsely quantized rectangles as a refinement is the most inefficient approach here.
Generally, when comparing $\text{QTMBR}^{b}_{c,a}$ and $\text{QTMAR}^{b,k}_{c,a}$ for their different Skyline parameterizations resulting in similar *rds* values, it is evident that the greater number of quantized refinement rectangles for $\text{QTMAR}^{b,k}_{c,a}$ is compensated with either a greater number of quadtree cells or a more detailed quantization for $\text{QTMBR}^{b}_{c,a}$—there is no evidence that one of these compensation alternatives is the preferable option. From ~38.4 B *rds* on, the $\text{QTMAR}^{b,k}_{c,a}$ Skyline is slightly ahead of the $\text{QTMBR}^{b}_{c,a}$ Skyline but the differences are steadily marginal (staying in range of $\sim(15 +/- 10)$ contacted resources for similar *rds* values).
For the Skyline parameterizations of $\text{QT}_{c,a}$, it is evident that $a = 1.0E{-}7$ and $a = 1.0E{-}8$ are too detailed values for efficient resource summaries: Both values are only represented once in the 27 $\text{QT}_{c,a}$ Skyline parameterizations and result in very large *rds* values (53.30 B for $\text{QT}_{8192,1.0E-7}$ and 57.28 B for $\text{QT}_{8192,1.0E-8}$). For $\text{GridQT}^{c,a}_{r}$, as for the other approaches of this group, parameter c has little meaning for the resulting *rds* values. These are once again primarily dependent on parameter a: For e.g. $\text{GridQT}^{512,1.0}_{4}$, 35.86 B *rds* result while for $\text{GridQT}^{256,1.0E-5}_{4}$, 44.93 B *rds* result. Similar to $\text{KDQT}^{c,a}_{n}$, the $\text{GridQT}^{c,a}_{r}$ Skyline is predominantly built by parameterizations featuring a low number of global subspaces (i.e. $r = 4$ or $r = 8$) but detailed quadtrees for refining occupied subspaces. Only for very big *rds* values (greater than 53.0 B), there are Skyline parameterizations with fine-grained grids, featuring $r = 32$ or $r = 64$ (and at the same time very detailed quadtrees). Since the $\text{GridQT}^{c,a}_{r}$ Skyline is very flat from ~45.0 B *rds* on, this cannot be seen as an indication that detailed grids might also be an efficient description base. It is rather attributable to the fact that the Skylines are forged by *all* non-dominated parameterizations—even those

resulting in only minimal *rfc* improvements while expending huge amounts of additional storage space.

Table 8: Comparison of the descriptive statistic *rds* and *rfc* values as well as of the shares for the resource description transmission methods at 'MBR-size' for the approaches being *based* on a local quadtree space partition as well as GridQT$_r^{c,a}$ (excerpt from Table 5).

technique → key figure ↓	QT$_{8192,0.001}$	GridQT$_4^{512,0.001}$	QTMBR$_{512,0.05}^{6}$	QTMAR$_{256,0.1}^{6,2}$
avg *rds* [in byte]	39.95	40.64	41.04	41.01
min *rds* [in byte]	34	34	36	36
25%-quant. *rds* [in byte]	36	36	36	36
median *rds* [in byte]	37	37	38	38
75%-quant. *rds* [in byte]	41	42	45	44
max *rds* [in byte]	1,948	2,235	1,154	1,303
avg *rfc* [in ‰]	0.92	0.72	0.39	0.31
min *rfc* [in ‰]	0.04	0.04	0.04	0.04
25%-quant. *rfc* [in ‰]	0.37	0.28	0.16	0.15
median *rfc* [in ‰]	0.73	0.46	0.25	0.23
75%-quant. *rfc* [in ‰]	1.12	0.73	0.51	0.44
max *rfc* [in ‰]	4.36	3.37	2.07	1.80
sum-z/lq-z [in %]	0	$\ll 0.01$	0	0
sum-nz/lq-nz [in %]	50.29	53.90	52.35	56.13
dr-z [in %]	0	0	$\ll 0.01$	0
dr-nz [in %]	36.20	36.75	42.00	40.50
cblq-z [in %]	0.01	0.01	$\ll 0.01$	$\ll 0.01$
cblq-nz [in %]	13.50	9.34	5.65	3.37

Table 8 shows the four quadtree-group approaches for their Skyline parameterizations at 'MBRsize'. For QT$_{c,a}$ and GridQT$_r^{c,a}$, there are no Skyline parameterizations with really close values. Since all the Skylines of the four approaches are already very flat at this point (only marginal selectivity improvements result even by investing a lot of additional storage space), it does not seriously affect the comparison, though. The numbers display that QTMBR$_{512,0.05}^{6}$ and QTMAR$_{256,0.1}^{6,2}$ are almost equally good (the latter being a bit better) while QT$_{8192,0.001}$ and GridQT$_4^{512,0.001}$ are ~1.8 to ~3 times worse. In general, all of these approaches show excellent results: In comparison with the MBR approach (*rfc*: 21.13 ‰), selectivity is improved by between ~95.6% (QT$_{8192,0.001}$) and ~98.5% (QTMAR$_{256,0.1}^{6,2}$). As always, the descriptive statistic *rfc* values ('min rfc'- to 'max rfc'-rows) correspond to the averaged values ('avg rfc'-row), i.e. the *rfc* values reflect the general performance well. The threshold surface areas (i.e. the respective a values which decide on a premature stop of the quadtree construction) for QTMBR$_{512,0.05}^{6}$ and QTMAR$_{256,0.1}^{6,2}$ are distinctly greater than for QT$_{8192,0.001}$

and $\text{GridQT}_4^{512,0.001}$—hence, the quadtree structures built for the former two should be significantly smaller. This is plausible since the latter two do not refine their quadtree structures any further. Nevertheless, in total, the former two achieve spatially more accurate descriptions: their *rfc* values are clearly better. Notably, the 'max rds'-values for $\text{QT}_{8192,0.001}$ and $\text{GridQT}_4^{512,0.001}$ are much greater than for $\text{QTMBR}_{512,0.05}^{6}$ and $\text{QTMAR}_{256,0.1}^{6,2}$, implying that the latter two are capable of describing spatially spread resources more efficiently at the respectively targeted spatial accuracy. Still, the majority of resource descriptions require *slightly* more storage space for $\text{QTMBR}_{512,0.05}^{6}$ and $\text{QTMAR}_{256,0.1}^{6,2}$ in comparison to the other two approaches (see 'min rds'- to '75%-quant. rds'-rows in Table 8).

In a comparison of $\text{QTMBR}_{512,0.05}^{6}$ and $\text{QTMAR}_{256,0.1}^{6,2}$, it shows that for the latter, less detailed quadtrees are built ($a = 0.05$ for $\text{QTMBR}_{512,0.05}^{6}$ versus $a = 0.1$ for $\text{QTMAR}_{256,0.1}^{6,2}$) but occupied cells are refined more accurately (one MBR quantized with $b = 6$ versus up to two MARs quantized with $b = 6$). By the numbers, the $\text{QTMAR}_{256,0.1}^{6,2}$ technique is superior ($\rightarrow$ better *rfc* value while requiring lower average *rds*)—but only marginally. Considering the notable additional computational efforts invested for $\text{QTMAR}_{256,0.1}^{6,2}$ over $\text{QTMBR}_{512,0.05}^{6}$ with regard to the summary creation (MAR computation and rectangle number reduction procedure for each occupied subspace), it does not really seem to be worthwhile overall: the difference corresponds to only ~20 contacted resources.

In total, $\text{QT}_{8192,0.001}$ and $\text{GridQT}_4^{512,0.001}$ are closely resembling each other in their key figures. This is because the $\text{GridQT}_4^{512,0.001}$ summaries should be fairly similar to the $\text{QT}_{8192,0.001}$ summaries: Only 32 global subspaces are built (i.e. the description base is very coarse) but the threshold surface area $a = 0.001$ for the quadtree construction abortion is the same. For both, summary sizes below 36 B occur ($\rightarrow$ 34 B, see 'min rds'-row) such that at least some resources that administer only one data point transmit a summary as their resource description. In contrast, for $\text{QTMBR}_{512,0.05}^{6}$ as well as $\text{QTMAR}_{256,0.1}^{6,2}$, no summary sizes below data-point-size occur (since the 'min rds'-value for both is 36 B), resulting in a greater share of directly represented resources for these. In total, this should be a slight advantage for $\text{QTMBR}_{512,0.05}^{6}$ and $\text{QTMAR}_{256,0.1}^{6,2}$ since the trade-off is a 34 B coarse summary representation which is a spatial approximation (additionally double-counting summary-represented resources contributing to the query result, see section 6.4.1) versus a 36 B direct representation which represents the exact spatial information.

For all approaches, hardly any zipping is applied. This is no surprise due to the high share of high entropy quadtree data in the summaries. For $\text{QT}_{8192,0.001}$ and $\text{GridQT}_4^{512,0.001}$, the share of CBLQ-encoded summaries is greater than for $\text{QTMBR}_{512,0.05}^{6}$ and $\text{QTMAR}_{256,0.1}^{6,2}$, again indicating that the LQ code is beneficial for quadtree structures with few black nodes while

the CBLQ code is better for quadtree structures with many black nodes (as parameter a has a significant impact on this).
For the further evaluation, $\text{QTMBR}^{b}_{c,a}$ is selected as most suitable representative from this group. It significantly outperforms $\text{QT}_{c,a}$ and $\text{GridQT}^{c,a}_{r}$ over the entire *rds* range. In comparison to $\text{QTMAR}^{b,k}_{c,a}$, it is clearly better for low *rds* values and basically on par for greater *rds* values. Additionally, $\text{QTMBR}^{b}_{c,a}$ is computationally much cheaper than $\text{QTMAR}^{b,k}_{c,a}$ and requires only three parameters to be specified instead of four.

7.5. Results for the Voronoi-based Approaches

In the following, the Voronoi-based approaches $\text{UFS}_{n,cc}$ and $\text{DFS}^{b}_{n,cc}$ are assessed in depth. This also involves a comparison of the spatial domain ranker and the metric-domain-like ranker (see section 5.1) for both approaches. The Skylines in Figure 40 show that $\text{UFS}_{n,cc}$ is superior to the refined $\text{DFS}^{b}_{n,cc}$ for very small resulting *rds* values (up until ~36.5 B *rds*) and for rather high resulting *rds* values (from ~43.0 B *rds* on). In between, $\text{DFS}^{b}_{n,cc}$ is better.

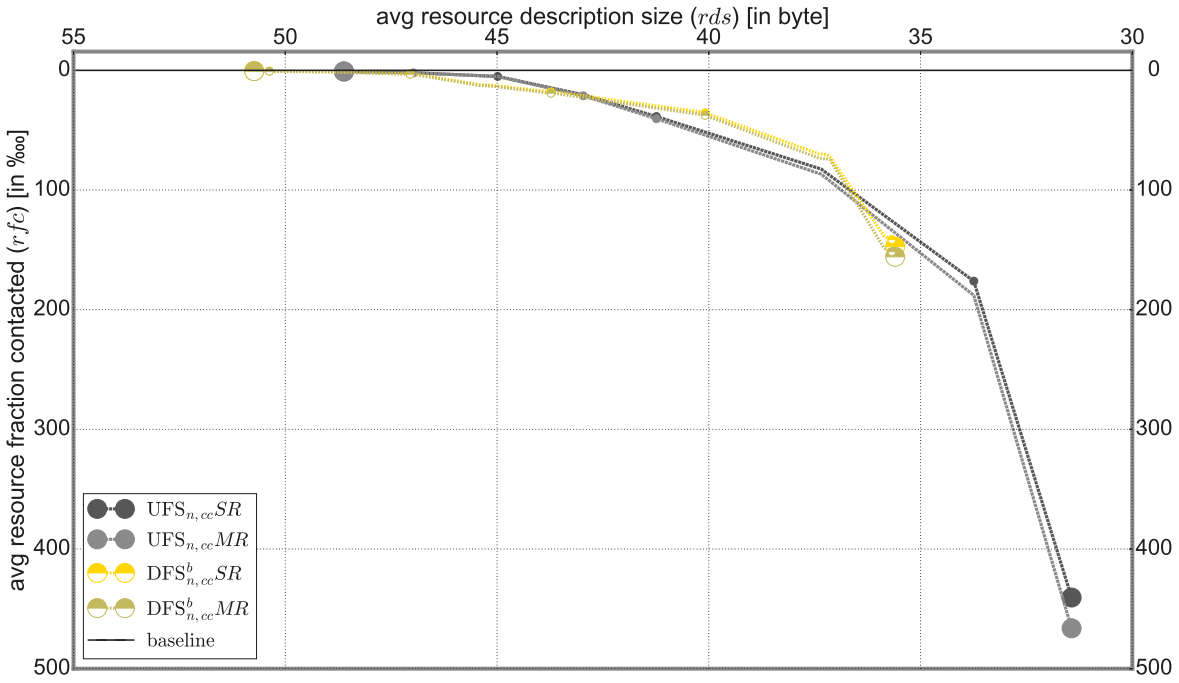


Fig. 40: Full Skylines of the two approaches being *based* on a global Voronoi-like space partitioning. For both $\text{UFS}_{n,cc}$ as well as $\text{DFS}^{b}_{n,cc}$, the results of the spatial domain ranker (SR) as well as the metric-domain-like ranker (MR) are depicted.

The $\text{DFS}^{b}_{n,cc}$ summaries are obviously not designed for very small resulting *rds* values since it is required to capture the quantization base value with 4 B at the beginning of the bit vector. Therefore, the $\text{DFS}^{b}_{n,cc}$ Skyline is behind the $\text{UFS}_{n,cc}$ Skyline for very low *rds* values even though generally, one would expect that a refinement is in particular effective for very

coarse description bases like global space partitions with only very few subspaces. For medium *rds* values, $\text{DFS}^b_{n,cc}$ is superior—which is attributable to the already discussed 'distinctiveness' added to the indexed areas by the resource-individual refinement of occupied subspaces with cluster balls. Nevertheless, the improvement is far smaller compared to the conceptually similar constellation with KD_n and KDMBR^b_n. The reason for this—aside from the 4 B storage overhead for the quantization base value—is that the $\text{DFS}^b_{n,cc}$ refinement (an approximated sphere which is always centered at the site of the Voronoi cell and potentially largely exceeds the cell boundaries[152]) is way coarser and less flexible than the KDMBR^b_n refinement (an approximated rectangle which can be placed freely within the occupied cell's quantization raster). In the long run, for Voronoi-based resource descriptions, it is obviously better to invest additional storage space into additional cells for the global space partition. The selectivity improvements achievable by capturing the r^{out} radii are not large enough to be worthwhile (even though only b bits are invested for refining an occupied cell instead of $4 \cdot b$ bits like for KDMBR^b_n) since the corresponding spheres are too coarse and the amount of global subspaces can be massively increased for the same storage space: for example, $\text{UFS}_{2048,cc}$ requires 44.98 B *rds* (*rfc*: 5.11 ‰) while $\text{DFS}^6_{256,cc}$ requires 45.10 B *rds* (*rfc*: 12.67 ‰).
Comparing the results of the *spatial domain* rankers (*SR* $\rightarrow$ $\text{UFS}_{n,cc}$ SR and $\text{DFS}^b_{n,cc}$ SR) and the respective *metric-domain-like* rankers (*MR* $\rightarrow$ $\text{UFS}_{n,cc}$ MR and $\text{DFS}^b_{n,cc}$ MR)[153], there are no big differences—which was to be expected since the information contained in the resource descriptions as well as their utilization for the pruning of resources in the kNN algorithm are the same, and only the ranking procedure is different. Generally, the differences diminish the larger the amount of subspaces n is. For e.g. $\text{UFS}_{32,cc}$ SR (*rfc*: 440.54 ‰) and $\text{UFS}_{32,16}$ MR (*rfc*: 466.27 ‰), the difference corresponds to ~6,000 contacted resources while for $\text{UFS}_{16384,cc}$ SR (*rfc*: 1.02 ‰) and $\text{UFS}_{16384,16}$ MR (*rfc*: 0.99 ‰), it corresponds to ~7 contacted resources. For both $\text{UFS}_{n,cc}$ and $\text{DFS}^b_{n,cc}$, the SR is always superior to the MR. Generally, the SR is more optimistic since resources are ordered by lower bounds (the MINDISTs between the query point and the specific areas indexed by a resource's summary—which are the smallest possible distances for data points located in these areas). For the MR, the resources are

[152] Note that such an approximated sphere exceeding the Voronoi cell's boundaries does not impair the accuracy of a $\text{DFS}^b_{n,cc}$ summary in comparison to a $\text{UFS}_{n,cc}$ summary since the binary cell occupancy information is also available in the $\text{DFS}^b_{n,cc}$ summary. Nonetheless, this remark shall underline the potential coarseness of the refining spheres which—despite requiring additional storage space—might improve the spatial accuracy of the summary only minimally.

[153] Note that in the following, we use the SR nomenclature when explanations generally apply to both the SR as well as the MR of one of the approaches respectively techniques. For example, $\text{DFS}^6_{16384,cc}$ might be used when an explanation applies to both $\text{DFS}^6_{16384,cc}$ SR as well as $\text{DFS}^6_{16384,16}$ MR.

ordered by site locations. A site's location is the average expectable position of a data point located in the corresponding Voronoi cell. One could imagine that the rather pessimistic ordering of the MR might be a better estimation if there are many fine-grained cells and thus, a resource will only administer very few (mostly one) data points in a single Voronoi cell. In suchlike situations, it is much less probable that there is actually a data point in this cell whose distance to the query point is equivalent to the lower bound distance from the cell to the query point (compared to cases when there are many data points in this cell). Nevertheless, the optimistic ordering of the SR still proves to be superior. For e.g. $\text{DFS}^{6}_{16384,cc}$ SR, the *rfc* value is 0.62 ‰ while for $\text{DFS}^{6}_{16384,16}$ MR, the *rfc* value is 0.68 ‰—corresponding to a difference of ~15 resources. This is also a bigger difference than between $\text{UFS}_{16384,cc}$ SR and $\text{UFS}_{16384,16}$ MR which corresponds to ~7 resources (see *rfc* values specified above in this paragraph)—even though the $\text{DFS}^{6}_{16384,cc}$ *rfc* values (both SR and MR) are clearly better than the $\text{UFS}_{16384,cc}$ *rfc* values.[154]

So, why is the absolute difference between the SR and the MR for $\text{DFS}^{6}_{16384,cc}$ twice as large as for the $\text{UFS}_{16384,cc}$ variants even though the *rfc* values for $\text{DFS}^{6}_{16384,cc}$ are much better? Typically, one would expect smaller improvements when starting from a better base. For an explanation, let us consider the ranking process for the SR: To decide on the ranking between two resources, each resource is represented by an ordered list of `R-Entries` which in turn represent the indexed areas of the resource, sorted in ascending order by their MINDIST to the query point. In the ranking process, both lists (respectively their `R-Entries`) are compared index-wise (see section 5.1). For $\text{DFS}^{b}_{n,cc}$ SR, the ranking decision should usually have been made after comparing the list elements at index 0—since each indexed area (delineated by the globally shared cell polygon *and* the *resource-individual* cluster ball) for $\text{DFS}^{b}_{n,cc}$ (at least most likely) is unique due to the additional r^{out} information. For $\text{UFS}_{n,cc}$ SR, the indexed areas (*only* delineated by the *globally shared* cell polygon) are not unique. Here, e.g. for all resources administering a data point in the query cell, their respective list's first `R-Entry` represents the polygon of the query cell. Hence, for these resources, not only the *single closest* area of each resource (or only the first `R-Entry` of each list) is considered to decide on the top positions of the ranking but also the *few next closest* areas (or the next `R-Entries` of each list). In this regard, the MR ranking algorithm (which for both $\text{UFS}_{n,cc}$ as well as $\text{DFS}^{b}_{n,cc}$ considers *only* binary cell occupencies in the sequence of the ordered list of sites, $L_{q,s}$) operates similar to the $\text{UFS}_{n,cc}$ SR: It oftentimes will also consider several list elements before deciding on the ranking between two resources. Hence, the $\text{UFS}_{n,cc}$ MR and the $\text{DFS}^{b}_{n,cc}$ MR ranking processes are very similar to $\text{UFS}_{n,cc}$ SR ranking process. As outlined,

[154] Note though that the $\text{DFS}^{6}_{16384,cc}$ resource descriptions also require significantly more storage space on average, so it cannot be said that $\text{DFS}^{6}_{16384,cc}$ is superior to $\text{UFS}_{16384,cc}$.

the $\text{DFS}^{b}_{n,cc}$ SR ranking process is different from these—which might be an explanation for the bigger differences between the $\text{DFS}^{6}_{16384,cc}$ SR and MR compared to the $\text{UFS}_{16384,cc}$ SR and MR despite the better baseline of $\text{DFS}^{6}_{16384,cc}$.

Table 9: Key figures related to the Skyline parameterizations for the approaches being based on a global Voronoi-like space partition. The key figures are grouped into the number of Skyline parameterizations (row 2), *rds*-related key figures (rows 3 to 8), and transfer-method-related key figures (rows 9 to 14). For the latter two groups, the given values for the approaches are aggregated over all the respective Skyline parameterizations. Note that for the spatial ranker variants (SR), this is an excerpt from Table 4. The values for the metric-domain-like ranker variants (MR) are newly depicted.

approach → key figure ↓	$\text{UFS}_{n,cc}$ SR	$\text{UFS}_{n,cc}$ MR	$\text{DFS}^{b}_{n,cc}$ SR	$\text{DFS}^{b}_{n,cc}$ MR
non-dominated parameterizations	8 (100%)	8 (30.8%)	24 (100%)	24 (30.8%)
avg *rds* [in byte]	31.4-48.6	31.4-48.6	36.0-50.7	36.0-50.7
min *rds* [in byte]	29-29	29-29	33-33	33-33
25%-quant. *rds* [in byte]	31-36	31-36	35-36	35-36
median *rds* [in byte]	32-44	32-44	36-44	36-44
75%-quant. *rds* [in byte]	32-60	32-60	36-64	36-64
max *rds* [in byte]	33-1,376	33-1,376	48-2,234	48-2,234
sum-z/lq-z [in %]	18.7 (0-37.1)	18.7 (0-37.1)	17.4 (0-34.5)	17.4 (0-34.5)
sum-nz/lq-nz [in %]	38.8 (0.7-100)	38.8 (0.7-100)	31.3 (0.6-72.7)	31.3 (0.6-72.7)
dr-z [in %]	0.09 (0-0.5)	0.09 (0-0.5)	0.2 (0-0.9)	0.2 (0-0.9)
dr-nz [in %]	42.4 (0-65.3)	42.4 (0-65.3)	51.1 (27.2-68.7)	51.1 (27.2-68.7)
cblq-z [in %]	-	-	-	-
cblq-nz [in %]	-	-	-	-

The SR generally does not consider the cc parameter. Hence, one parameter less has to be set and tested for the SR. For $\text{DFS}^{b}_{n,cc}$ SR, the selectivity and the storage consumption of the resource descriptions are dominated by parameter n (as already stated in section 7.1.2), i.e. the number of Voronoi cells. Raising parameter b (even from 3 to 6) never leads to a comparable increase in the *rds* values as raising any n to its next greater tested value. Hence, since there is no 'competition' between both parameters (which would mean that raising one of the parameters would prove to be more efficient than raising the other), *all* tested parameterizations are part of the $\text{DFS}^{b}_{n,cc}$ SR Skyline. For $\text{UFS}_{n,cc}$ SR, *all* tested parameteriza-

tions are also part of the corresponding Skyline which is natural since it is only dependent on parameter n: Varying a single parameter such that the accuracy of the summaries is increased always results in more selective resource descriptions requiring more storage space, so no parameterization can be dominated by another.

For the MR and both approaches, given a fixed value for n, it shows that considering $cc = 16$ sites always leads to the best results—which is convenient with regard to the computational costs: the less sites are considered, the less computational effort is required for the ranking. For $\text{UFS}_{n,cc}$ MR and a fixed value for parameter n, the resulting *rfc* values are always exactly the same—no matter how many sites cc are considered in the ranking process.[155] For $\text{DFS}^b_{n,cc}$ MR, the resulting *rfc* values are slightly better in case the cc parameter is set to 16 (for e.g. $\text{DFS}^6_{16384,16}$, it is 0.684 ‰ while for $\text{DFS}^6_{16384,16384}$, it is 0.686 ‰). In general, it is not reasonable to also consider cell occupations of sites which are far away from the query point in the ranking process. The 50 NN generally are so close that only the immediate neighborhood of the query point is relevant for kNN queries. Therefore, it is comprehensible that low cc values such as $cc = 16$ result in a better ranking than high values such as $cc = 2048$. In fact, our measurements show that also the *relative error*[156] is slightly smaller for lower cc values—namely for both $\text{DFS}^b_{n,cc}$ MR *and* $\text{UFS}_{n,cc}$ MR. For example, when $m = 30$ (i.e. 30 of the 50 closest data points have been found), the *rfc* value for $\text{UFS}_{128,16}$ is 24.45 ‰ while for $\text{UFS}_{128,128}$, it is 25.89 ‰. The differences in the relative error are always relatively small. This implies that parameter cc has not much influence on the ranking—at least after it is set to a 'high enough' value such as 16 (which is the lowest tested value for cc). This is intuitively understandable: For spatially narrow resources, the data points are mostly contained in a single cell while for spatially spread resources, it is very unlikely that their cell occupancies are *exactly the same* for *all* the $cc = 16$ closest sites to the query point. It is all the more unlikely for the 64 (which is the second lowest tested value for *cc*) or even more closest sites. Hence, it is not much of a difference for the ranking if $cc = 16$ or $cc = 64$ sites are considered. Consequently, the differences between the tested values for cc are minimal. Future work could consider evaluating very low values for cc such as 4.

Given $\text{UFS}_{n,cc}$ MR and a fixed number of cells n, the slight differences in the relative error do not lead to different final *rfc* values since for determining these, we measure the resource fraction contacted until the top 50 data points have *definitely* been determined (i.e. *all* resources have been contacted or pruned *safely*). As for $\text{UFS}_{n,cc}$, only the cell polygons of the

[155] Since the cc parameter is only used in the ranking process, the storage-space-related *rds* values are generally not affected by its variation.

[156] For the *relative error*, it is assessed how large the resource fraction contacted is when m of the final k closest data points have been found, $m < k$. A more detailed discussion of the relative error for selected approaches can be found in section 8.7.

globally shared space partition are available for pruning (i.e. when the final query ball intersects a specific cell, *all* resources administering data points in this cell have to be contacted) and additionally, always $n_{rp} = 10$ resources are queried in parallel (see section 5.2), the resulting *rfc* values are the same. For $\text{DFS}^b_{n,cc}$ MR, the resulting numbers differ slightly since in addition, the individual r^{out} radii are used for pruning. All of the parameterizations featuring $cc = 16$ (and not a single parameterization featuring a different cc value)[157] build the Skylines for both $\text{UFS}_{n,cc}$ MR and $\text{DFS}^b_{n,cc}$ MR, yielding eight Skyline parameterizations for $\text{UFS}_{n,cc}$ MR and 24 Skyline parameterizations for $\text{DFS}^b_{n,cc}$ MR. These are the same numbers of Skyline parameterizations as for the SR variants (also see Table 9, which depicts the aggregated key figures for the Skyline parameterizations of the SR and MR variants for completeness once again[158]).
Table 10 shows the key figures for the respective Skyline parameterizations at 'MBRsize'. Note that the information encoded in the resource descriptions is the same for the variants of $\text{UFS}_{n,cc}$ respectively the variants of $\text{DFS}^b_{n,cc}$: the difference between the SR and MR variants is solely how the information is utilized in the ranking process. Hence, the *rds*-related key figures are the same for $\text{UFS}_{n,cc}$ SR and $\text{UFS}_{n,cc}$ MR respectively $\text{DFS}^b_{n,cc}$ SR and $\text{DFS}^b_{n,cc}$ MR.[159] The $\text{DFS}^b_{n,cc}$ parameterizations feature $n = 128$ subspaces and $b = 6$ bit per r^{out} radius while the $\text{UFS}_{n,cc}$ parameterizations feature the double amount of subspaces ($n = 256$). On average, the resource descriptions are greater for $\text{UFS}_{256,cc}$ (41.24 B *rds* versus 40.32 B *rds* for $\text{DFS}^6_{128,cc}$). Interestingly, the '75%-quant. rds'-value is the only descriptive statistic *rds* value which is smaller for $\text{DFS}^6_{128,cc}$—all the other *rds*-related key figures are greater or equal. Nevertheless, in the very end, it results in considerably smaller 'avg rds'-values for $\text{DFS}^6_{128,cc}$. The smallest size of an $\text{DFS}^6_{128,cc}$ resource description is 33 B from which 32 B are the usual 27 B serialization overhead plus 1 B metadata and the 4 B for the quantization base value. The remaining 1 B is the (byte-wise clipped) cell occupancy information including the 6 bits for the refinement of a single cell. The 'max rds'-value of $\text{DFS}^6_{128,cc}$ is more than twice the one of $\text{UFS}_{256,cc}$ (123 B versus 61 B) which shows that the refinement information can account for considerable amounts of storage space even though 'only' (in com-

[157] Note that in case various parameterizations result in exactly the same *rds* and *rfc* values, the parameterization which should theoretically lead to the spatially least accurate resource descriptions and the lowest computational complexity is considered to be part of the Skyline. This has already been noted at the beginning of section 7.1 (where the point of computational complexity was omitted for convenience). For example, if the same *rds* and *rfc* values result for $\text{UFS}_{128,16}$ and $\text{UFS}_{128,128}$, the former is considered to be part of the Skyline.

[158] Note that the key figures for the SR variants have already been listed in Table 4 on page 132.

[159] Mind that in the following, again, in case explanations apply to both the SR and the MR variant, we address these key figures by using the SR nomenclature, e.g. $\text{DFS}^6_{128,cc}$.

Table 10: Comparison of the descriptive statistic *rds* and *rfc* values as well as of the shares for the resource description transmission methods at 'MBRsize' for the approaches being *based* on a global Voronoi-like space partition. For the SR variants, this is an excerpt from Table 5. The values of the MR variants are newly depicted.

technique → key figure ↓	$\text{UFS}_{256,cc}$ SR	$\text{UFS}_{256,16}$ MR	$\text{DFS}^{6}_{128,cc}$ SR	$\text{DFS}^{6}_{128,16}$ MR
avg *rds* [in byte]	41.24	41.24	40.32	40.32
min *rds* [in byte]	29	29	33	33
25%-quant. *rds* [in byte]	36	36	36	36
median *rds* [in byte]	38	38	39	39
75%-quant. *rds* [in byte]	49	49	44	44
max *rds* [in byte]	61	61	123	123
avg *rfc* [in ‰]	38.62	40.48	34.33	36.27
min *rfc* [in ‰]	2.25	4.33	0.04	2.36
25%-quant. *rfc* [in ‰]	20.38	21.64	11.75	14.07
median *rfc* [in ‰]	34.69	35.24	30.13	32.31
75%-quant. *rfc* [in ‰]	53.41	53.50	47.50	49.87
max *rfc* [in ‰]	154.4	154.4	175.0	175.6
sum-z/lq-z [in %]	16.84	16.84	0	0
sum-nz/lq-nz [in %]	34.34	34.34	55.37	55.37
dr-z [in %]	0.04	0.04	0	0
dr-nz [in %]	48.78	48.78	44.63	44.63
cblq-z [in %]	-	-	-	-
cblq-nz [in %]	-	-	-	-

parison to other hybrid approaches) 6 bits are used for the refinement of an occupied subspace.

With regard to the selectivity (i.e. the resulting *rfc*-values in the 'avg rfc'-row), the $\text{DFS}^{6}_{128,cc}$ variants (SR and MR) are both slightly better than the $\text{UFS}_{256,cc}$ variants—despite using only half the amount of subspaces. Hence, for a low number of subspaces[160], the r^{out} radii information—though generally rather inflexible and coarse—are definitely worthwhile. The $\text{DFS}^{6}_{128,cc}$ variants are better for almost all descriptive statistic *rfc* values. Solely the 'max rfc'-value is greater compared to the $\text{UFS}_{256,cc}$ variants. Nevertheless, this shows that also for this low amount of subspaces, there are some queries where the covering radii information of $\text{DFS}^{b}_{n,cc}$ is not beneficial in comparison to the simple $\text{UFS}_{n,cc}$ approach.[161] For all descriptive statistic *rfc* values, the MR is worse or at most equal[162] to the SR (applying

[160]Note that as discussed before, this does not apply to a *very low* number of subspaces since the 4 B overhead for the quantization base must amortize themselves first.

[161]As discussed before, for greater numbers of n, the simple depiction of binary occupency information generally becomes more efficient than additionally depicting the r^{out} radii.

[162]It is equal in a single case—the 'max rfc'-value for $\text{UFS}_{256,cc}$.

to both $\text{UFS}_{256,cc}$ and $\text{DFS}^{6}_{128,cc}$). Hence, the better *rfc*-value of the SR is a result of a consistent superiority and not caused by single outliers.
The entropy of the $\text{DFS}^{6}_{128,cc}$ summaries is increased by the quantization base value and the quantized r^{out} radii information to such an extent that none of them is zipped. For $\text{UFS}_{256,cc}$, 16.84% of the resources transmit zipped summaries. Furthermore, its share of resources transmitting their data points ('dr-z'-value + 'dr-nz'-value) is ~4.2% higher than for $\text{DFS}^{6}_{128,cc}$—both being advantageous for $\text{UFS}_{256,cc}$. Nevertheless, in total, $\text{DFS}^{b}_{n,cc}$ is superior at 'MBRsize'. Figure 40 shows that here, we are in the *rds* range where the occasional advantage of $\text{DFS}^{b}_{n,cc}$ over $\text{UFS}_{n,cc}$ is the greatest.
For the further evaluation, it is clear that the SR is applied since its results are better and additionally, it is specifically designed for the spatial domain anyway. As concrete approach, $\text{UFS}_{n,cc}$ is chosen over $\text{DFS}^{b}_{n,cc}$ as representative for the continued consideration for the following reasons:

— One parameter less has to be set and tested for $\text{UFS}_{n,cc}$.
— $\text{UFS}_{n,cc}$ is computationally cheaper with regard to both the summary creation and the query processing.
— For $\text{UFS}_{n,cc}$, diverse quantitative key figures are easily calculable (such as the 'overlap between summaries' or the 'data space coverage', see section 8.1). This is not the case for $\text{DFS}^{b}_{n,cc}$.
— The performance of both approaches is fairly similar. $\text{UFS}_{n,cc}$ is moderately superior in low and high *rds* ranges while $\text{DFS}^{b}_{n,cc}$ is moderately superior in the mid *rds* range around 'MBRsize'. We proclaimed the range around 'MBRsize' to be the most important one—but then again, in this range, both approaches are far worse than the other approaches selected so far, anyway. Hence, it does not matter as much which one is selected to assess the general suitability of Voronoi-based approaches. It is more important that the quantitative key figures (which can be calculated for $\text{UFS}_{n,cc}$) allow for more insight with respect to why the Voronoi-based approaches are not suited.

7.6. Comparison of the Pure Space Partitioning Approaches and the Improvements Achievable by their Simplest Hybrid Extensions

In the following, we additionally compare the three pure space partitioning approaches (KD_n, $\text{UFS}_{n,cc}$, and $\text{QT}_{c,a}$) and the improvements achievable by their respective simplest hybrid refinements (KDMBR^{b}_{n}, $\text{DFS}^{b}_{n,cc}$, and $\text{QTMBR}^{b}_{c,a}$). Out of the former three, only $\text{UFS}_{n,cc}$ was selected as a representative for the detailed comparison. Nevertheless, a more detailled comparison of especially the pure space partitioning approaches is certainly of great interest.

Both pure global space partitioning approaches (KD_n and $UFS_{n,cc}$) generally show fairly similar performances (see Figure 41). For extremely small *rds* values (below ~33.5 B), KD_n is a bit better—at least in a visual comparison of the Skylines. In fact, the anchor points of the respective Skylines are too far apart to state that KD_n is undoubtedly superior in this *rds* range. The first anchor points are KD_{32} (*rds*: 30.50 B, *rfc*: 539.86 ‰) and KD_{64} (*rds*: 32.65 B, *rfc*: 217.89 ‰) respectively $UFS_{32,cc}$ (*rds*: 31.43 B, *rfc*: 440.54 ‰) and $UFS_{64,cc}$ (*rds*: 33.74 B, *rfc*: 188.08 ‰). Altogether, these four anchor points result in the visual impression that the KD_n Skyline is above the $UFS_{n,cc}$ Skyline—even though the comparable anchor points (e.g. KD_{32} and $UFS_{32,cc}$) are fairly far apart from each other in both dimensions (*rfc* and *rds*).

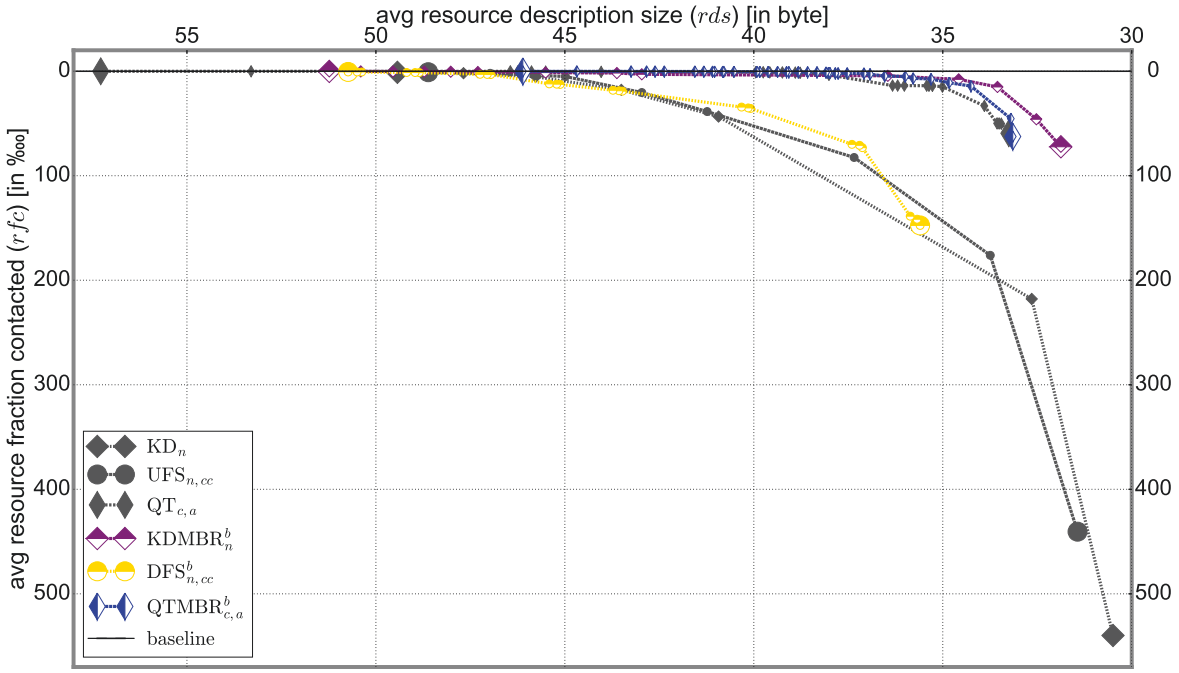


Fig. 41: Full Skylines of the three pure space partitioning approaches and their respective simplest hybrid extensions.

In suchlike situations, the density of the anchor points on the Skylines is too coarse. Ideally, we would require much more anchor points at specifiable *rds* values for each approach—which is not feasible from both a computational as well as a methodological point of view (as it can only be roughly foreseen which *rds* values result when setting the parameters to certain values). As helpful as the Skylines are for assessing the results of the different approaches, they are not perfect and still exhibit some flaws. In general, the Skylines depict the 'power relations' between different approaches very accurately, though. Nevertheless, one has to be careful with conclusions when the respective anchor points are too far apart from each other with respect to the *rds*-dimension since the course of Skylines is not linear but curved. Hence, it is debatable whether KD_n is really better than $UFS_{n,cc}$ for very small *rds* values.

For larger *rds* values, the KD_n Skyline runs slightly but completely below the $UFS_{n,cc}$ Skyline. In this case, it is an accurate depiction of the relative qualities: For e.g. KD_{16384} (*rds*: 49.43 B, *rfc*: 1.10 ‰), a worse selectivity results even though more storage space is used on average in comparison to $UFS_{16384,cc}$ (*rds*: 48.61 B, *rfc*: 1.00 ‰). Hence, $UFS_{16384,cc}$ *dominates* KD_{16384} and is therefore undoubtedly better. Note that the rather big gap between both Skylines for the *rds* range from ~33.5 B to ~41.0 B is caused by that the KD_n Skyline misses the parameterization with $n = 128$—which is tested for $UFS_{n,cc}$ but not for KD_n. It can be expected that the KD_n Skyline and the $UFS_{n,cc}$ Skyline would be very close to each other in this range if KD_{128} would have been tested. Once again, this shall be a reminder to assess the results displayed by the Skylines with care. Nevertheless, in total, $UFS_{n,cc}$ can certainly be considered as slightly superior.
For a very low number of subspaces, $UFS_{n,cc}$ is possibly susceptible due to the selection procedure for the sites: Since *each* of the randomly selected sites creates its own subspace, for e.g. $UFS_{32,cc}$, the entire space partition is dependent on only 32 random data points. In contrast, for KD_n, a configurable bucket overflow approach is applied for determining the space partition (in terms of bucket capacity and hence in the amount of training data points utilized). Hence, for e.g. KD_{32}, the space partition is still dependent on several thousand data points (and therefore should be adjusted more accurately to the data collection's data point distribution) while for $UFS_{32,cc}$, single unfavorably selected sites might easily lead to a suboptimal space partition since the number of considered data points is so low. This might explain why KD_n, in relative terms, seems to perform comparatively better for a very low number of subspaces.
With regard to the superiority of $UFS_{n,cc}$ for many subspaces, there are two feasible explanations:

— The arbitrary polygonal shape of the Voronoi cells results in a more suitable space partition than the rectangular shape of the k-d cells.
— The procedure of building subspaces for $UFS_{n,cc}$ (*each* randomly selected data point has its own subspace) is more suitable than the bucket-overflow-based approach for KD_n (which considers a set of data points before creating a new subspace, minimizing the chance that single 'outlier' data points lead to the creation of subspaces). Possibly, few 'outlier' subspaces are beneficial for the overall selectivity.

Additionally, it is noteworthy that the *rds* values for the KD_n resource descriptions are smaller than for the corresponding $UFS_{n,cc}$ resource descriptions until $n = 256$. From $n = 512$ on, the *rds* values for the KD_n resource descriptions are greater, though. Generally, lower *rds* values might even be a hint that a concrete global space partition *fits the data collection better* than a comparable other space partition resulting in higher *rds* values: Fewer bits are set to '1' in the summaries and therefore, the resources' spatial footprints as a whole are less dispersed over the subspaces of the global

space partition. For a final clarification of the observed phenomenona, additional experiments would be required—from which we refrain at this point.

The local space partitioning approach $\text{QT}_{c,a}$ is vastly superior to the global approaches in all *rds* ranges. For high *rds* values, the superiority naturally declines a little but is still absolutely significant. For example, $\text{QT}_{8192,1.0E-5}$ (*rds*: 46.44 B, *rfc*: 0.143 ‰—corresponding to ~35.6 resources) still is ~7.7 respectively 7 times more selective than the aforementioned KD_{16384} respectively $\text{UFS}_{16384,cc}$—despite requiring significantly less storage space on average. The reasons behind this have already been discussed: the number of resources in the T1 collection is 2,491,785. Even with 16,384 subspaces and the (totally unrealistic) assumption that all the data points of each resource are maintained in a single subspace of the space partition, still ~152 resources would share the same subspace with each other on average. This means their indexed areas are exactly the same—resulting in a huge total amount of spatial overlap between the resource summaries which impedes the distinction between relevant and irrelevant resources.[163] When querying, at least all the summary-represented resources administering data points in the query cell (i.e. the space partition's cell the query point is located in) have to be contacted: They can never be pruned since the query cell (which they all share as their spatial description) overlaps the query point, i.e. all these resources' MINDISTs to the query point are 0. Since there are also many resource administering data points in *more* than a single subspace, there is too much overlap between the resource descriptions overall to achieve *rfc* values comparable to $\text{QT}_{8192,1.0E-5}$. For $\text{QT}_{c,a}$, the space partition is *individually adapted* to the resources' data points—resulting in less overlap and in particular a much lower amount of *exactly congruent* indexed areas for different resources (though due to the application of a regular space decomposition, there are certainly cases in which two or more different resources are described by exactly the same indexed area(s)).

In total, local space partitioning is vastly superior to global space partitioning for pure space partitioning approaches and the T1 collection. But how do results change when the refinement of hybrid approaches is added, i.e. a greater degree of distinctiveness with regard to the indexed areas is awarded especially to the approaches *based* on global space partitioning? For $\text{DFS}^{b}_{n,cc}$ and KDMBR^{b}_{n}, the respective 'starting base' is almost equivalent in performance ($\text{UFS}_{n,cc}$ respectively KD_{n}). Nevertheless, the quan-

[163] Concrete numbers for the overlap between the resource summaries of specific $\text{UFS}_{n,cc}$ parameterizations and the T1 collection are presented in section 8.1. Of course, it has to be taken into account that a lot of resources select the direct representation instead of a summary, i.e. they are described by the exact locations and not by approximated areas. For e.g. $\text{UFS}_{16384,cc}$, only ~34.23% of the resources transmit summary data. Hence, under the unrealistic assumption that each resource administers its data points in a single cell, still $(2,491,785 \cdot 0.3423/16,384) = \sim 52$ resources would share a cell with each other on average.

tized MBRs of KDMBR^b_n prove to be much better suited for refinement than the quantized 'cluster balls' of $\text{DFS}^b_{n,cc}$. This has also been discussed rudimentarily in previous sections: Spheres or balls are a much coarser description than MBRs for most data point sets. Due to the additional fixation of a ball's center at the corresponding site's location, the $\text{DFS}^b_{n,cc}$ refinement is very inflexible compared to the KDMBR^b_n refinement whose MBR can be placed freely within the quantization grid of the occupied cell. Finally, $\text{DFS}^b_{n,cc}$ also requires 4 B storage space overhead for the quantization base value. The sole advantage of $\text{DFS}^b_{n,cc}$ is that the refinement of an occupied subspace requires only b bits instead of $4 \cdot b$ bits. Nevertheless, this alone can by no means compensate for the disadvantages. Hence, despite a similar starting base, KDMBR^b_n is vastly superior to $\text{DFS}^b_{n,cc}$ for the entire *rds* range. Consequently, $\text{DFS}^b_{n,cc}$ improves $\text{UFS}_{n,cc}$ only slightly for *a part* of the *rds* range whilst KDMBR^b_n is a big improvement over KD_n for the *entire rds* range. Obviously, the quantized MBRs are sufficient to overcome large portions of the overlap problems for global space partitioning bases. The benefit of combining a global space partition with quantized MBRs also shows in comparison to $\text{QTMBR}^b_{c,a}$: Even though the $\text{QT}_{c,a}$ starting base for $\text{QTMBR}^b_{c,a}$ is better by a large margin, the Skylines of $\text{QTMBR}^b_{c,a}$ and KDMBR^b_n are fairly close to each other. For small *rds* values, KDMBR^b_n is even better than $\text{QTMBR}^b_{c,a}$. From ~36.5 B on, $\text{QTMBR}^b_{c,a}$ is better—still significantly, but by far not as much as $\text{QT}_{c,a}$ is better compared to KD_n. Since both $\text{QTMBR}^b_{c,a}$ and KDMBR^b_n are selected for further evaluation, a more detailed comparison can be found in the following sections. Overall, $\text{QTMBR}^b_{c,a}$ is only a small improvement over $\text{QT}_{c,a}$ (which was the maximum to be expected since the results for $\text{QT}_{c,a}$ are already very good). Nevertheless, it is still a consistent improvement over the entire *rds* range—which e.g. $\text{DFS}^b_{n,cc}$ did not manage for $\text{UFS}_{n,cc}$. Overall, the quantized MBRs for refinement prove to be beneficial in general whereas the r^{out} radii information are of little value for *spatial* resource description problems.[164]
Table 11 shows the six approaches for their respective Skyline parameterizations at 'MBRsize' once again. The occuring differences between the pure space partitioning approaches and their corresponding hybrid refinements have already been discussed in section 7.3 to section 7.5. In general, it is evident that also at 'MBRsize', the quadtree-based approaches show a lower tendency to zip summaries as well as to directly represent resources (see 'sum-z/lq-z'- to 'cblq-nz'-rows)—both being rather disadvantageous when evaluating for storage space efficiency and selectivity. Nevertheless, both $\text{QT}_{8192,0.001}$ and $\text{QTMBR}^6_{512,0.05}$ are significantly better compared to even KDMBR^6_{256} (by a factor of ~3.2 respectively ~7.6)—not to men-

[164]The case is different in the metric domain where only distance information is available. There, utilizing the r^{out} radii information (which corresponds to $\text{DFS}^b_{n,cc}$) leads to significant selectivity improvements compared to the use of only binary occupancy information (which corresponds to $\text{UFS}_{n,cc}$). See [Blank 2015, p. 126ff].

Table 11: Comparison of the descriptive statistic *rds* and *rfc* values as well as of the shares for the resource description transmission methods at 'MBRsize' for the three pure space partitioning approaches and their respective simplest hybrid extension (excerpt from Table 5).

technique → / key figure ↓	KD_{256}	$UFS_{256,cc}$	$QT_{8192,0.001}$	$KDMBR^{6}_{256}$	$DFS^{6}_{128,cc}$	$QTMBR^{6}_{512,0.05}$
avg *rds* [in byte]	40.93	41.24	39.95	42.96	40.32	41.04
min *rds* [in byte]	29	29	34	31	33	36
25%-quant. *rds* [in byte]	36	36	36	36	36	36
median *rds* [in byte]	37	38	37	40	39	38
75%-quant. *rds* [in byte]	47	49	41	51	44	45
max *rds* [in byte]	61	61	1,948	565	123	1,154
avg *rfc* [in ‰]	43.41	38.62	0.92	2.97	34.33	0.39
min *rfc* [in ‰]	6.22	2.25	0.04	0.09	0.04	0.04
25%-quant. *rfc* [in ‰]	27.23	20.38	0.37	1.03	11.75	0.16
median *rfc* [in ‰]	38.39	34.69	0.73	1.56	30.13	0.25
75%-quant. *rfc* [in ‰]	49.24	53.41	1.12	3.02	47.50	0.51
max *rfc* [in ‰]	202.8	154.4	4.36	24.36	175.0	2.07
sum-z/lq-z [in %]	12.76	16.84	0	11.81	0	0
sum-nz/lq-nz [in %]	44.09	34.34	50.29	39.95	55.37	52.35
dr-z [in %]	0	0.04	0	$\ll 0.01$	0	$\ll 0.01$
dr-nz [in %]	43.15	48.78	36.20	48.24	44.63	42.00
cblq-z [in %]	-	-	0.01	-	-	$\ll 0.01$
cblq-nz [in %]	-	-	13.50	-	-	5.65

tion the remaining approaches which are much worse than $KDMBR^{6}_{256}$. Concerning the descriptive statistic *rds* values of the pure space partitioning approaches ('avg rds'- to 'max rds'-rows in Table 11), it shows that the *rds* value of $QT_{8192,0.001}$ is noticeably smaller than the corresponding values of KD_{256} and $UFS_{256,cc}$ (see 'avg rds'-row)—even though the occuring minimum resource description size ('min rds'-row, listing 34 B for $QT_{8192,0.001}$ versus 29 B for both KD_{256} and $UFS_{256,cc}$) as well as the occuring maximum resource description size ('max rds'-value, 1,948 B versus 61 B) are distinctly bigger for $QT_{8192,0.001}$. Generally, of the descriptive statistic *rds* values, only the '75%-quant. rds'-value is lower for $QT_{8192,0.001}$ (41 B versus 47 B/49 B). Nevertheless, this results in a quite significant lower average resource description size (see 'avg rds'-row). Therefore, it can be concluded that not many resource descriptions of KD_{256} and $UFS_{256,cc}$ are clipped to the absolute theoretical minimum of 29 B. On the other hand, also the extreme scaling of the occuring maximal resource description size for $QT_{8192,0.001}$ (see 'max rds'-value of 1,948 B) is clearly restricted to rare cases which are compensated by the storage space economy of the majority of the $QT_{8192,0.001}$ resource descriptions. Hence, all these few outliers have no significant consequences on the resulting *rds* values.
With respect to the descriptive statistic *rfc* values ('avg rfc'- to 'max rfc'-rows in Table 11), the resulting *rfc* values ('avg rfc'-row) are well reflected

in the other values once again. Consequently, it can be stated that e.g. $\mathrm{QT}_{8192,0.001}$ is consistently superior to $\mathrm{UFS}_{256,cc}$. For the pure space partitioning approaches, the only exception for which the other descriptive statistic *rfc* values do not correspond to the average *rfc* value is the '75%-quant. rfc'-value for KD_{256} and $\mathrm{UFS}_{256,cc}$—which is negligibly better for KD_{256} (49.24 ‰ for KD_{256} versus 53.41 ‰ for $\mathrm{UFS}_{256,cc}$) even though on average, $\mathrm{UFS}_{256,cc}$ is slightly better (43.41 ‰ for KD_{256} versus 38.62 ‰ for $\mathrm{UFS}_{256,cc}$). But then again, these '75%-quant. rfc'-values do not differ by much and for all the other descriptive statistic *rfc* values, $\mathrm{UFS}_{256,cc}$ is better. Thus, it cannot be claimed that the *rfc* value does not reflect the general performance difference between both approaches. Consequently, it is valid to state that in most cases and in total, $\mathrm{UFS}_{256,cc}$ offers slightly better selectivity than KD_{256}.

8. IN-DEPTH EVALUATION OF THE SELECTED APPROACHES

In this section, the approaches selected for further evaluation are compared in detail. For this purpose, they are examined for different data collections and different underlying conditions. For example, the permitted transmission methods for the resource descriptions or the selection procedure of the query points are varied. Furthermore, the approaches are considered on the basis of concrete quantitative key figures as well as by qualitative analyses. The main focus in this section is not to discuss *which* approach is best suited in the given scenario (which in principle can be immediately seen from the Skylines) but to elaborate *why* certain approaches are better and which changes occur when the conditions of the search scenario are varied.

In general, the quality of an approach manifests itself in the selectivity achievable for a given average resource description size, i.e. the resulting *rfc* value for a given *rds* value. There are several obvious influencing factors on the selectivity:

— **overlap between summaries**: The total overlap between the (spatial footprints) of all the *summary-represented* resources. Ideally, the areas indexed by the resource summaries are not only delineated as tightly as possible but also mutually non-overlapping or 'unique', i.e. no two or more resource summaries cover the same subarea(s) of the data space. Then, the ranking algorithm can decide on a ranking between the resources with maximum precision. Of course, it is not possible to describe absolutely unique areas without any overlap for the whole data collection: In specific regions of interest (such as major cities), there is always some overlap between the summaries as the resources' data point clouds are strongly blended. Nevertheless, minimizing the inevitable overlap between the resource summaries can help to provide more precise rankings.

  For determining the overlap between the resource summaries, the indexed areas of the resources are reconstructed and for every resource, the overlap of its area(s) with the area(s) of every other resource (i.e. the 'inter-resource' overlap) is calculated.[165] Since for all approaches examined in this section, the indexed areas of a *single resource* do not overlap each other, a discussion on the necessity of calculating the 'intra-resource' overlap of the resource summaries and its consideration with regard to the total overlap key figure is dispensible.[166] The measurement unit for the overlap between the summaries is the surface area in the

[165] Of course, the overlap between two specific resources res_a and res_b is calculated only once.

[166] Note that for the MAR-utilizing approaches (RecMAR$_{k,sl}$, KDMAR$_{n}^{b,k}$, QTMAR$_{c,a}^{b,k}$, and MARQT$_{k,sl}^{c,a}$), an intra-resource overlap can occur.

x/y-coordinate space, i.e. for example a quadratic overlap of 10 *dsu* in the x-dimension and 10 *dsu* in the y-dimension results in an overlap area of 100 dsu^2.

— **data space coverage**: The number of times the data space can be covered by the sum total of the surface areas indexed by the resource summaries. The smaller the 'data space coverage', the less dead space is contained in the areas indexed by the summaries. As discussed before (see e.g. section 2.4), minimizing the dead space greatly facilitates the distinction between potentially relevant and irrelevant resources. Therefore, a small 'data space coverage' should lead to better results. In general, the 'data space coverage' should correlate strongly with the 'overlap between summaries', i.e. the bigger the 'data space coverage', the more likely the single resource summaries will also overlap each other.
For the determination of the 'data space coverage', the total surface area indexed by the resource summaries is calculated first (in dsu^2). Afterwards, this sum is divided by the size of the data space[167], resulting in the numerical value for the 'data space coverage'.

— **surface area per summary**: The surface area which is described by a summary *on average*. Both the 'overlap between summaries' as well as the 'data space coverage' are key figures accumulated over the *entirety* of the resource summaries. In these key figures, it is not depicted that a varying amount of resources transmits the coordinates of their data points instead of a summary—hence, the number of resource summaries contributing to both key figures is different for each concrete technique.[168] The smaller the 'surface area per summary', the more competitive the corresponding approach should be: the incorporated dead space is minimized as well as the probability of resource summaries overlapping each other. However, when regarding this key figure, the share of directly represented resources must be kept in mind since the total indexed surface area is also strongly dependent on how many summary-represented resources exist for a specific technique.
For the determination of the 'surface area per summary', the sum total of the indexed surface area for all summaries is calculated in dsu^2 and divided by the amount of resource summaries afterwards.

— **aspect ratio of the boxes**: The aspect ratio of a geometric shape is the ratio of its sizes in different dimensions.[169] We do not calculate the aspect ratio for $\text{UFS}_{n,cc}$ since for arbitrary polygons, the resulting values are not easily interpretable. For the approaches describing rectangular

[167] The size of the data space is its height multiplied with its width, i.e. $[90-(-90)]$ *dsu* $\cdot[180-(-180)]$ *dsu* $= 64,800$ dsu^2.

[168] Of course, the average indexed surface area per resource summary can be determined by knowledge of the data space coverage and the share of resources transmitting summaries. Nevertheless, we explicitly include the 'surface area per summary'-values in the respective tables for transparency and convenience.

[169] See https://en.wikipedia.org/wiki/Aspect_ratio, last visit: 23.11.2018.

areas, the aspect ratio is assessed. Here, the aspect ratio is the ratio of the length of the greater side to the length of the shorter side. It can be interpreted more clearly and is relevant when evaluating for the selectivity of an approach: As described in section 5.2, the kNN queries are implemented as range queries with decreasing query radius. Consequently, the ideal shape of the boxes describing the areas containing data points is quadratic, i.e. most similar to the shape of the 'query balls'. Hence, the ideal aspect ratio is 1:1. Thin, elongated boxes with a large aspect ratio do not cover a lot of space but are often unnecessarily intersected by the query ball due to their large extent in one dimension. Therefore, they impede the early pruning of irrelevant resources.[170]
For the aspect ratio, we depict both the mean as well as the median ratio of the boxes for the different rectangle-describing approaches. The median aspect ratio is included since few very elongated boxes (as outliers) may have great effect on the resulting average values. In order to determine the aspect ratio of a box, the length of the greater side (in *dsu*) is divided by the length of the shorter side (in *dsu*). The boxes featuring an infinite aspect ratio (i.e. boxes which in fact are vertical or horizontal lines) are excluded when determining the average values. The resulting values are not substantially distorted by this procedure: boxes with an infinite aspect ratio only occur in seldom cases—even for the MBR approach as the approach with the *by far* highest tendency of describing suchlike lines.[171] For boxes describing points, an aspect ratio of 1:1 is recorded. For determining the mean and median values of the aspect ratios occuring for a technique, we use the Apache Commons Math library (version 3.4.1) once again.

Besides the listed key figures, we also include the total number of areas which are indexed by the entirety of the resource summaries for a technique in the comparative tables. Note that this 'number of indexed areas' is interesting rather for comparing the costs of query processing (or at least for the initial ranking) rather than drawing conclusions with regard to the selectivity of the different approaches. It is determined by simply accumulating the number of areas constituting the spatial footprint of the single resources. For the MBR approach, the total number of indexed areas matches the number of resources transmitting summaries since each MBR summary describes exactly one area: the full-precision MBR. For the other approaches, potentially several areas might be indexed by the summary of

[170] Note that in case unfavorable aspect ratios result, at least for the k-d-based approaches, an adaption of the split strategy would be possible. Also for quadtree-based approaches, reasonable adjustments would certainly be specifiable.

[171] For the MBR approach, it is 0.7% of the resources of the `T1` collection and 1.11% of the resources of the `F` collection for which their MBR summaries describe rectangles with an infinite aspect ratio. For all the randomly checked parameterizations of the other approaches, there were less than 3 resource summaries describing any rectangle with an infinite aspect ratio (for both the `T1` and the `F` collection).

a resource, i.e. the total 'number of indexed areas' can be (much) higher for these.
Furthermore, also the share of 'directly represented resources' is included which—as just discussed—is important to bear in mind when assessing the 'overlap between summaries' and the 'data space coverage'. Additionally, it also affects the resulting *rfc* values since the exact coordinates of a resource's data points are a 'perfect' description of the resource's spatial footprint. Therefore (as already explained in section 6.4.1), the directly represented resources are not ranked but only contacted in case they actually contribute to the final query result—whereas summary-represented resources contributing to the result are contacted twice in total: once for their potentially relevant data points in the ranking process, once for the relevant media item(s) after the final kNN have been determined. Consequently, there is no 'risk' of unnecessarily contacting irrelevant directly represented resources during query processing as they are not ranked. All in all, a high share of directly represented resources is beneficial for both the *rds* and *rfc* values. Nevertheless, it requires some investigation on how great the impact of the direct representation really is, and if the query results are not nonetheless dominated by the summary-represented resources: They usually administer many more data points than directly represented resources and therefore have a higher probability of contributing to the final query result.
Finally, also the resulting *rds* and *rfc* values are included in the comparison tables for each technique.
In the following, first, we further evaluate the T1 collection by assessing the listed key figures (section 8.1). Since the determination of these key figures requires concrete parameterizations, we start with comparing the approaches for their respective parameterizations at 'MBRsize'. For evaluating the occurring changes in the 'balance of power' between the approaches, they are additionally compared for significantly smaller *rds* values (at 'MBRsize-') as well as significantly bigger *rds* values (at 'MBRsize+'). Afterwards, the F collection is evaluated in the same way to see if the results for the T1 collection can be confirmed for other long tail distributions of data points to resources (section 8.2). Subsequently, the underlying conditions are varied. In section 8.3 and section 8.4, the T1 and F collection are evaluated while disallowing the direct representation of resources (i.e. all resources are described by a summary). In section 8.5, the F collection is assessed when altering the procedure of selecting the query points. In section 8.6, we evaluate the T2 collection. In section 8.7, the approaches are assessed for approximate similarity search. An excursus that includes the evaluation of additional properties is presented in section 8.8. Finally, we summarize the key results for the distributed application scenario in section 8.9.

8.1. Further Evaluation of the T1 Collection

In this section, the six approaches selected for the in-depth comparison are further evaluated for the T1 collection. This time, we take a look at several specific techniques, the key figures just discussed, and conduct some qualitative analyses.

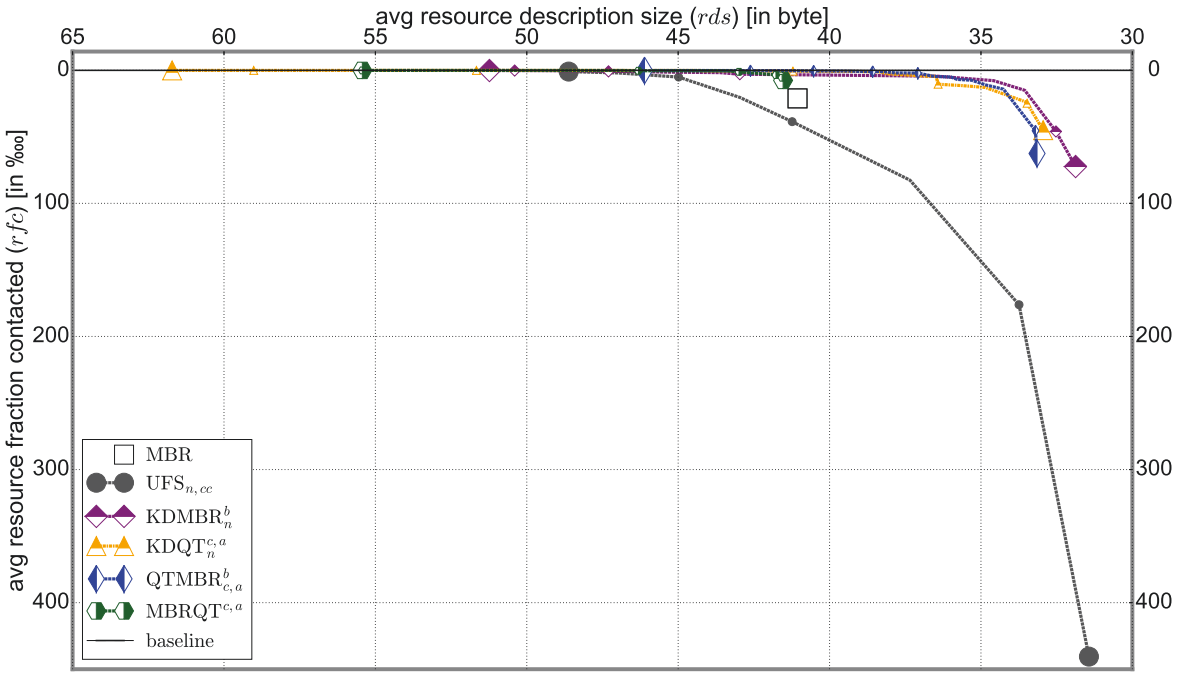


Fig. 42: Full Skylines of the six approaches selected for further evaluation and the T1 collection.

In Figure 42, the Skylines of these six approaches are depicted once again.[172] Let us quickly recapitulate the key results which are apparent from the Skylines: UFS$_{n,cc}$ is far behind the other approaches for most of the *rds* range and is even worse than the MBR approach for comparable *rds* values. By nature, the MBRQT^{c,a} approach is selectivity- and storage-space-bound by its exterior MBR. Therefore, its Skyline sets in late and at rather high *rfc* values. In its very early phase, MBRQT^{c,a} is significantly worse than the other approaches except UFS$_{n,cc}$ but rapidly overtakes KDMBR$^{b}_{n}$ (at ~41.9 B *rds*) and soon converges towards the other quadtree-utilizing approaches (at ~43.5 B *rds*).[173] KDMBR$^{b}_{n}$ is best for very small *rds* values (up until ~36.0 B *rds*) but is then overtaken by both QTMBR$^{b}_{c,a}$ and KDQT$^{c,a}_{n}$. Between ~34.0 B *rds* and ~42.0 B *rds*, QTMBR$^{b}_{c,a}$ is superior to KDQT$^{c,a}_{n}$ which is better by the thinnest margin afterwards. By assessing the Skylines, QTMBR$^{b}_{c,a}$ is the most promising approach overall, followed by KDQT$^{c,a}_{n}$ and KDMBR$^{b}_{n}$.

[172] Note that Figure 42 is the same as Figure 33 on page 128 just without the discarded approaches.

[173] Also already see Figure 43 for a zoomed in depiction of Figure 42.

In Table 12, the key figures discussed in the previous section are displayed for the Skyline parameterizations of the respective approaches at 'MBR-size'. Note that we do not focus on the differences in the *rds* values and regard them as equivalent for the following evaluation: the main purpose is to find out *why* some approaches are better than others. To support the analysis, we always create rankings for the examined techniques with regard to five of the key figures. The key figures for which respective rankings are created are: 'overlap between summaries', 'data space coverage', 'surface area per summary', 'aspect ratio (mean)', and 'aspect ratio (median)'. For each ranking, the best rank is assigned to the technique featuring the most advantageous value while the worst rank is assigned to the one with the most disadvantageous value. In the key figure tables, color codes are assigned for an easy assessment of the ranks. The color codes range from from dark blue for the backmost rank to light blue for the foremost rank. The ranks in between are colored with corresponding gradations. For example, for the 'data space coverage' in Table 12, the ranking is as follows:

1. $\text{QTMBR}^{6}_{512,0.05}$ (with a value of 0.34)
2. $\text{KDQT}^{64,1.0E-5}_{32}$ (with a value of 3.51)
3. KDMBR^{6}_{256} (with a value of 11.36)
4. $\text{MBRQT}^{16,1.0}$ (with a value of 58.90)
5. MBR (with a value of 412.3)
6. $\text{UFS}_{256,cc}$ (with a value of 6,994.6)

The color codes are also used to indicate the ranks of the techniques with regard to their *rfc* values. For the 'number of indexed areas' (rather relevant for assessing the costs of the initial ranking) as well as the *rds* values (considered to be equivalent here), we omit from awarding ranks. The same goes for the share of directly represented resources whose impact on the resulting *rfc* values still has to be figured out at this point of the evaluation.

8.1.1. RESULTS AT 'MBRSIZE'. In this subsection, we start our detailed analysis by evaluating the approaches at 'MBRsize'. See Figure 43 for a depiction of the concrete techniques which are compared. Table 12 shows that the values of the area-related key figures ('overlap between summaries', 'data space coverage', and 'surface area per summary') are the greatest for $\text{UFS}_{256,cc}$ by far. This is due to the global space partition: In general, there are only n different Voronoi cell polygons which are available for describing the spatial footprints of summary-represented resources for $\text{UFS}_{n,cc}$. Hence, this polygon set is 'shared' by all the resources. Consequently, in case two summary-represented resources administer data points in the same Voronoi cell, the whole surface area of the cell's polygon is automatically the overlap between these two resources' summaries. As there are only 256 global subspaces for $\text{UFS}_{256,cc}$, the average size of a cell is very large ($64,800\ dsu^2\ /256\ =\ \sim 253.1\ dsu^2$). Additionally, summary-

represented resources oftentimes administer data points in more than one cell of the global space partition.[174] Therefore, the extremely high values for all three area-related key figures are no surprise. As $UFS_{256,cc}$ also features the worst *rfc* value, this—for the moment—confirms the intuitive assumption that the area-related key figures are correlated with the resulting *rfc* values—at least to some extent. For the aspect ratios, no value has been determined for $UFS_{256,cc}$ due to the arbitrary shape of the Voronoi cell polygons (as already noted).[175]

Table 12: Listing of the key figures alongside the resulting *rds* and *rfc* values for the T1 collection and the benchmarked techniques at 'MBRsize'.

technique → / key figure ↓	MBR	$UFS_{256,cc}$	$KDMBR^{6}_{256}$	$KDQT^{64,1.0E-5}_{32}$	$QTMBR^{6}_{512,0.05}$	$MBRQT^{16,1.0}$
overlap between summaries [in dsu^2]	2.40E+10	2.92E+12	2.20E+8	1.02E+6	81,192	6.18E+8
data space coverage	412.3	6,994.6	11.36	3.51	0.34	58.90
surface area per summary [in dsu^2]	22.50	334.84	0.57	0.14	0.02	3.21
number of indexed areas	1,187,747	1,739,402	1,617,235	4,947,405	2,760,692	1,363,445
aspect ratio (mean) [:1]	6.10	-	1.92	1.54	2.47	5.00
aspect ratio (median) [:1]	2.00	-	1.83	2.00	2.00	1.87
directly represented resources [in %]	52.33	48.82	48.24	34.21	42.00	52.33
avg *rfc* [in ‰]	21.13	38.62	2.97	0.65	0.39	7.62
avg *rds* [in byte]	41.06	41.24	42.96	41.21	41.04	41.53

For the remaining techniques, the by far biggest values for the area-related key figures are exhibited by the MBR approach—which was to be expected due to the coarseness of the MBR. Notably, the MBR-interior quadtrees of $MBRQT^{16,1.0}$ greatly reduce these three key figures (by factors of 38.9/7.0/7.0)[176] even though only 175,698 or 14.8% additional areas are indexed in comparison to the MBR approach (the share of directly repre-

[174] Since 48.82% of the resources are directly represented and in total, there are 1,739,402 indexed areas for $UFS_{256,cc}$ and the T1 collection, an $UFS_{256,cc}$ summary describes $1,739,402/(2,491,785 \cdot (1-0.4882)) = 1.36$ areas on average.

[175] Note that since no aspect ratios are determined for Voronoi cell polygons, we always assign the backmost rank for the aspect ratio key figures to the $UFS_{n,cc}$ variant in these comparisons.

[176] The given values result as e.g. $2.4E+10/6.18E+8 = \sim 38.9$.

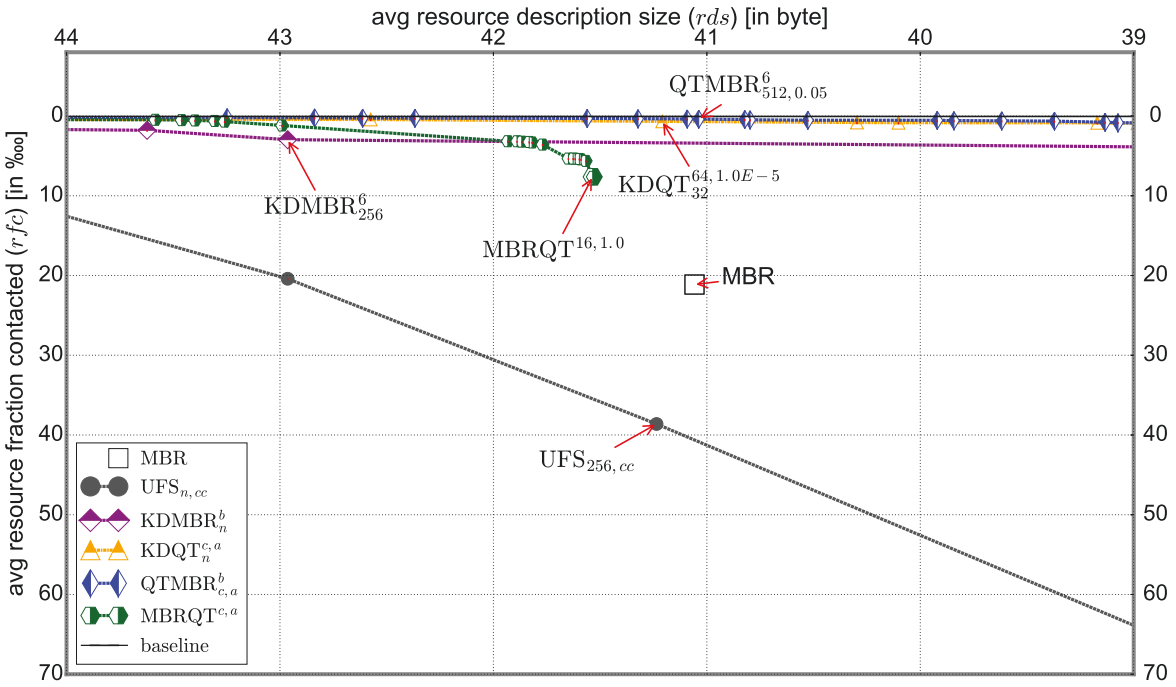


Fig. 43: Zoomed in version of Figure 42, showing the Skylines for the T1 collection around 'MBRsize'. The respective Skyline parameterizations of the different approaches at 'MBRsize' are marked with red arrows.

sented resources is the same for both techniques, 52.33%).[177] This means that even though only rather few exterior MBRs are refined by internal quadtrees for MBRQT$^{16,1.0}$, the quadtree refinement—even with a maximum of only $c = 16$ cells—can reduce the surface areas indexed by the summaries greatly. It has to be noted, though, that MBRQT$^{16,1.0}$ is still far behind KDMBR$^{6}_{256}$, KDQT$^{64,1.0E-5}_{32}$, and QTMBR$^{6}_{512,0.05}$ for these key figures. As rather small internal quadtrees are built in rather few cases, the obviously great capability of quadtrees to minimize the indexed surface areas is not fully utilized here. This great capability of the quadtrees also shows for both KDQT$^{64,1.0E-5}_{32}$ and QTMBR$^{6}_{512,0.05}$, especially in comparison to KDMBR$^{6}_{256}$: The former two both exhibit much lower values for the three area-related key figures than KDMBR$^{6}_{256}$—even though the share of directly represented resources is significantly greater for the latter (48.24% versus 34.21% for KDQT$^{64,1.0E-5}_{32}$ respectively 42.00% for QTMBR$^{6}_{512,0.05}$) which is advantageous with regard to the two key figures which are accumulated ('overlap between summaries' and 'data space coverage'). Furthermore, the k-d space partition of KDQT$^{64,1.0E-5}_{32}$ only features 32 cells (as opposed to the 256 cells of KDMBR$^{6}_{256}$), i.e. the 'description base' is much

[177] Note that in our case, the description of a greater amount of areas is rather an indication of a more accurate delineation of a resource's spatial footprint: This is because a single large area is usually divided into several smaller areas which in sum cover a much smaller surface. Therefore, the higher the amount of indexed areas, the less 'data space coverage' can be expected in tendency which might seem counterintuitive at first thought.

coarser for $\text{KDQT}_{32}^{64,1.0E-5}$ compared to KDMBR_{256}^{6}. The minimization of the indexed surface areas proves to be especially effective for $\text{QTMBR}_{512,0.05}^{6}$, outclassing $\text{KDQT}_{32}^{64,1.0E-5}$ by factors of 12.6/10.3/7.0 for the area-related key figures—even though the 'number of indexed areas' of $\text{KDQT}_{32}^{64,1.0E-5}$ is 1.8 times greater (implying a more fine-grained description of the spatial footprints). Generally, the techniques' ranks for the three area-related key figures in Table 12 correspond to each other and also coincide with the ranks for the *rfc* values: the less 'overlap between summaries'/'data space coverage'/'surface area per summary', the more selective the respective technique.

Notably, the dominance of $\text{QTMBR}_{512,0.05}^{6}$ for these three keys figures does not translate into the same dominance for the *rfc* value. There, $\text{QTMBR}_{512,0.05}^{6}$ dominates $\text{KDQT}_{32}^{64,1.0E-5}$ 'only' by a factor of 1.67 (0.65 ‰ divided by 0.39 ‰). This is even more notable since in addition, the share of directly represented resources is also greater for $\text{QTMBR}_{512,0.05}^{6}$ (42.00% versus 34.21% for $\text{KDQT}_{32}^{64,1.0E-5}$) which—at least in theory—should benefit $\text{QTMBR}_{512,0.05}^{6}$, too. The only key figure favoring $\text{KDQT}_{32}^{64,1.0E-5}$ is the mean aspect ratio: It is distinctly closer to the ideal value of 1 (1.54 versus 2.47 $\text{QTMBR}_{512,0.05}^{6}$). At this point of the evaluation, it is unclear whether the disparity in the dominance of $\text{QTMBR}_{512,0.05}^{6}$ between the area-related key figures and the *rfc* value is caused by the usual convergence of the approaches post certain *rds* values, the less favorable aspect ratios of the rectangles described by $\text{QTMBR}_{512,0.05}^{6}$, other inherent properties of the different approaches, or simply arbitrariness. Nevertheless, it is a hint that the ranks for the area-related key figures may not be straightly transferable into ranks for the resulting *rfc* values. We keep track of this observation in the following.

In general, for the aspect ratios, it shows that the *mean* values for both techniques utilizing full-precision MBRs (MBR approach and $\text{MBRQT}^{16,1.0}$) are most unfavorable. This is caused by rather few, very elongated MBRs skewing the mean value (note that for $\text{MBRQT}^{16,1.0}$, the interior quadtree subspaces are of the same shape as the exterior MBR). For the *median* values, all techniques are in reasonable ranges of $\leq$ 2:1. In total, the k-d-based techniques (KDMBR_{256}^{6} and $\text{KDQT}_{32}^{64,1.0E-5}$) describe the most favorably-shaped boxes. This is because for their global k-d space partition description base, cells with an aspect ratio of both 2:1 as well as 1:1 can occur (see Figure 12 on page 80 for KD_{32} as an example)—depending on a cell's level in the hierarchy of the corresponding binary tree depicting the space partition. For the quadtree-based, regular local space partition description base of $\text{QTMBR}_{512,0.05}^{6}$, the resulting cells all have an aspect ratio of 2:1 (the larger side always in the x-dimension) since the aspect ratio of the data space is also 2:1 (see Figure 13 on page 82 for $\text{QT}_{32,1.0}$ as an example). Quantized MBRs most often have the same aspect ratio as the *exterior cell* they are quantized into: Due to the spatial distribution of the resources'

data points in the T1 collection (spatially very narrow in general) and the small resource sizes, most likely, only a *single cell of the exterior cell's internal quantization grid* is occupied with data points. In these cases, the quantized MBRs reproduce the shape of the exterior cell. For the internal quadtrees of $\text{KDQT}_n^{c,a}$, the quadtree subspaces always have the same aspect ratio as the cell they are embedded into. Consequently, the indexed rectangular areas for $\text{QTMBR}_{512,0.05}^6$ are on average more elongated than for both KDMBR_{256}^6 as well as $\text{KDQT}_{32}^{64,1.0E-5}$.
The costs for the initial ranking are highest for $\text{KDQT}_{32}^{64,1.0E-5}$ by far, followed by $\text{QTMBR}_{512,0.05}^6$. See the 'number of indexed areas' in Table 12. The lowest costs in fact occur for $\text{UFS}_{256,cc}$—despite describing more areas than KDMBR_{256}^6, $\text{MBRQT}^{16,1.0}$, and the MBR approach: Since there are only 256 different global subspaces for $\text{UFS}_{256,cc}$, it is sufficient to calculate their respective distances to the query point as well as their surface areas *once*, and to cache them afterwards. The resources' R-Entries used in the ranking (see section 5.1) can then be built from the cached results.
Overall, $\text{QTMBR}_{512,0.05}^6$ proves to be the best technique at 'MBRsize'. Interestingly, its dominance for the *rfc* values is far less pronounced than for the area-related key figures—despite a general correlation between these results. This is to be investigated in the following evaluations.

8.1.2. RESULTS AT 'MBRSIZE-'. In this subsection, the approaches are compared not at 'MBRsize' but at significantly lower target-*rds*-values—which we call 'MBRsize-' in the following. As noted before, we refrain from setting 'MBRsize-' to a predefined target-*rds*-value such as '8 B payload data' since that way, possibly no anchor points at nearby *rds* values are present on *all* the approaches' respective Skylines. Instead, we consider a target-*rds*-value which is significantly smaller than 'MBRsize' *and* where each approach has a nearby anchor point on its Skyline. For the T1 collection, this is the case around 36.5 B *rds*. Consequently, for each approach, we consider the respective Skyline parameterization closest to 36.5 B *rds* to be at 'MBRsize-'. For 'MBRsize-', only four approaches are considered since both the MBR Skyline and the $\text{MBRQT}^{c,a}$ Skyline only begin beyond 41.0 B *rds*. See Figure 44 for the specific techniques at 'MBRsize-'. In Table 13, the corresponding key figures are listed. For KDMBR_{64}^8, $\text{KDQT}_{32}^{32,0.001}$, and $\text{QTMBR}_{256,1.0}^4$, the *rds* values of the respective Skyline parameterizations are almost identical (between 36.45 B *rds* for KDMBR_{64}^8 and 36.54 B *rds* for $\text{QTMBR}_{256,1.0}^4$) which is great for comparing these techniques.
Generally, the area-related key figures exhibit much greater values compared to 'MBRsize'.[178] For example, the 'data space coverage' is between ~2.2 (KDMBR_{64}^8) to ~4.3 ($\text{QTMBR}_{256,1.0}^4$) times greater at 'MBRsize-'. The differences in the 'overlap between summaries' are even bigger, ranging from ~6.2 (KDMBR_{64}^8) to a tremendous ~256 ($\text{QTMBR}_{256,1.0}^4$) times greater.

[178] See Table 12 on page 183 as a reference for the comparative numbers given in the following.

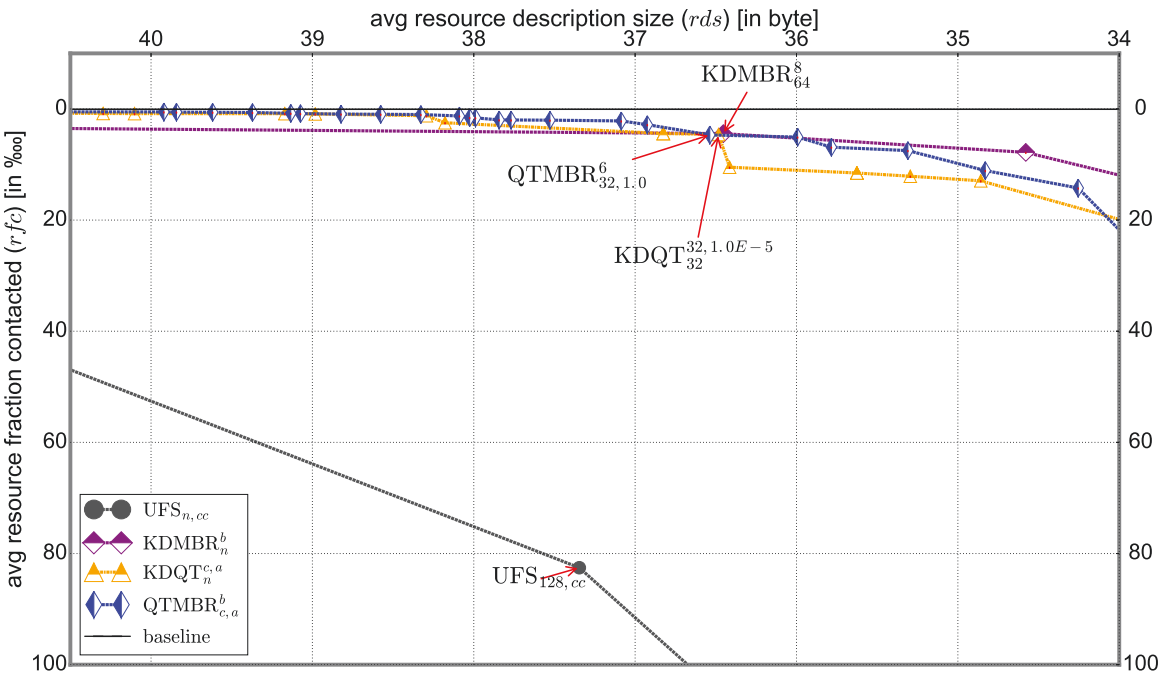


Fig. 44: Zoomed in version of Figure 42, showing the Skylines for the T1 collection around 'MBRsize-'. The respective Skyline parameterizations of the different approaches at 'MBRsize-' are marked with red arrows.

Table 13: Listing of various key figures alongside the resulting *rds* and *rfc* values for the T1 collection and the benchmarked techniques at 'MBRsize-'.

technique → / key figure ↓	UFS$_{128,cc}$	KDMBR$^{8}_{64}$	KDQT$^{32,0.001}_{32}$	QTMBR$^{4}_{256,1.0}$
overlap between summaries [in dsu^2]	2,90E+13	1.36E+9	129,952,707	20,787,526
data space coverage	21,472.4	25.49	10.88	1.45
surface area per summary [in dsu^2]	807.1	0.81	0.32	0.05
number of indexed areas	2,031,498	2,244,699	3,932,243	2,508,472
aspect ratio (mean) [:1]	-	2.10	1.54	2.07
aspect ratio (median) [:1]	-	2.00	2.00	2.00
directly represented resources [in %]	33.77	18.37	11.40	20.14
avg *rfc* [in ‰]	82.56	4.44	4.49	4.68
avg *rds* [in byte]	37.35	36.45	36.48	36.54

Hence, the 'overlap between summaries' obviously grows disproportianally with the 'data space coverage'. In general, the employment of more detailed

quadtrees for the quadtree-utilizing approaches ($KDQT_n^{c,a}$ and $QTMBR_{c,a}^b$) is much more effective for reducing the 'overlap between summaries' compared to the employment of the 'higher' parameterizations[179] (such as additional Voronoi cells) for the other approaches. Let us quickly compare the differences between 'MBRsize' and 'MBRsize-': for e.g. $KDQT_{32}^{64,1.0E-5}$ by contrast with $KDQT_{32}^{32,0.001}$, the overlap is ~127 times lower—while for e.g. $UFS_{256,cc}$ by contrast with $UFS_{128,cc}$, it is 'only' ~9.9 times. Nevertheless, when assessing these numbers (which are accumulated over the entirety of summary-represented resources), it has to be considered that also the amount of summary-represented resources is generally significantly larger at 'MBRsize-'. Hence, for comparing the extent to which the summaries are spatially coarser at 'MBRsize-', it is more suitable to assess the 'surface area per summary'. It shows to be ~1.4 ($KDMBR_n^b$) to ~2.5 ($QTMBR_{c,a}^b$) times greater at 'MBRsize-' compared to 'MBRsize'. As ultimate consequence of the resource descriptions' significantly greater spatial coarseness, the *rfc* values are between ~1.5 ($KDMBR_n^b$) and 11.5 ($QTMBR_{c,a}^b$) greater at 'MBRsize-'. Concerning the 'number of indexed areas', there are even more for $UFS_{n,cc}$ and $KDMBR_n^b$ at 'MBRsize-'[180]—which is against the general rule: It can usually be assumed that 'higher' parameterizations result in a greater 'number of indexed areas' since due to the greater level of description, the data point clouds are divided into more groups. This occuring divergence for $UFS_{n,cc}$ and $KDMBR_n^b$ is attributable to the greater number of summary-represented resources at 'MBRsize-'. For $QTMBR_{c,a}^b$ and especially $KDQT_n^{c,a}$, the 'number of indexed areas' is smaller at 'MBRsize', though.[181]

Looking at the concrete numbers in Table 13, the area-related key figures are best for $QTMBR_{256,1.0}^4$ by far, followed by $KDQT_{32}^{32,0.001}$, $KDMBR_{64}^8$, and $UFS_{128,cc}$. The differences between all the specific techniques are very large. Hence, with regard to the resulting *rfc* values, one would expect the result to be as follows: $QTMBR_{256,1.0}^4 \ll KDQT_{32}^{32,0.001} \ll KDMBR_{64}^8 \ll UFS_{128,cc}$. Nevertheless, this is only true for the placement of $UFS_{128,cc}$—which indeed offers the greatest *rfc* value and thus the worst selectivity by far. For $KDMBR_{64}^8$, $KDQT_{32}^{32,0.001}$, and $QTMBR_{256,1.0}^4$, the differences between the resulting *rfc* values are very small, and the selectivity ranking of the techniques is in fact the exact opposite of the expectation: Despite tremen-

[179] By 'higher' parameterization we mean that parameters are set in such way that spatially more accurate summaries result. For example, $UFS_{2048,cc}$ is a 'higher' parameterization in comparison to $UFS_{512,cc}$.

[180] The concrete numbers are 2,031,498 ($UFS_{128,cc}$) and 2,244,699 ($KDMBR_{64}^8$) indexed areas at 'MBRsize-' respectively 1,739,402 ($UFS_{256,cc}$) and 1,617,235 ($KDMBR_{256}^6$) indexed areas at 'MBRsize'.

[181] The concrete numbers are 2,508,472 ($QTMBR_{256,1.0}^4$) and 3,932,243 ($KDQT_{32}^{32,0.001}$) indexed areas at 'MBRsize-' respectively 2,760,692 ($QTMBR_{512,0.05}^6$) and 4,947,405 ($KDQT_{32}^{64,1.0E-5}$) indexed areas at 'MBRsize'.

dously worse key figures, KDMBR^{8}_{64} surprisingly achieves a slightly lower *rfc* value than both $\text{KDQT}^{32,0.001}_{32}$ and $\text{QTMBR}^{4}_{256,1.0}$.
To find the explanation for these results, we first take a look at the aspect ratios and the share of directly represented resources:

— The median aspect ratio is the same for all three techniques.[182] The mean aspect ratio is even most unfavorable for KDMBR^{8}_{64}.
— Compared to both KDMBR^{8}_{64} as well as $\text{KDQT}^{32,0.001}_{32}$, the share of directly represented resources is greater for $\text{QTMBR}^{4}_{256,1.0}$—which in theory should favor $\text{QTMBR}^{4}_{256,1.0}$ even more.

Thus, both the aspect ratio key figures as well as the share of directly represented resources do not favor KDMBR^{8}_{64} in this comparison. The amount of *relevant* data points administered by directly represented resources is small in general. For the four listed techniques, it is between 1.8 ($\text{UFS}_{128,cc}$) and 0.1 ($\text{KDQT}^{32,0.001}_{32}$) of the top 50 data points on average (see Table 14). Hence, at this point of the evaluation, it seems that the impact of the direct representation on the results is rather low. In section 8.3, this issue is studied in detail. For now, we do not further pursue this topic as the listed numbers cannot explain the occurring results.

Table 14: Number of top 50 data points being contributed by directly represented resources for the respective techniques at 'MBRsize-', averaged over the 50 queries.

	$\text{UFS}_{128,cc}$	KDMBR^{8}_{64}	$\text{KDQT}^{32,0.001}_{32}$	$\text{QTMBR}^{4}_{256,1.0}$
number of top 50 data points from directly represented resources	1.8	1.3	0.1	0.9

With regard to contacting summary-represented resources contributing to the query result twice (see section 6.4.1), the resulting *rfc* values are also not affected much: An *rfc* value of 4.5 ‰ corresponds to ~1,120 resources—while in general, only 12.92 different resources (regardless of being directly represented or summary-represented) contribute to the query result on average. Hence, both the aspect ratio as well as the share of directly represented resources can be excluded as explanations for the severe deviation between the area-related key figures and the resulting *rfc* values.
As the *rfc* values are averaged over 50 queries, an assessment of the descriptive statistic *rfc* values is worthwhile to review if the averaged values are influenced by outliers. See Table 15 for a corresponding listing. Nevertheless, from these values, no anomalies are apparent which might ex-

[182] Note that the ranks are assigned to the techniques based on the exact results determined by the Apache Commons Math library. These exact values often differ from about the sixth decimal place on. Hence, e.g. the ranks of KDMBR^{8}_{64}, $\text{KDQT}^{32,0.001}_{32}$, and $\text{QTMBR}^{4}_{256,1.0}$ are different for the mean aspect ratio even though the rounded values of the mean aspect ratio are 2.00 for all of them.

plain the resulting (average) *rfc* values. In fact, the outlier values ('min rfc'-value and 'max rfc'-value) are even rather convenient for the apparently underperforming $\text{KDQT}_{32}^{32,0.001}$ and $\text{QTMBR}_{256,1.0}^{4}$ whilst the 25%-quantile, median, and 75%-quantile values are fairly similar for all. The smaller average *rfc* value for KDMBR_{64}^{8} in fact results because for $\text{KDQT}_{32}^{32,0.001}$ and $\text{QTMBR}_{256,1.0}^{4}$, the values *between* the 75%-quantile and the maximum value are greater. If for all three techniques, the *rfc* values occurring for the 50 queries are sorted in ascending order, the results are as shown in Table 16.

Table 15: Descriptive statistic *rfc* values for KDMBR_{64}^{8}, $\text{KDQT}_{32}^{32,0.001}$, and $\text{QTMBR}_{256,1.0}^{4}$ which are assessed in the qualitative analysis for the `T1` collection at 'MBRsize-'.

technique → desc. stat. *rfc* values ↓	KDMBR_{64}^{8}	$\text{KDQT}_{32}^{32,0.001}$	$\text{QTMBR}_{256,1.0}^{4}$
avg *rfc* [in ‰]	4.44	4.60	4.68
min *rfc* [in ‰]	0.38	0.10	0.04
25%-quant. *rfc* [in ‰]	1.25	1.12	1.07
median *rfc* [in ‰]	3.14	2.87	3.22
75%-quant. *rfc* [in ‰]	6.01	6.14	5.99
max *rfc* [in ‰]	37.63	23.39	22.87

Table 16: Listing of selected *rfc* values for the three techniques assessed in the qualitative analysis for the `T1` collection at 'MBRsize-'. The *rfc* values have been sorted in ascending order, and some exemplary ranks and the respective *rfc* values are displayed for assessing the occuring distributions of the *rfc* values.

technique → *rfc* values by rank↓	KDMBR_{64}^{8} *rfc* [in ‰]	$\text{KDQT}_{32}^{32,0.001}$ *rfc* [in ‰]	$\text{QTMBR}_{256,1.0}^{4}$ *rfc* [in ‰]
rank 1 (min *rfc*)	0.38	0.10	0.04
...			
rank 5	0.58	0.41	0.31
...			
rank 40 (80%-quant. *rfc*)	6.22	6.32	6.92
...			
rank 48	8.60	19.97	19.81
...			
rank 50 (max *rfc*)	37.63	23.39	22.87

In general, the spread of the *rfc* values occuring for the single queries is bigger for $\text{KDQT}_{32}^{32,0.001}$ and $\text{QTMBR}_{256,1.0}^{4}$: There are more queries with a

very low *rfc* value but also more queries with a rather large *rfc* value. The 'max rfc'-value of 37.63 ‰ for KDMBR_{64}^{8} is a single outlier contradicting this general observation. In total, the underperformance of $\text{KDQT}_{32}^{32,0.001}$ and $\text{QTMBR}_{256,1.0}^{4}$ compared to KDMBR_{64}^{8} cannot be explained by few outlier queries distorting the average *rfc* values.

Table 17: Examplarily selected queries and the resulting *rfc* values for the three techniques assessed in the qualitative analysis for the T1 collection at 'MBRsize-'. Note that for the given query locations, x corresponds to *long* and y corresponds to *lat*.

query	x	y	'real-world' location	KDMBR_{64}^{8} *rfc* [in ‰]	$\text{KDQT}_{32}^{32,0.001}$ *rfc* [in ‰]	$\text{QTMBR}_{256,1.0}^{4}$ *rfc* [in ‰]
$ID = 7$	-73.94	40.83	New York City, New York	6.22	23.39	17.82
$ID = 14$	-121.97	37.40	San José, California	7.85	7.97	7.28
$ID = 37$	-43.26	-22.88	Rio de Janeiro, Brasil	37.63	12.84	9.35

Of course, the performances of the single techniques differ for specific queries. The numbers for three examplarily selected queries are depicted in Table 17. For the query with $ID = 7$ (query 7), KDMBR_{64}^{8} is clearly the best performing technique. The x/y-coordinates of the query point correspond to a location in New York City where the point density is extremely high on a global scale (also see Figure 30b on page 113): 8,940 resources administer data points in a *close neighborhood* (i.e. within a radius of 5 km) to the query point.[183] For query 14, the techniques show fairly similar performances. The query location corresponds to San José at the US West Coast where the point density is also high but not nearly as high as in New York City. In the *close neighborhood* to query 14, an amount of 989 resources administers data points. For query 37, KDMBR_{64}^{8} shows the worst performance by far. In comparison to query 14, there is a higher point density immediately around query 37—1,838 resources administer data points in *close neighborhood* to the query point. The query location corresponds to Rio de Janeiro which is in a region with one of the highest point densities on the South American continent. Nevertheless, the data point density in South America as a whole is significantly lower compared to North America.

As described in section 6.4.3, the basic k-d space partition for KDMBR_{64}^{8} respectively $\text{KDQT}_{32}^{32,0.001}$ is learned by utilizing a bucket overflow approach. For the T1 collection, it uses training data that has a similar data point distribution as the test data collection. Hence, the global k-d cells should be much coarser around Rio de Janeiro compared to New York City. This

[183] The specified kilometer values were determined with the tool from [Veness 2016].

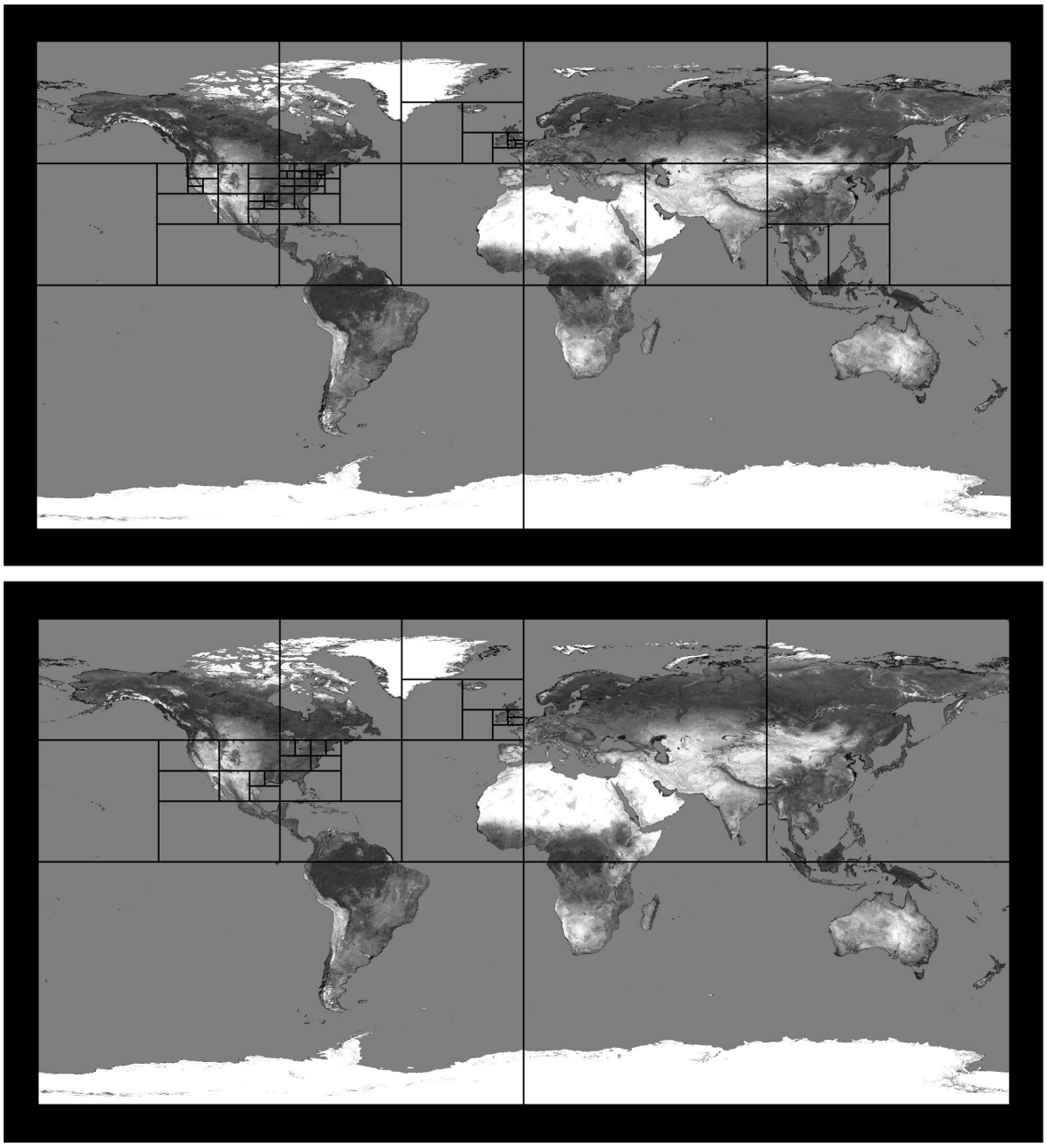


Fig. 45: Visualization of the resulting global k-d space partition for $n = 64$ (top) respectively $n = 32$ (bottom). These space partitions result for the T1 collection and all the k-d-based approaches (KD$_n$, KDMBR$_n^b$, KDMAR$_n^{b,k}$, and KDQT$_n^{c,a}$) whenever parameter $n = 64$ respectively $n = 32$ (due to the seeding in the training phase).

assumption can be verified: In Figure 45, the concrete basic k-d space partitions resulting for $n = 64$ respectively $n = 32$ are shown. In the region around New York City, the k-d cells are most detailed while the whole southern hemisphere (containing Rio de Janeiro) only features two cells in total. With regard to the quantized MBRs, 256 different positions can be distinguished per bound when $b = 8$. As a consequence, in the region around New York City, the resource summaries for KDMBR$_{64}^8$ *can be* spatially very accurate. In Figure 46, we exemplarily visualize the KDMBR$_{64}^8$

summary for the resource with $ID = 501,162$ (resource 501,162) which administers the nearest neighbor to query 7. At the top of Figure 46, the resource's summary is shown in the maximum zoom level of our visualization tool. The two areas indexed by the summary are so small they cannot be recognized. Therefore, at the bottom of Figure 46, there is additionally a zoomed in depiction of the image file where the two indexed areas of resource 501,162 can be spotted. As a comparison, Figure 47 depicts the $\text{QTMBR}^4_{256,1.0}$ summary for resource 501,162 in the maximum zoom level of the visualization tool. Despite also being small in the depiction, the two indexed areas can be spotted and are significantly greater compared to those indexed for KDMBR^8_{64}. Notably, the *basic cells* of the local quadtree space partition are smaller than those of the global k-d space partition—but there are only $b = 4$ bits or 16 positions per bound available for the cell-interior quantized MBRs of $\text{QTMBR}^4_{256,1.0}$. In total, this combination obviously results in less accurate $\text{QTMBR}^4_{256,1.0}$ summaries compared to the KDMBR^8_{64} summaries for the resources administering data points in the region of New York City.

Considering the resulting *rfc* values and the exemplary visualizations, we conclude the following: In case the query point is located in a region which on a global scale is densely populated with data points, the basic k-d space partition of KDMBR^8_{64} is sufficiently accurate. In combination with its precise MBR refinement using $b = 8$ bits per bound, this results in its smallest indexable spatial unit to have a significantly smaller surface area in this region compared to the corresponding smallest indexable spatial units of $\text{QTMBR}^4_{256,1.0}$.[184] Notably, the resources administering data points in New York City are oftentimes spatially *concentrated* (i.e. confined to a very narrow area). In addition, KDMBR^8_{64} benefits from the fact that the border between two basic k-d cells runs through the city zone of New York City (see Figure 46). In total, this results in more accurate KDMBR^8_{64} summaries for the resources from which data points are located in the neighborhood of query 7.

For regions not as densely populated with data points (such as the South American continent), the basic k-d cells of KDMBR^8_{64} can be very large such that the 256 positions per bound are not sufficient for its smallest indexable spatial units to be equally small as for $\text{QTMBR}^4_{256,1.0}$. We exemplarily investigated this claim for query 37 with our visualization tool, too. The corresponding depiction can be found in Figure 48.

Furthermore, for such big basic cells, very large quantized MBRs containing a lot of dead space can occur—in case the data points located in this cell are spatially spread. For example, imagine a resource administering data

[184] In general, the smallest indexable spatial unit of KDMBR^b_n and $\text{QTMBR}^b_{c,a}$ is dependent on a *basic cell*'s internal quantization grid. A *cell of the internal quantization grid* is the smallest spatial unit which can be described for both approaches. The more fine-grained the combination of the basic cell and the basic cell's internal quantization grid, the smaller the surface area of a cell of the internal quantization grid.

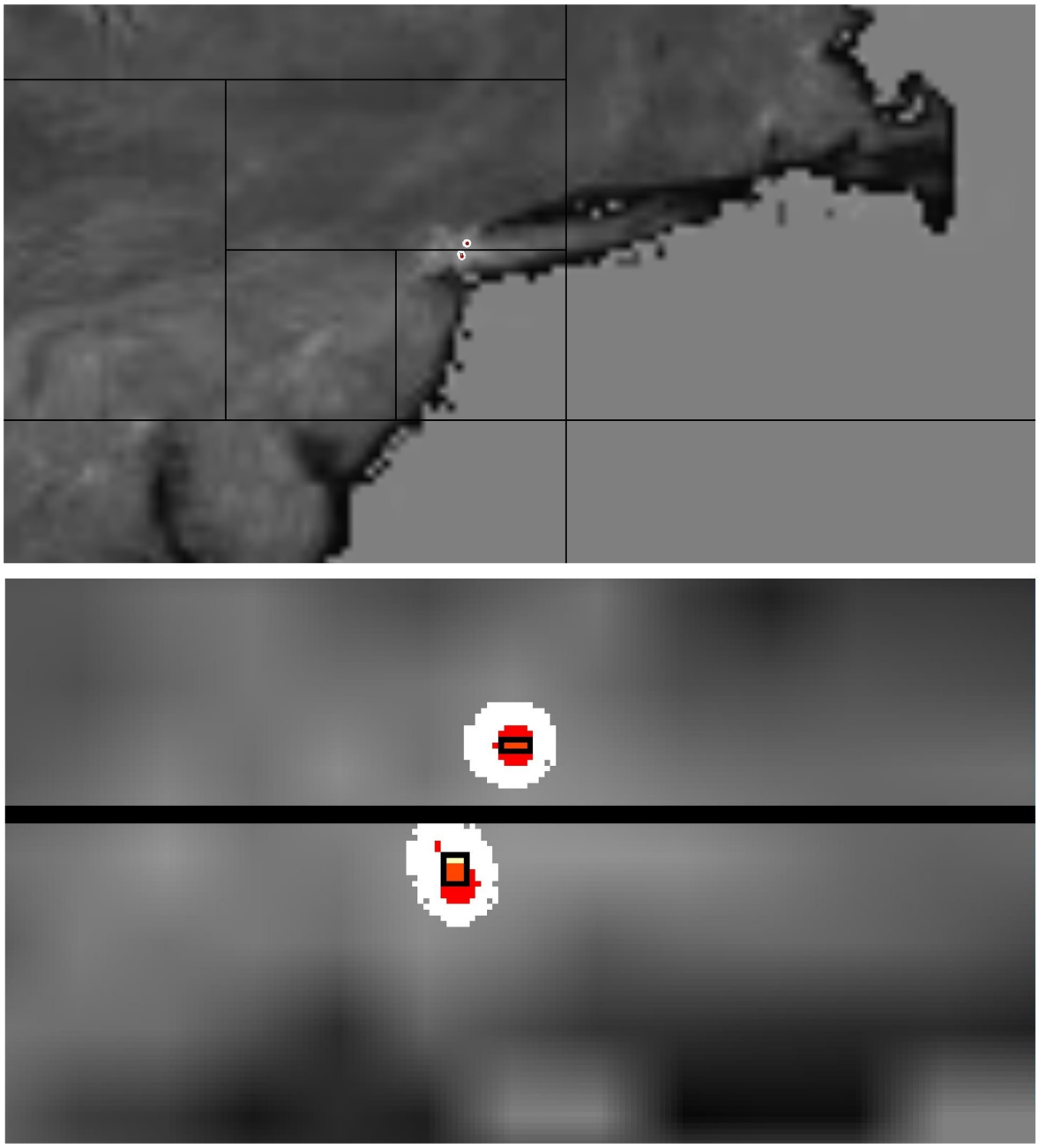


Fig. 46: Visualization of the $KDMBR^{8}_{64}$ summary for resource 501,162 which administers 25 data points (from which one is the nearest neighbor for query 7, located in New York City). The summary is non-zipped and requires 43 B. On top, the summary is depicted in the maximum zoom level of our visualization tool. Since the two indexed rectangular areas cannot be spotted there, we also show a zoomed in depiction of the image file (bottom) where the areas are recognizable. Note that the white-stroked red circles depict the data points of resource 501,162.

points in Angola and New Zealand for the k-d space partitions of Figure 45. In such a case, a large portion of the basic k-d cell is covered by its quantized MBR. Such very large indexed areas contain a lot of dead space and have a negative impact on an approach's selectivity, obviously. Although only a relatively small share of the resources is spatially very spread, such

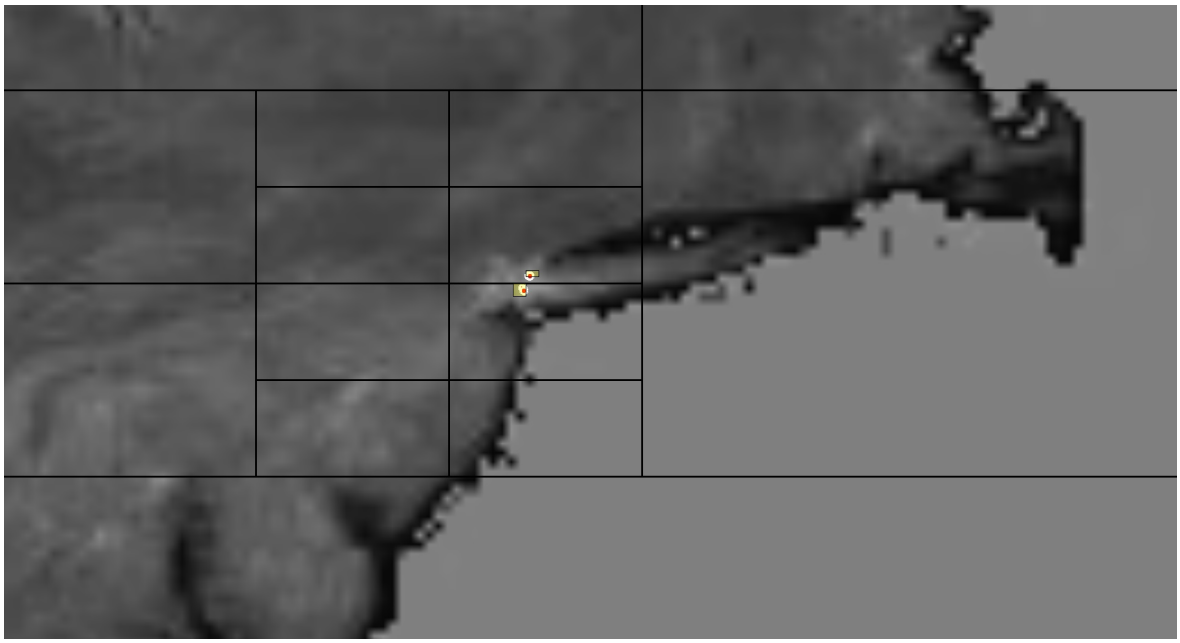


Fig. 47: Visualization of the $\text{QTMBR}^{4}_{256,1.0}$ summary for resource 501,162, depicted in the maximum zoom level of our visualization tool. The summary is `lq-nz`-coded and requires 40 B.

resources exist and can have noticeable effects on the resulting *rfc* values due to the great cardinality of the resource set (in contrast to e.g. the double contact to summary-represented resources contributing to the query result). Additionally, these indexed areas lead to the enormous numbers of the area-related key figures for KDMBR^{8}_{64} in Table 13. The greater the basic cells, the greater the risk that such quantized MBRs of little benefit occur. For $\text{QTMBR}^{4}_{256,1.0}$, due to the local space partitioning, it makes no difference if a region, on a global scale, is densely populated with data points or not—the descriptions of the occupied regions solely adjust to the data points of the resource to describe. Hence, since there are no as huge basic cells as they inevitably occur for the k-d space partition, the 'potential' for 'overlap between summaries' and 'data space coverage' is drastically reduced for $\text{QTMBR}^{4}_{256,1.0}$. In Figure 49, there is an example comparison for the KDMBR^{8}_{64} summary (top) and the $\text{QTMBR}^{4}_{256,1.0}$ summary (bottom) of resource 5,971, illustrating the previous explanations.
$\text{KDQT}^{32,0.001}_{32}$ with its very coarse basic k-d space partition (only 32 global cells as opposed to the 64 global cells of KDMBR^{8}_{64}, see Figure 45) and the detailed refinement of occupied cells by the cell-interior quadtrees is somewhere in between KDMBR^{8}_{64} and $\text{QTMBR}^{4}_{256,1.0}$. Since the construction of the cell-interior quadtrees is stopped prematurely at a certain point due to parameter $a = 0.001$, the smallest indexable spatial unit of $\text{KDQT}^{32,0.001}_{32}$ is not as accurate as for e.g. KDMBR^{8}_{64} in the region around New York City. In general, its smallest indexable spatial unit is also coarser compared to $\text{QTMBR}^{4}_{256,1.0}$ (visually verified with our visualization tool, not depicted in this work). Nevertheless, as each global k-d cell is refined with a quadtree, the 'threat' of very large cell-interior refinements containing a lot of dead

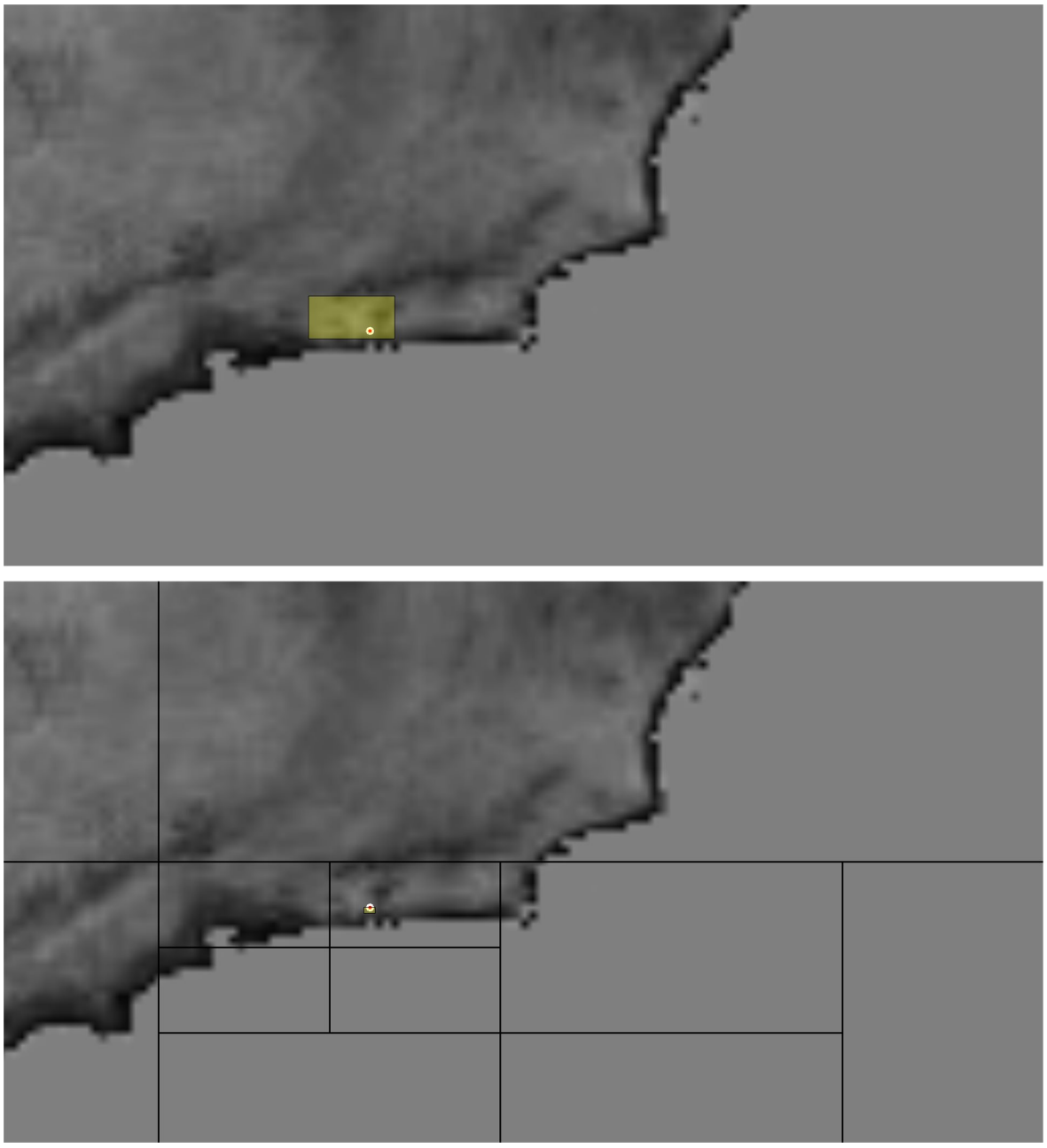


Fig. 48: Visualization of the KDMBR^{8}_{64} (top, non-zipped, 33 B) and $\mathrm{QTMBR}^{4}_{256,1.0}$ (bottom, `lq-nz`-coded, 35 B) summaries for resource 412,415 of the `T1` collection which administers 11 data points (from which one is the nearest neighbor to query 37, located in Rio de Janeiro). For both, the visualization tool is in its maximum zoom level. All the data points of resource 412,415 are at positions which are extremely close to each other, i.e. they are all located in a single cell of the respective basic cell's interior quantization grid. The smallest indexable spatial unit for $\mathrm{QTMBR}^{4}_{256,1.0}$ is much smaller than for KDMBR^{8}_{64}.

space is significantly reduced: possibly multiple, *mutually independent* areas are indexed within an occupied cell. This contrasts with a quantized MBR where all the data points located in an occupied cell must be covered by the MBR. Hence, the area-related key figures of $\mathrm{KDQT}^{32,0.001}_{32}$ are

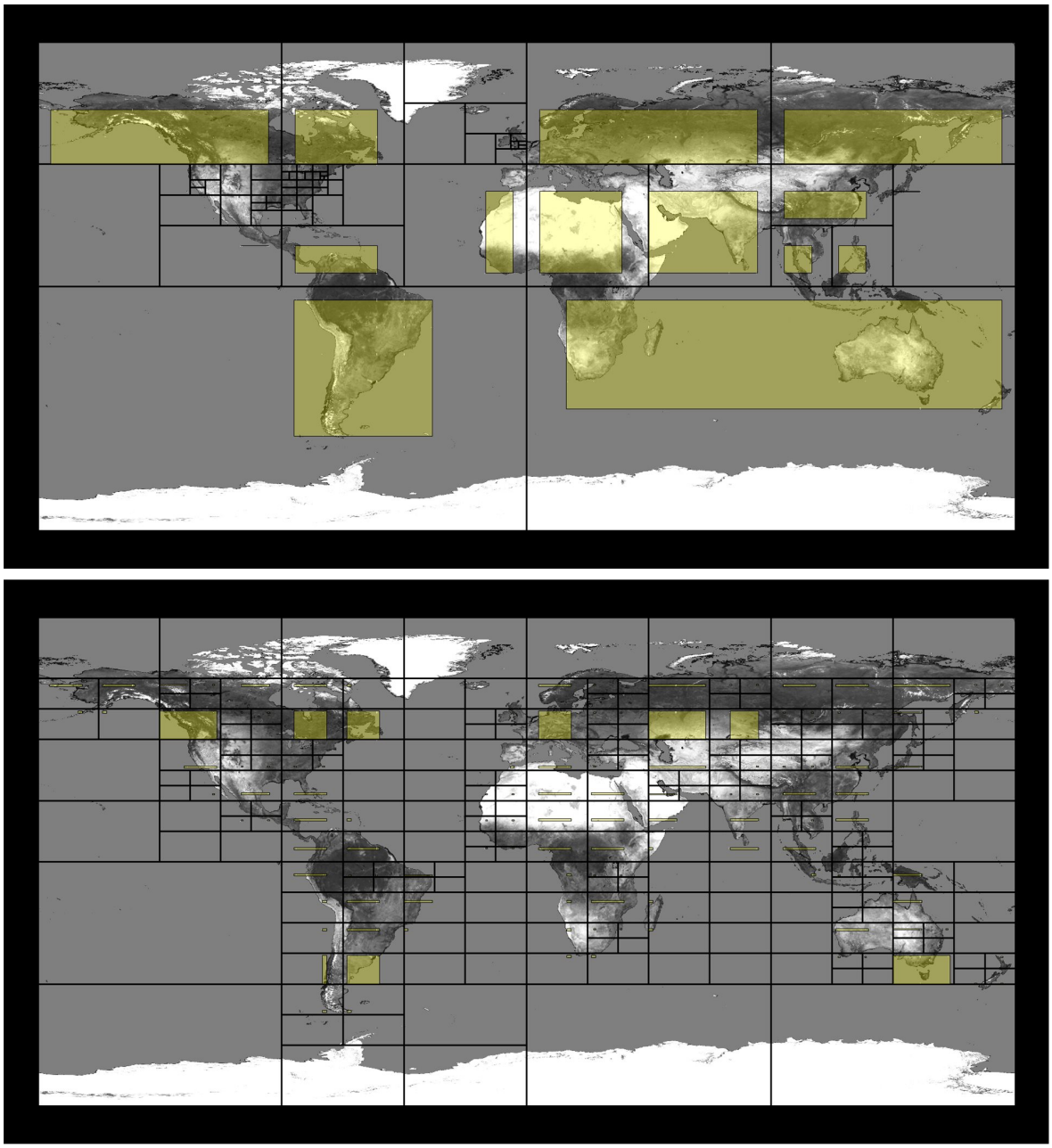


Fig. 49: Visualization of the KDMBR^{8}_{64} summary (top, non-zipped, 149 B) and the $\mathrm{QTMBR}^{4}_{256,1.0}$ summary (bottom, `cblq-z`-coded, 379 B) for resource 5,971 of the `T1` collection. Note that the resource administers 780 data points.

much better compared to KDMBR^{8}_{64} and closer to those of $\mathrm{QTMBR}^{4}_{256,1.0}$. By tendency, for $\mathrm{KDQT}^{32,0.001}_{32}$, also the resulting *rfc* values for the specific queries are closer to those of $\mathrm{QTMBR}^{4}_{256,1.0}$ than to those of KDMBR^{8}_{64}. Hence, KDMBR^{8}_{64} outperforms $\mathrm{KDQT}^{32,0.001}_{32}$ and $\mathrm{QTMBR}^{4}_{256,1.0}$ for queries in regions with a high global point density while for queries in other regions, it is inferior.

As the query points are randomly drawn from the training data set, they are mostly from regions which are densely populated with data points—and sometimes from extremely dense regions (such as New York City; also

compare Figure 30b on page 113 on the left). For query points in such regions, high *rfc* values result for all techniques. KDMBR^{8}_{64} is at an advantage here since its smallest indexable spatial unit is the most accurate of the three techniques (also see the ranks ≥ 40 in Table 16). For sparsely populated regions, there is a much lower amount of resources administering data points. This means that the amount of candidate resources which might erroneously be placed at high ranking positions due to coarse summaries is small compared to densely populated regions (which require very accurate summaries for a precise ranking). Hence, the comparative coarseness of the KDMBR^{8}_{64} summaries in sparsely populated regions is not as big of a disadvantage as one could suspect from an assessment of the area-related key figures. All in all, it results in about the same *rfc* values for KDMBR^{8}_{64}, $\text{KDQT}^{32,0.001}_{32}$, and $\text{QTMBR}^{4}_{256,1.0}$ on average—despite the poor area-related key figures for KDMBR^{8}_{64}. In general, KDMBR^{8}_{64} also profits from the fact that both $\text{KDQT}^{32,0.001}_{32}$ and $\text{QTMBR}^{4}_{256,1.0}$ are limited in the maximum accuracy of their smallest indexable spatial units—which are obviously not accurate enough for extreme agglomerations such as New York City. For 'higher' parameterizations of the approaches and therefore more accurate resource descriptions, we expect greater convergence between the *rfc* values and the corresponding area-related key figures:

— The maximum spatial accuracy of the approaches which adapt stronger to the individual resource (such as $\text{QTMBR}^{b}_{c,a}$) is much higher with these parameterizations. Hence, also in extreme agglomerations such as New York City, the spatial accuracy should then be high enough to effectively distinguish between relevant and irrelevant resources.
— For KDMBR^{b}_{n}, the basic k-d space partition is finer when utilizing greater values for n. Hence, the basic k-d cells are then not as huge as the two cells for the southern hemisphere when $n = 64$. Consequently, the problems with the too big smallest indexable spatial units and the large quantized MBRs should be alleviated. Also, the extremely large values for the 'overlap between summaries' and the 'data space coverage' should be reduced.

For 'MBRsize-', in total, KDMBR^{8}_{64} is the most suitable technique since in addition to its minimally superior *rfc* value, it also offers the lowest computational costs for the initial resource ranking compared to $\text{KDQT}^{32,0.001}_{32}$ and $\text{QTMBR}^{4}_{256,1.0}$ (see 'number of indexed areas' in Table 13).

8.1.3. RESULTS AT 'MBRSIZE+'. In this subsection, the approaches are compared at 'MBRsize+' which is analogously determined as 'MBRsize-'. The target-*rds*-value for 'MBRsize+' is 46.5 B. In Figure 50, the respective techniques are shown while Table 18 depicts the corresponding key figures. The non-parameterizable MBR approach is missing at 'MBRsize+' as its sole Skyline anchor point has an *rds* value of 41.1 B. As expected, there is a strong correlation between the area-related key fig-

ures and the resulting *rfc* values: the greater the area-related key figures, the worse usually the resulting *rfc* values. Hence, for more accurate resource descriptions, the resulting *rfc* values can indeed be better estimated by the area-related key figures as opposed to 'MBRsize-'. Notably, the 'number of indexed areas' is by far highest for the techniques utilizing internal quadtrees ($\text{KDQT}_{128}^{256,1.0E-5}$ and $\text{MBRQT}^{512,1.0E-5}$) even though for $\text{QTMBR}_{1024,0.001}^{8}$, there are significantly more summary-represented resources.

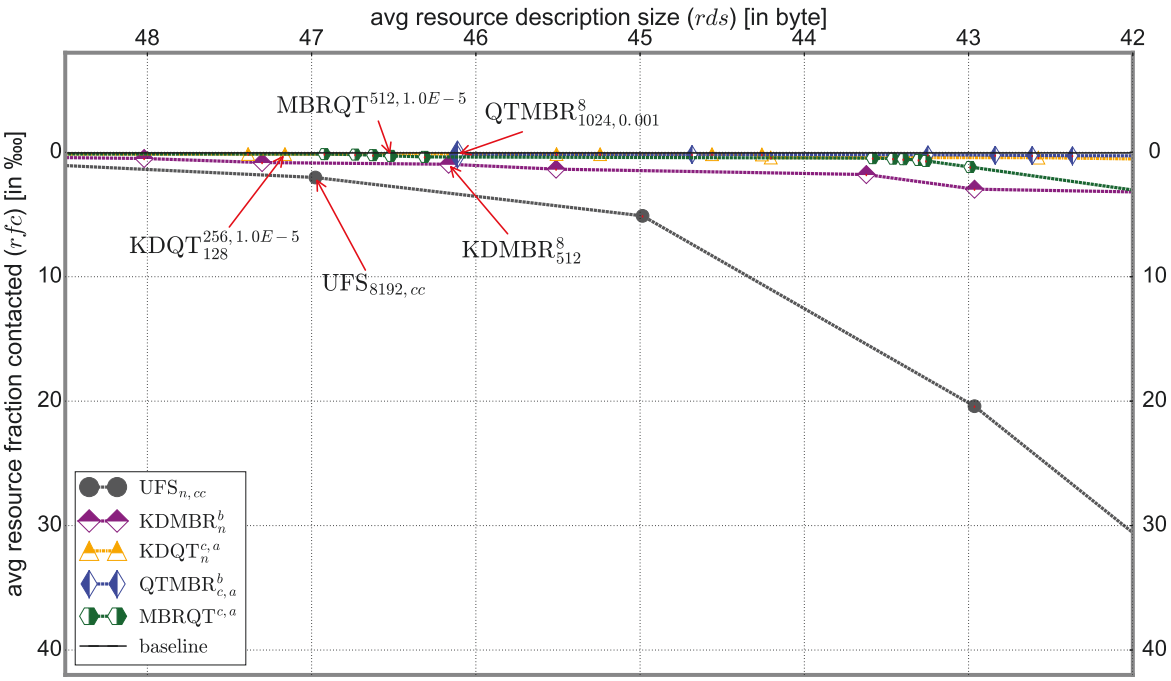


Fig. 50: Zoomed in version of Figure 42 on page 181, showing the Skylines for the T1 collection around 'MBRsize+'. The respective Skyline parameterizations of the different approaches at 'MBRsize+' are marked with red arrows.

For the single techniques, the differences between the respective *rfc* values as well as the area-related key figures are usually large at 'MBRsize+'. For example, $\text{KDMBR}_{512}^{1.0E-5}$ (at rank 5 of the *rfc* ranking, *rfc*: 0.96 ‰) is more than twice as selective than $\text{UFS}_{8192,cc}$ (rank 6, *rfc*: 2.03 ‰). Only between $\text{KDQT}_{128}^{256,1.0E-5}$ (rank 1, *rfc*: 0.14 ‰) and $\text{QTMBR}_{1024,0.001}^{8}$ (rank 2, *rfc*: 0.17 ‰), it is very close: The absolute difference of 0.03 ‰ corresponds to 7 resources. Despite the greater comformance compared to 'MBRsize-', the techniques' ranks for the area-related key figures do not all coincide with the ranks for the *rfc* values at 'MBRsize+'. It is also the first time that different rankings result for the three area-related key figures. See the 'data space coverage'- and the 'surface area per summary'-rows in Table 18: There, $\text{KDQT}_{128}^{256,1.0E-5}$ is only second despite achieving the best *rfc* value and the smallest 'overlap between summaries'. In the end, the *rfc* ranking only fully coincides with the ranking for the 'overlap between summaries'. $\text{QTMBR}_{1024,0.001}^{8}$ is slightly behind $\text{KDQT}_{128}^{256,1.0E-5}$ with regard

Table 18: Listing of various key figures alongside the resulting *rds* and *rfc* values for the T1 collection and the benchmarked techniques at 'MBRsize+'.

technique → / key figure ↓	$UFS_{8192,cc}$	$KDMBR^{8}_{512}$	$KDQT^{256,1.0E-5}_{128}$	$QTMBR^{8}_{1024,0.001}$	$MBRQT^{512,1.0E-5}$
overlap between summaries [in dsu^2]	2,533,813,617	8,168,538	1,441	2,120	405,342
data space coverage	208.6	5.53	0.14	0.05	2.70
surface area per summary [in dsu^2]	14.12	0.33	0.007	0.003	0.16
number of indexed areas	2,313,821	1,501,747	5,282,215	3,554,076	4,956,441
aspect ratio (mean) [:1]	-	2.40	1.54	2.77	4.45
aspect ratio (median) [:1]	-	2.00	2.00	2.00	1.91
directly represented resources [in %]	61.69	56.92	51.23	45.81	54.79
avg *rfc* [in ‰]	2.03	0.96	0.14	0.17	0.29
avg *rds* [in byte]	46.98	46.17	47.16	46.11	46.52

to the *rfc* value but exhibits the smallest values for 'data space coverage' and 'surface area per summary'. Still, $QTMBR^{8}_{1024,0.001}$ is at a disadvantage compared to $KDQT^{256,1.0E-5}_{128}$ with respect to the mean aspect ratio (2.77 for $QTMBR^{8}_{1024,0.001}$ versus 1.54 for $KDQT^{256,1.0E-5}_{128}$) and the share of directly represented resources (45.8% versus 51.2%). We shortly assess the impact of both in the following.

In Table 19, the number of top 50 data points which are contributed from directly represented resources is depicted for the different techniques (averaged over all 50 queries). The difference between the biggest number (4.82 for $UFS_{8192,cc}$) and the smallest number (3.26 for $QTMBR^{8}_{1024,0.001}$) is 1.5, only. This is even though there are almost 400,000 more directly represented resources for $UFS_{8192,cc}$ than for $QTMBR^{8}_{1024,0.001}$. Furthermore, these numbers are almost completely contradictory to the *rfc* values: the biggest number occurs for $UFS_{8192,cc}$ which has the worst *rfc* value while the smallest number occurs for $QTMBR^{8}_{1024,0.001}$ which has almost the best *rfc* value. Thus, at this point of the evaluation, the impact of directly represented resources still has to be considered as low.[185]

With regard to the aspect ratio, the mean values show that for $QTMBR^{8}_{1024,0.001}$, there is a tendency that some of the quantized MBRs feature a very high aspect ratio. This could already be seen in Figure 49

[185] See section 8.3 for a detailed analysis of this issue.

Table 19: Average number of top 50 data points being contributed by directly represented resources for the respective techniques at 'MBRsize+' of the T1 collection.

	$\mathrm{UFS}_{8192,cc}$	KDMBR^{8}_{512}	$\mathrm{KDQT}^{256,1.0E-5}_{128}$	$\mathrm{QTMBR}^{8}_{1024,0.001}$	$\mathrm{MBRQT}^{512,1.0E-5}$
number of data points	4.82	4.72	3.54	3.26	4.04

(bottom) in the qualitative analysis of the techniques at 'MBRsize-' (section 8.1.2). From $\mathrm{QTMBR}^{4}_{256,1.0}$ (at 'MBRsize-') over $\mathrm{QTMBR}^{6}_{512,0.05}$ (at 'MBRsize') to $\mathrm{QTMBR}^{8}_{1024,0.001}$ (at 'MBRsize+'), the mean aspect ratio continuously worsens at significant rates (from 2.07 over 2.47 to 2.77). Hence, the tendency for high aspect ratio MBRs seems to increase for 'higher' parameterizations of $\mathrm{QTMBR}^{b}_{c,a}$. In contrast, the mean aspect ratio for $\mathrm{KDQT}^{c,a}_{n}$ is very stable (1.54 for all its techniques at 'MBRsize-', 'MBRsize', and 'MBRsize+'). The aspect ratio is *possibly* part of the explanation why $\mathrm{QTMBR}^{8}_{1024,0.001}$ is inferior to $\mathrm{KDQT}^{512,1.0E-5}_{128}$ despite its lower 'data space coverage'. At this point, we dive into the qualitative analysis of the results at 'MBRsize+'.

Since the *rfc* ranking positions of the other techniques can easily be explained on basis of the huge differences between their area-related key figures, we only consider $\mathrm{KDQT}^{256,1.0E-5}_{128}$ and $\mathrm{QTMBR}^{8}_{1024,0.001}$ in the qualitative analysis. With regard to the descriptive statistic *rfc* values for the 50 queries (see Table 20), the values correspond to the resulting average *rfc* value: $\mathrm{KDQT}^{256,1.0E-5}_{128}$ is always better or at least as good as $\mathrm{QTMBR}^{8}_{1024,0.001}$. The same applies when the respective *rfc* values occuring for the 50 queries are sorted in ascending order and then, both lists are compared to each other position-wise (analogously to Table 16 on page 190).[186] A comparison of the individual queries shows that $\mathrm{QTMBR}^{8}_{1024,0.001}$ is slightly better for 11 queries, equally well for 14 queries, and (slightly) worse for 25 queries.[187] Table 21 depicts the queries for which $\mathrm{KDQT}^{256,1.0E-5}_{128}$ is most superior to $\mathrm{QTMBR}^{8}_{1024,0.001}$ (query 22, located in Athens, Ohio) respectively for which $\mathrm{QTMBR}^{8}_{1024,0.001}$ is most superior to $\mathrm{KDQT}^{256,1.0E-5}_{128}$ (query 25, located in Corinth, Texas, a suburb of Dallas) along with the corresponding *rfc* values. The gap for query 25 corresponds to ~22 resources. Since this gap is not very significant, we do not further investigate query 25. For query 22, we first investigate the $\mathrm{KDQT}^{256,1.0E-5}_{128}$ and $\mathrm{QTMBR}^{8}_{1024,0.001}$ summaries

[186] Note that the corresponding lists are not depicted in this work.

[187] Note that for determining if $\mathrm{QTMBR}^{8}_{1024,0.001}$ is better, equally well, or worse, the *rfc* values [in ‰] have been rounded to two decimal places.

Table 20: Descriptive statistic *rfc* values for $\text{KDQT}_{128}^{256,1.0E-5}$ and $\text{QTMBR}_{1024,0.001}^{8}$ which are assessed in the qualitative analysis for the T1 collection at 'MBRsize+'.

technique → / desc. stat. *rfc* values ↓	$\text{KDQT}_{128}^{256,1.0E-5}$	$\text{QTMBR}_{1024,0.001}^{8}$
avg *rfc* [in ‰]	0.14	0.17
min *rfc* [in ‰]	0.04	0.04
25%-quant. *rfc* [in ‰]	0.05	0.06
median *rfc* [in ‰]	0.11	0.12
75%-quant. *rfc* [in ‰]	0.22	0.22
max *rfc* [in ‰]	0.81	1.44

Table 21: Examplarily selected queries and the corresponding *rfc* values for $\text{KDQT}_{128}^{256,1.0E-5}$ and $\text{QTMBR}_{1024,0.001}^{8}$ which are assessed in the qualitative analysis for the T1 collection at 'MBRsize+'.

query	x	y	$\text{KDQT}_{128}^{256,1.0E-5}$ *rfc* [in ‰]	$\text{QTMBR}_{1024,0.001}^{8}$ *rfc* [in ‰]
$ID = 22$	-82.10	39.33	0.81	1.44
$ID = 25$	-97.06	33.16	0.29	0.20

for resource 219,186 which administers the nearest neighbor to the query point. For both, the indexed areas are so small that in the maximum zoom level of our visualization tool, these areas are far from being recognizable. Even when the image files are zoomed down to the pixel level, no rectangular areas are perceptible. This is no surprise when assessing the respective surface areas: For the $\text{KDQT}_{128}^{256,1.0E-5}$ summary, it is 2.64E-5 dsu^2 (for nine indexed areas) while for $\text{QTMBR}_{1024,0.001}^{8}$, it is 8.52E-5 dsu^2 (for one indexed area).[188] Hence, the surface area numbers for resource 219,186 favor the $\text{KDQT}_{128}^{256,1.0E-5}$ summary as being more accurate—especially when considering that the total surface area is spread over significantly more indexed areas which implies a more specific description of the data point clouds. In Table 22, we additionally list the surface areas and the 'number of indexed areas' for other resources contributing to the result of query 22. All of these resources are spatially narrow and administer data points only in or near Athens. In total, 19 different resources contribute to the result of query 22. For the listing in Table 22, the average surface area of the $\text{KDQT}_{128}^{256,1.0E-5}$ summaries is 40% smaller while indexing 8.8 areas on average—as opposed to 1.6 areas for $\text{QTMBR}_{1024,0.001}^{8}$. The given numbers for the few exemplarily selected resources are a hint that in the region of

[188] Note that the surface area of 8.52E-5 dsu^2 for $\text{QTMBR}_{1024,0.001}^{8}$ corresponds to a square of ~800 meters edge length in Athens if a single, quadratic area was to be described.

query 22, the $\text{KDQT}_{128}^{256,1.0E-5}$ summaries are simply more accurate and better adjusted to the data point clouds than the $\text{QTMBR}_{1024,0.001}^{8}$ summaries. Nevertheless, we dive further into the analysis since the difference in the occurring *rfc* values (which corresponds to ~157 fewer resources contacted for $\text{KDQT}_{128}^{256,1.0E-5}$) still appears to be unusually large.

Table 22: Exemplarily selected resources of the T1 collection which contribute to the result for query 22 alongside the respective surface areas covered by their $\text{KDQT}_{128}^{256,1.0E-5}$ respectively $\text{QTMBR}_{1024,0.001}^{8}$ summaries and the corresponding numbers of indexed areas.

resource		$\text{KDQT}_{128}^{256,1.0E-5}$		$\text{QTMBR}_{1024,0.001}^{8}$	
ID	# data points	total surface area [in dsu^2]	# indexed areas	total surface area [in dsu^2]	# indexed areas
219,186	92	2.64E-05	7	8.52E-05	1
175,825	23	3.39E-05	9	5.61E-05	2
408,262	36	2.64E-05	7	5.04E-05	1
1,373,342	7	1.89E-05	5	9.15E-05	2
38,402	63	7.17E-05	16	1.14E-05	2

Table 23: Exemplarily selected resources alongside their ranking positions in the $\text{KDQT}_{128}^{256,1.0E-5}$ respectively $\text{QTMBR}_{1024,0.001}^{8}$ ranking for query 22 (T1 collection).

resource	$\text{KDQT}_{128}^{256,1.0E-5}$	$\text{QTMBR}_{1024,0.001}^{8}$
$ID = 219,186$	3	236
$ID = 1,373,342$	6	19
$ID = 38,402$	11	262
$ID = 175,825$	21	160
$ID = 315,374$	29	200
$ID = 408,262$	104	147
$ID = 307,773$	200	71
$ID = 118,673$	359	193
$ID = 1,681,668$	542	359
$ID = 190,276$	1,010	1,011
$ID = 475,431$	1,024	1,019
$ID = 2,794$	1,041	1,039
$ID = 801,868$	1,100	1,100
$ID = 37$	85,214	84,348
$ID = 85,532$	92,560	90,146
$ID = 59,715$	604,202	602,963

In general, there are over 1,050 resources administering data points in the *immediate neighborhood* to the query point (i.e their data points are so close to the query point one cannot discern them from the query point by utilizing our visualization tool in its maximum zoom level).[189] Athens is a city of about 22,000 inhabitants which has a university of its own—which is most likely the reason for the high level of Twitter activity there. Since the interesting resources' indexed areas for both $\text{KDQT}_{128}^{256,1.0E-5}$ as well as $\text{QTMBR}_{1024,0.001}^{8}$ cannot be recognized by use of our visualization tool, Figure 51 shows the visualizations of some exemplarily selected resources in a different presentation format: The indexed areas are depicted as black surfaces, the dark-grey crosses depict the location of the query point, and the light-grey dots represent the resources' data points. The selected resources' ranks in the corresponding $\text{KDQT}_{128}^{256,1.0E-5}$ and $\text{QTMBR}_{1024,0.001}^{8}$ rankings are listed alongside some other exemplarily selected resources' ranks in Table 23.[190]

In general, the visualizations confirm that the $\text{KDQT}_{128}^{256,1.0E-5}$ summaries are spatially more accurate: While for $\text{QTMBR}_{1024,0.001}^{8}$, always at least one large indexed rectangular area is present, the indexed surface areas for $\text{KDQT}_{128}^{256,1.0E-5}$ are steadily spread over many smaller rectangles. Although this does not always result in a significantly smaller indexed surface area for $\text{KDQT}_{128}^{256,1.0E-5}$ (see e.g. resource 1,373,342 in Figure 51), the more arbitrary shape of the union of the indexed areas usually represents a better approximation of the data point clouds. In contrast, the comparatively large, single indexed areas for $\text{QTMBR}_{1024,0.001}^{8}$ show that a refinement with only one quantized MBR is too coarse in extreme situations such as for query 22 (with its many resources whose data points are *spread* over the spatially very confined area of Athens' city zone)—as it can happen that large parts of the basic quadtree cell are covered by the quantized MBR. Consequently, the $\text{KDQT}_{128}^{256,1.0E-5}$ ranking is much more precise *at the top ranks*: The exemplarily selected resources contributing to the query result all have significantly better ranks in the $\text{KDQT}_{128}^{256,1.0E-5}$ ranking (see Table 23 again). The distance of the 50th closest neighbor to query 22 is 1.70E-4 *dsu*, corresponding to 14.6 meters. A visualization of the final query ball

[189] In the *close neighborhood* of Athens (i.e. within a radius of 5 km), an amount of 1,081 resources administers data points. Aside from the approximately 1,050 resources administering data points in the immediate neighborhood, not many other resources administer data points in this region and thus, the resources' distances to the query point increase very quickly.

[190] Note that for the specified ranking positions, the directly represented resources have been included in the ranking, i.e. all resources have been ranked. Due to the different sets of summary-represented resources for $\text{KDQT}_{128}^{256,1.0E-5}$ and $\text{QTMBR}_{1024,0.001}^{8}$, a comparison of the ranking positions for specific resources would make no sense otherwise. Also, the inclusion of the directly represented resources is necessary for assessing how many resources administer data points immediately in Athens. Among the 1,050 top ranked resources for query 22, there are 309 directly represented resources for $\text{KDQT}_{128}^{256,1.0E-5}$ and 133 directly represented resources for $\text{QTMBR}_{1024,0.001}^{8}$.

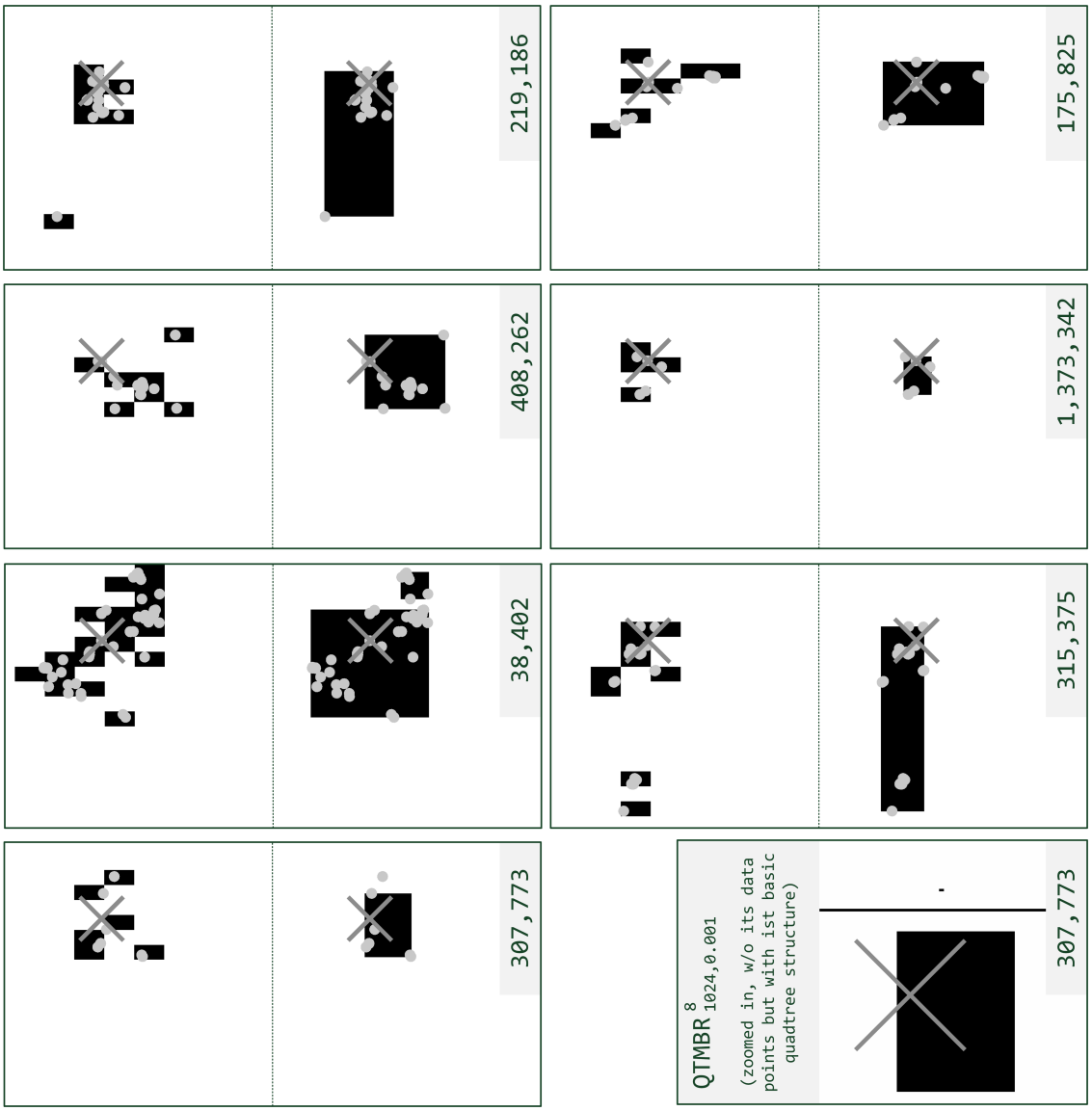


Fig. 51: Visualization of the $\text{KDQT}_{128}^{256,1.0E-5}$ and $\text{QTMBR}_{1024,0.001}^{8}$ summaries for exemplarily selected resources which administer data points in or near Athens, Ohio (T1 collection). The respective resource whose summaries are shown in each block is indicated by its ID in the light-grey field at the top right of the block. In each block, the respective $\text{KDQT}_{128}^{256,1.0E-5}$ ($\text{QTMBR}_{1024,0.001}^{8}$) summary is always on the left (right). Note that both resource 315,375 as well as resource 307,773 do not contribute to the result for query 22. In the bottom right block, the $\text{QTMBR}_{1024,0.001}^{8}$ summary of resource 307,773 is shown once more—without its data points but with the cell borders of its basic quadtree structure. In this visualization, the second, tiny indexed area of the $\text{QTMBR}_{1024,0.001}^{8}$ summary can be seen (which is covered by the corresponding data point in the block of resource 307,773). This indexed area corresponds to a cell of the internal quantization grid embedded into the exterior quadtree cell, i.e. it is the smallest indexable spatial unit for this quadtree cell.

containing all 50 nearest neighbors is shown in Figure 52. On the basis of these visualizations, it is clear now why for query 22, the resource rank-

ing works so much better for $\text{KDQT}_{128}^{256,1.0E-5}$ although the indexed areas for $\text{KDQT}_{128}^{256,1.0E-5}$ *as a whole* are often similar in shape to those indexed for $\text{QTMBR}_{1024,0.001}^{8}$.

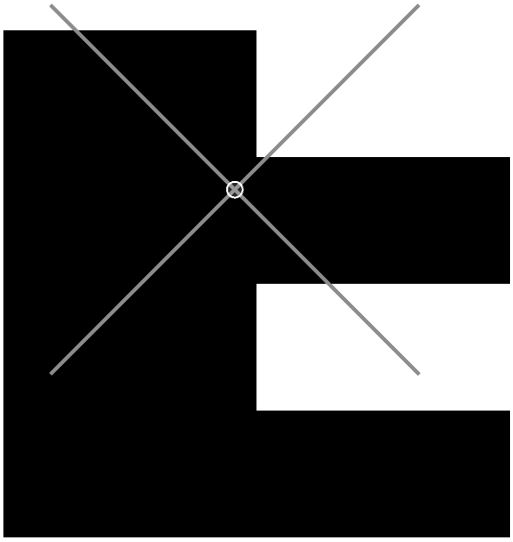


Fig. 52: Visualization of the final query ball containing all top 50 data points for query 22. The query ball is the tiny, white-bordered circle centered at the intersection of the cross. The black surface is part of the indexed area for the $\text{KDQT}_{128}^{256,1.0E-5}$ summary of resource 219,186 (see Figure 51).

In Table 23, it can also be seen that the ranking positions greater 1,000 are astonishingly similar for the $\text{KDQT}_{128}^{256,1.0E-5}$ and the $\text{QTMBR}_{1024,0.001}^{8}$ ranking. This is because the indexed areas of the $\text{KDQT}_{128}^{256,1.0E-5}$ summaries (as just mentioned) as a whole often take a similar form to the coarser indexed areas of the $\text{QTMBR}_{1024,0.001}^{8}$ summaries. Consequently, if the query point is not located in an indexed area or not located right between the 'indexed area cloud' (as e.g. for the $\text{KDQT}_{128}^{256,1.0E-5}$ summary of resource 307,773 in Figure 51), their closest borders are oftentimes similarly close to the query point. With regard to the aspect ratios of the indexed areas and on basis of assessing the visualizations, it does not seem like the $\text{QTMBR}_{1024,0.001}^{8}$ summaries are 'handicapped' to a significant extent—it is simply a question of the accuracy of the quantized MBRs used for refinement. In this regard, the maximum accuracy is not a problem for $\text{QTMBR}_{1024,0.001}^{8}$: Its smallest indexable spatial unit in Athens is much smaller compared to $\text{KDQT}_{128}^{256,1.0E-5}$ (see Figure 51 at the bottom right). Nevertheless, when the data points are scattered over an occupied cell, the area indexed by a single quantized MBR becomes too large. It is the same problem the MBR approach has—only on a smaller scale. With its quadtree refinement, $\text{KDQT}_{128}^{256,1.0E-5}$ is able to divide the same data point set into several mutually independent parts, in total resulting in more accurate descriptions—even if its smallest indexable spatial unit is significantly larger compared to $\text{QTMBR}_{1024,0.001}^{8}$. The

quadtree refinement with several areas as opposed to a quantized MBR refinement of just one area is also the reason why the 'number of indexed areas' is almost 50% greater for $\text{KDQT}_{128}^{256,1.0E-5}$ even though $\text{QTMBR}_{1024,0.001}^{8}$ features a higher share of summary-represented resources. Note that also in all the example visualizations of Figure 51, the amount of indexed areas is greater for $\text{KDQT}_{128}^{256,1.0E-5}$.

Regarding the impact of the directly represented resources for query 22, we assume it to be negligible: Only 1 of the top 50 data points originates from a directly represented resource for both $\text{KDQT}_{128}^{256,1.0E-5}$ and $\text{QTMBR}_{1024,0.001}^{8}$. One the other hand, when *all* resources are ranked (including the directly represented resources), there are 309 directly represented resources among the top 1,050 ranks for $\text{KDQT}_{128}^{256,1.0E-5}$ but only 133 for $\text{QTMBR}_{1024,0.001}^{8}$. Hence, as the set of resources constituting the top 1,050 ranks of the $\text{KDQT}_{128}^{256,1.0E-5}$ and $\text{QTMBR}_{1024,0.001}^{8}$ rankings is very similar, this set includes about 170 resources which are directly represented for $\text{KDQT}_{128}^{256,1.0E-5}$ but summary-represented for $\text{QTMBR}_{1024,0.001}^{8}$. The remaining question is how many of these resources cannot be pruned before being contacted for $\text{QTMBR}_{1024,0.001}^{8}$ because of the spatial inaccuracy of the summary representation (compared to a direct representation). Due to our software architecture and the general complexity of our experimental setup, this would be very laborious to reconstruct. Hence, instead of reconstructing the situation at this point, we reiterate the comparison of $\text{KDQT}_{128}^{256,1.0E-5}$ and $\text{QTMBR}_{1024,0.001}^{8}$ for query 22 in section 8.3, where the direct representation of resources is disallowed for the `T1` collection.

Note that the given explanations are only valid for regions where the basic k-d space partitioning is decently accurate. For regions featuring coarser basic k-d cells, the results may look different with regard to the spatial accuracy between the $\text{KDQT}_{128}^{256,1.0E-5}$ and $\text{QTMBR}_{1024,0.001}^{8}$ summaries (also already see Figure 54 and Figure 55). Nevertheless, these possible differences do not have a negative impact on the $\text{KDQT}_{128}^{256,1.0E-5}$ performance due to the acceptable greater coarseness of summaries in low-density regions (as discussed in section 8.1.2). This is also substantiated by the individual comparison of the respective *rfc* values which occur for the 50 queries between $\text{KDQT}_{128}^{256,1.0E-5}$ and $\text{QTMBR}_{1024,0.001}^{8}$—where $\text{KDQT}_{128}^{256,1.0E-5}$ is only 0.09 ‰ worse at most.

Concerning the area-related key figures, it is unusual that on the one hand, the 'data space coverage' of $\text{KDQT}_{128}^{256,1.0E-5}$ is ~3 times greater compared to $\text{QTMBR}_{1024,0.001}^{8}$ but on the other hand, the 'overlap between summaries' for $\text{QTMBR}_{1024,0.001}^{8}$ is ~1.5 times greater than for $\text{KDQT}_{128}^{256,1.0E-5}$. Hence, the ratio of the 'overlap between summaries' / 'data space coverage' is ~4.5 times greater for $\text{QTMBR}_{1024,0.001}^{8}$. Basically, one would expect the overlap to grow disproportionately with the 'data space coverage' (which is generally confirmed by the other numbers listed in Table 18 and similar tables like Table 13 on page 187). Since the indexed surface areas of spatially narrow resources are so small, the major part of the 'overlap between sum-

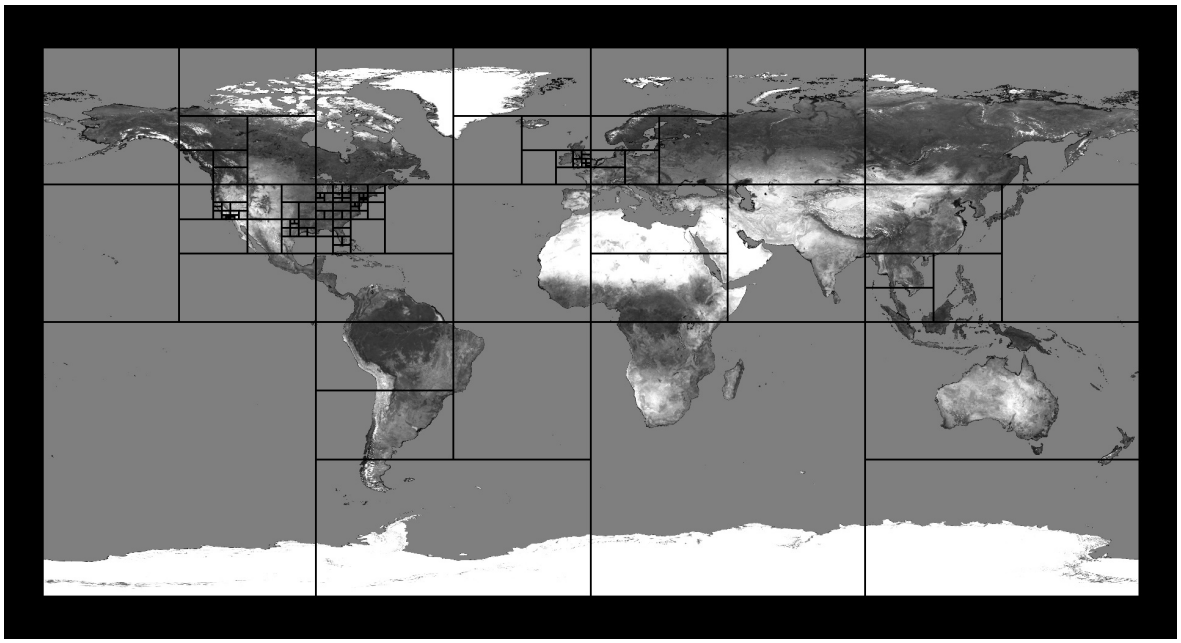


Fig. 53: Visualization of the resulting global k-d space partition for $n = 128$ (T1 collection).

maries' and the 'data space coverage' originates from the summaries of spatially spread resources.[191] For the regions in the northern hemisphere which are densely populated with data points, the basic k-d resolution for $\text{KDQT}_{128}^{256,1.0E-5}$ is relatively detailed (see Figure 53). As each occupied k-d cell is refined with an internal quadtree of up to 256 cells, mostly small indexed areas result for the northern hemisphere—even when the resource is spatially very spread. Also compare Figure 51 as an example (on a smaller scale) for the indexing of spatially spread data points within in a cell. For the southern hemisphere, on the other hand, the basic k-d cells are very large. Even with the detailed quadtrees as refinement, large indexed areas can result. As an example, consider Figure 54. It shows the $\text{KDQT}_{128}^{256,1.0E-5}$ summaries of three of the spatially most spread resources of the T1 collection, featuring data points all over the data space. In the northern hemisphere, the resulting indexed areas are small whereas in the southern hemisphere, the resulting indexed areas can be very large in comparison. In contrast, for $\text{QTMBR}_{1024,0.001}^{8}$, the spatial resolution for spread resources is basically the same everywhere. See Figure 55 as an example: The indexed areas have roughly the same sizes in both hemispheres. Notably, the issue of the refinement with quantized MBRs is visible for $\text{QTMBR}_{1024,0.001}^{8}$, again: large quantized MBRs can occur which cover almost the entire cell

[191] As an illustrative example: With its surface area of 2.64E-5 dsu^2, the $\text{KDQT}_{128}^{256,1.0E-5}$ summary of resource 219,186 features the 326,228th largest surface area for $\text{KDQT}_{128}^{256,1.0E-5}$ (alongside 1,212 other resources) and the T1 collection. If this was the average surface area of a $\text{KDQT}_{128}^{256,1.0E-5}$ *summary*, the data space coverage would be $2.64E{-}5\ dsu^2 \cdot (2,491,785 \cdot (1 - 0.5123))/64,800\ dsu^2 = 4.95E{-}4$, i.e. about 283 times smaller than the actual value for $\text{KDQT}_{128}^{256,1.0E-5}$.

they are embedded into. In Figure 55, this occurs quite frequently, for both the northern as well as the southern hemisphere.

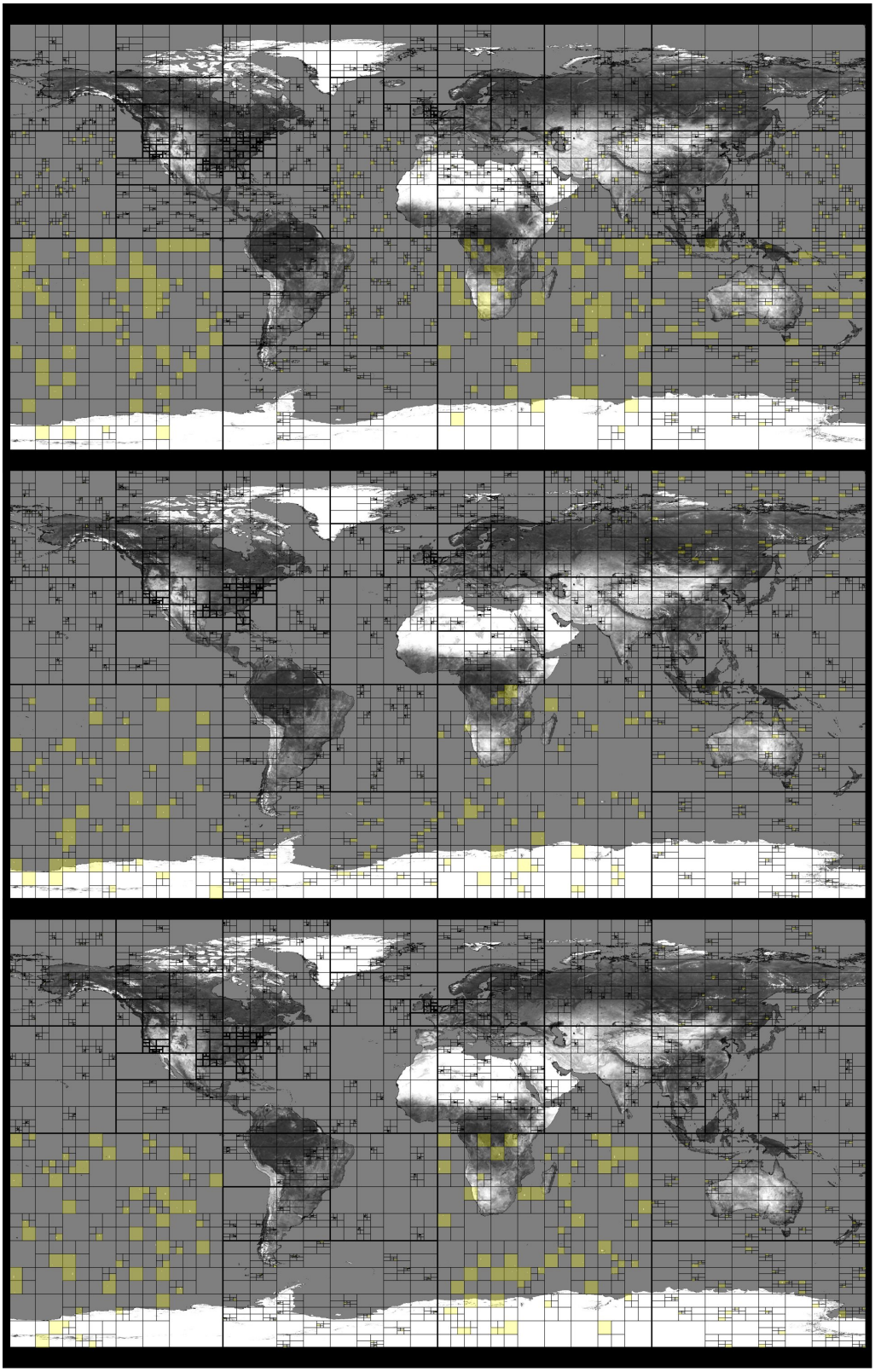


Fig. 54: Visualization of the $\text{KDQT}_{128}^{256,1.0E-5}$ summaries for resource 37 (top, `lq-z`-coded, 1,764 B), resource 85,532 (middle, `lq-z`-coded, 1,395 B), and resource 59,715 (bottom, `lq-z`-coded, 1,349 B) of the `T1` collection.

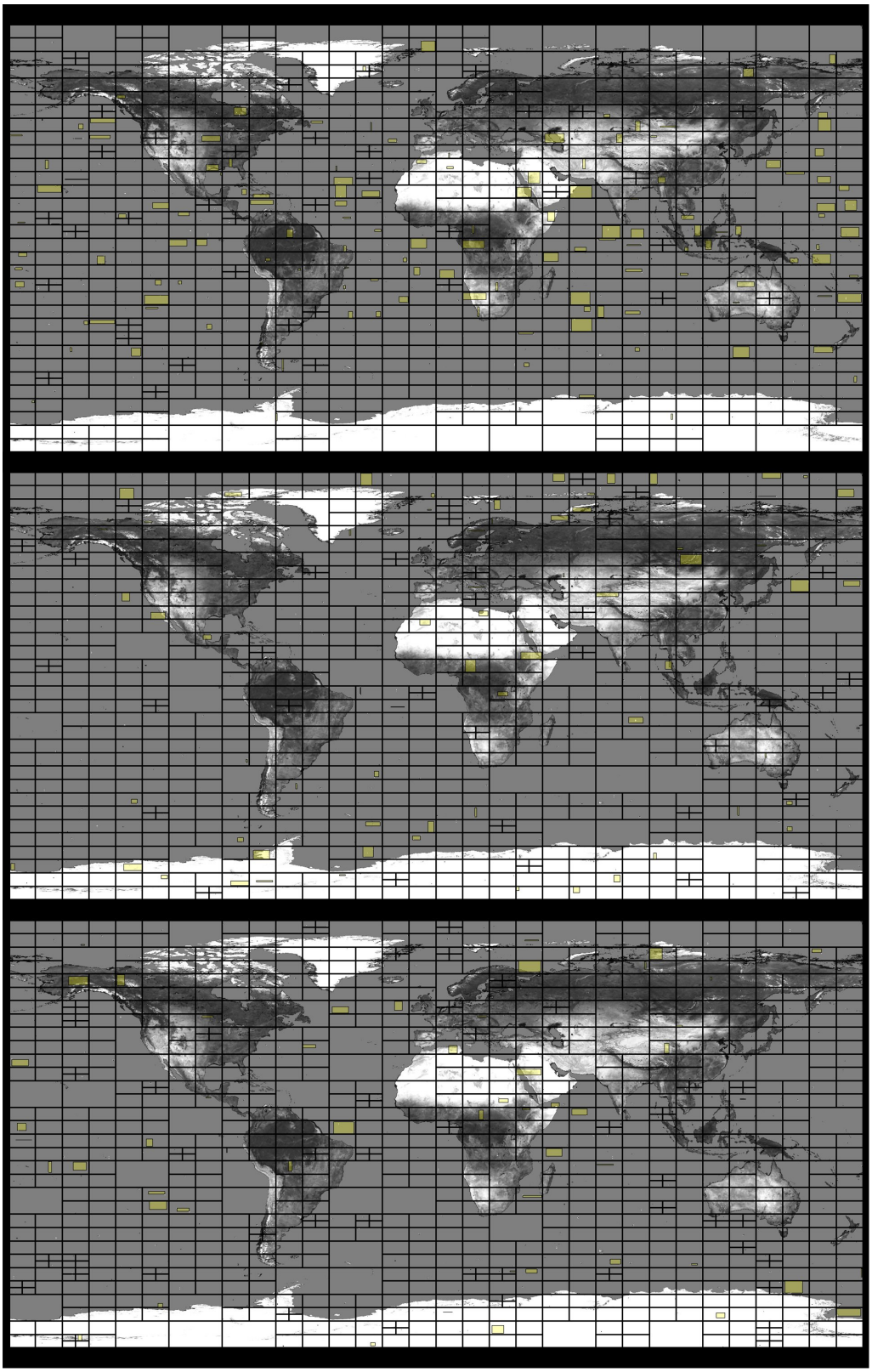


Fig. 55: Visualization of the $\text{QTMBR}^{8}_{1024,0.001}$ summaries for resource 37 (top, `cblq-nz`-coded, 2,263 B), resource 85,532 (middle, `cblq-nz`-coded, 1,863 B), and resource 59,715 (bottom, `cblq-nz`-coded, 1,799 B) of the `T1` collection.

Hence, for $\text{KDQT}^{256,1.0E-5}_{128}$, the indexed areas for *spatially spread resources* are small in the northern hemisphere while in the southern hemisphere,

they can be very large. For $\text{QTMBR}^{8}_{1024,0.001}$, the indexed areas for these resources oftentimes are of medium size, independent of the hemisphere they are located in. Since most data points (and therefore resources) are located in the northern hemisphere, this is disadvantageous with regard to the 'overlap between summaries' for $\text{QTMBR}^{8}_{1024,0.001}$: For spatially spread resources, it features significantly larger single areas than $\text{KDQT}^{256,1.0E-5}_{128}$ in the 'resource-stuffed' northern hemisphere. In the very end, this results in a greater 'overlap between summaries' for $\text{QTMBR}^{8}_{1024,0.001}$. On the other hand, for these resources, the $\text{QTMBR}^{8}_{1024,0.001}$ summaries generally cover less total surface area than the $\text{KDQT}^{256,1.0E-5}_{128}$ summaries, nonetheless. Therefore, the 'data space coverage' of $\text{QTMBR}^{8}_{1024,0.001}$ is smaller. See Table 24 for the concrete surface area numbers of examplarily selected resources (including the resources displayed in Figure 54 and Figure 55). There, the resources are sorted in descending order by their surface areas. It shows that the greatest surface areas of $\text{KDQT}^{256,1.0E-5}_{128}$ are significantly greater than those of $\text{QTMBR}^{8}_{1024,0.001}$.

Table 24: Excerpt of the ranking by surface area for the $\text{KDQT}^{256,1.0E-5}_{128}$ respectively the $\text{QTMBR}^{8}_{1024,0.001}$ summaries of the T1 collection.

rank	$\text{KDQT}^{256,1.0E-5}_{128}$		$\text{QTMBR}^{8}_{1024,0.001}$	
	resource	surface area [in dsu^2]	resource	surface area [in dsu^2]
1	37	3,900.6	37	1,463.1
2	59,715	2,476.8	85,532	618.5
3	85,532	1,727.0	59,715	564.6
...				
10	37,804	25.65	3,980	19.33
...				
100	780,894	0.11	57,978	1.82E-2
...				
1,000	232,080	3.82E-3	164,598	1.72E-3
...				
2,400	97,901	1.25E-3	116,199	1.26E-3
...				
10,000	114,272	1.96E-04	17,497	7.32E-04
...				
100,000	438,705	5.66E-05	624,175	1.93E-04
...				
1,000,000	1,165,196	7.54E-06	663,901	5.94E-08

Figure 56 shows all the resources sorted in descending order according to their summaries' surface areas (for both $\text{KDQT}^{256,1.0E-5}_{128}$ and $\text{QTMBR}^{8}_{1024,0.001}$). Due to the Zipf distribution of the surface area sizes, the surfaces quickly become very small. Because of this rapid decrease, the re-

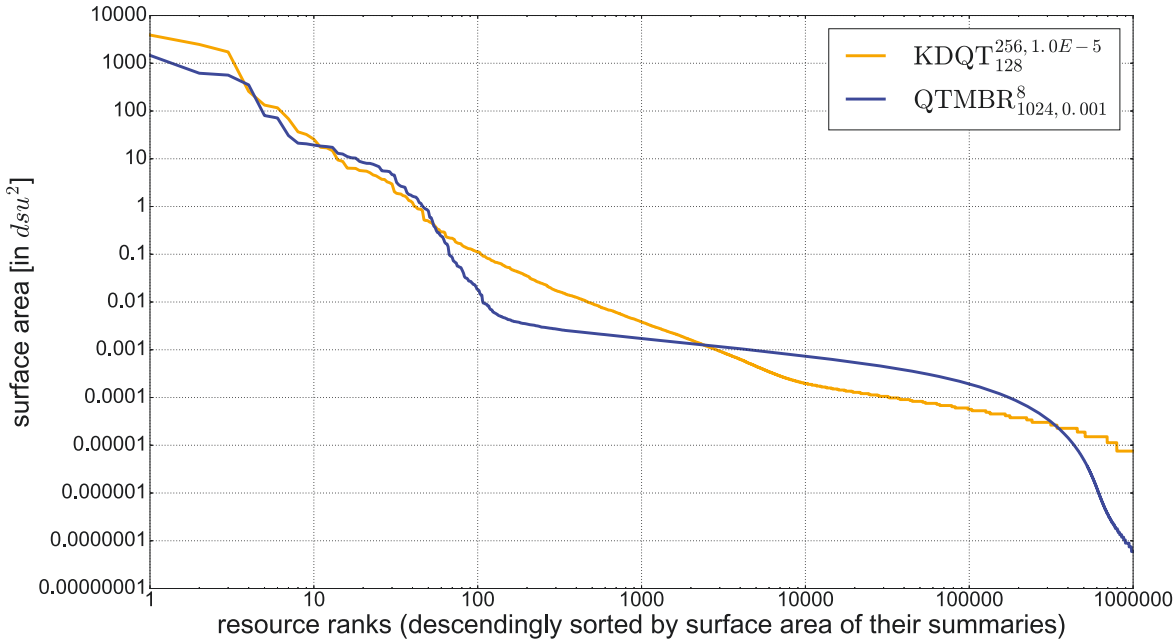


Fig. 56: Distribution of the summary-represented resources' surface areas for both KDQT$^{256,1.0E-5}_{128}$ and QTMBR$^{8}_{1024,0.001}$. On the x-axis, the resources are sorted in descending order by their summaries' indexed surface area. Both x-axis and y-axis are log-scaled.

sources further back in the ranking do not contribute much surface area to the total amount.[192] Hence, the 'data space coverage' is mainly determined by the surface areas of the spatially spread resources. In Figure 56, it can be seen that the QTMBR$^{8}_{1024,0.001}$ summaries almost continuously feature smaller surface areas for about the respective 2,400th greatest summaries (rank 2,400). From then to about rank 340,000, the KDQT$^{256,1.0E-5}_{128}$ summaries have smaller surface areas, before the QTMBR$^{8}_{1024,0.001}$ summaries again feature smaller surface areas until the end.[193] Hence, once again, the line chart of Figure 56 shows that the smallest indexable spatial unit for QTMBR$^{8}_{1024,0.001}$ is significantly smaller. However, its cell refinement with the quantized MBRs oftentimes leads to larger indexed surface areas compared to KDQT$^{256,1.0E-5}_{128}$. This happens when data points are widely spread over the basic cell to refine, obviously.
All in all, KDQT$^{256,1.0E-5}_{128}$ and QTMBR$^{8}_{1024,0.001}$ are the best techniques at 'MBRsize+'. KDQT$^{256,1.0E-5}_{128}$ offers the best overall query performance of all techniques. The slightly worse QTMBR$^{8}_{1024,0.001}$, on the other, hand has 32.7% less indexed areas than KDQT$^{256,1.0E-5}_{128}$. Hence, the costs of the initial ranking are significantly lower for QTMBR$^{8}_{1024,0.001}$. As a side note, it must be mentioned that even though it was specified at the beginning of sec-

[192] For KDQT$^{256,1.0E-5}_{128}$ (QTMBR$^{8}_{1024,0.001}$), the 100 greatest surface areas account for 99.47% (98.18%) of the 'data space coverage'.
[193] Note that for convenience, both rankings are limited to 1,000,000 resources. In total, there are 1,215,362 (1,350,322) summary-represented resources for KDQT$^{256,1.0E-5}_{128}$ (QTMBR$^{8}_{1024,0.001}$).

tion 8.1 that the *rds* values are considered to be equivalent, $\mathrm{KDQT}_{128}^{256,1.0E-5}$ requires 1.05 B *rds* more than $\mathrm{QTMBR}_{1024,0.001}^{8}$, after all. Using this additional memory, the *rfc* values of $\mathrm{QTMBR}_{c,a}^{b}$ could certainly be improved a bit. Nevertheless, based on the visualizations in Figure 51, it seems questionable whether the same performance as for $\mathrm{KDQT}_{128}^{256,1.0E-5}$ can be achieved for $\mathrm{QTMBR}_{c,a}^{b}$ with only 1.05 B *rds* in addition.

8.2. Evaluation of the F Collection

In this section, the F collection is evaluated for the selected approaches. For each approach, we test the same parameterizations as for the T1 collection. Figure 57 and Figure 58 show the resulting Skylines for the F collection. Similar to the T1 collection, there are bigger differences between the approaches for low *rds* values and a general convergence of the Skylines for high *rds* values. The resulting *rfc* values are much higher for the F collection since the set of resources has a cardinality of only 5,951 (~419 times less than for the T1 collection) but still, 7.74 resources contribute to the query result on average (instead of 12.92 resources for a cardinality of 2,491,785).[194] In addition, the resources are also spatially more spread in tendency (generally complicating the selection task). Furthermore, the assignment of data points to resources is not as extremely Zipf-distributed as for the T1 collection: the share of very small resources is significantly lower for the F collection (also see Table 2 on page 114). As a consequence of both the spatial spreading and the assignment, the respective Skylines start and end at higher *rds* values, in general. For example, the KDQT$^{c,a}_{n}$ Skyline ranges from 32.95 B *rds* to 61.71 B *rds* for the T1 collection but from 35.82 B *rds* to 170.41 B *rds* for the F collection.

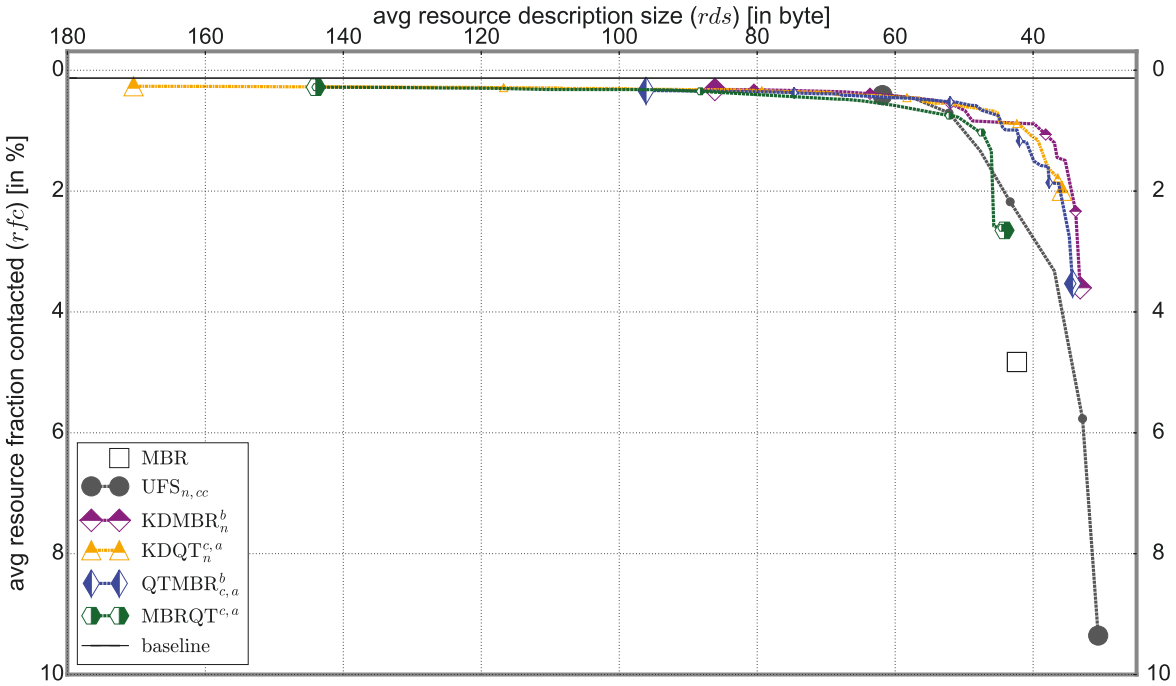


Fig. 57: Full Skylines of the six approaches selected for further evaluation and the F collection.

The sole exception from this rule is the UFS$_{n,cc}$ Skyline which starts sooner compared to the T1 collection (30.57 B *rds* versus 31.42 B *rds*, both for UFS$_{32,cc}$). Usually, spatially more spread resources (as in the F collection) should result in greater *rds* values since more bins in the bit vectors of

[194] Also see section 6.2 and section 6.3.

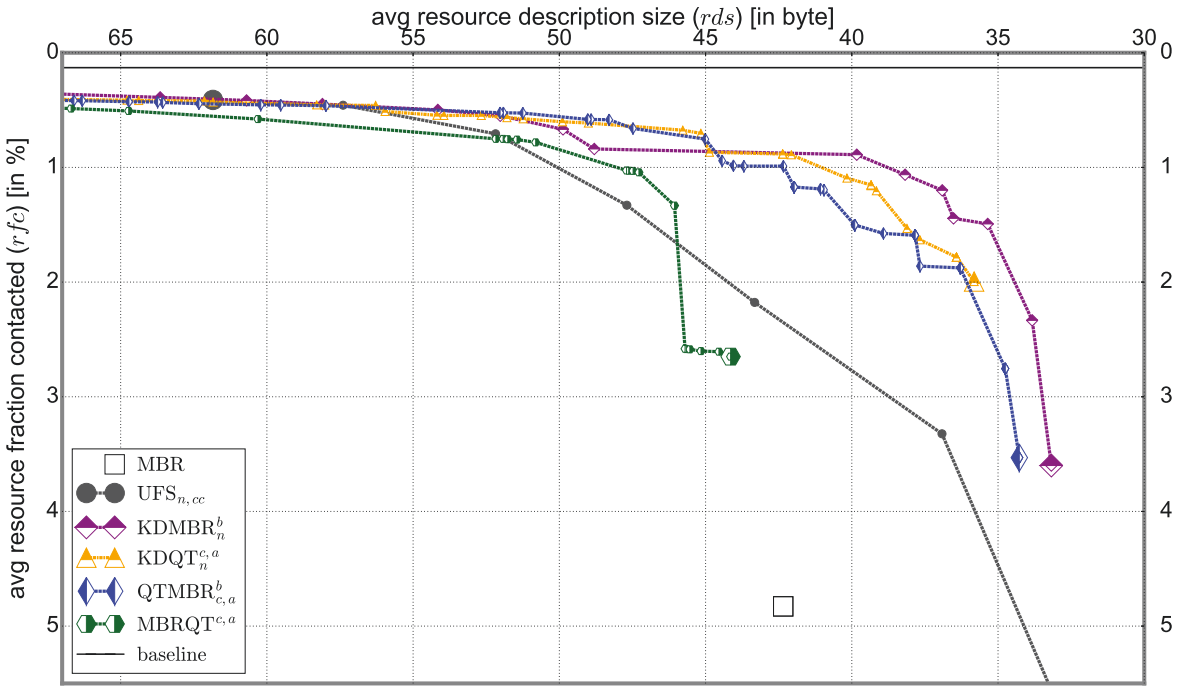


Fig. 58: Zoomed in depiction of the Skylines of the approaches selected for further evaluation and the F collection.

the UFS$_{n,cc}$ summaries are set to 1. For $n > 32$, the UFS$_{n,cc}$ *rds* values are then indeed greater for the F collection compared to the T1 collection. For $n = 32$, it is simply that the IDs of the cells are more favorably assigned for the F collection, i.e. more bit vectors can be clipped prematurely. See Table 25 for a comparison of UFS$_{32,cc}$ for both collections. The average number of occupied subspaces (and thus the average number of bins set to '1') is significantly greater for the F collection (1.69) compared to the T1 collection (1.06). Nevertheless, it can be seen that while the minimally and maximally occuring *rds* values are the same, the quantile and median *rds* values *favor* the F collection. In terms of a visual comparison of the respective space partitions and considering the different global distribution of the data points, there are also no indications why the UFS$_{32,cc}$ summaries are smaller for the F collection than for the T1 collection. Hence, it has to be the assignment of the cell IDs which causes the observed anomaly.
Overall, for the F collection, the *rds* range of the Skylines is from 30.57 B *rds* (UFS$_{32,cc}$) to 170.41 B *rds* (KDQT$^{512,1.0E-8}_{8192}$) while the *rfc* range is from 9.36% (UFS$_{32,cc}$) to 0.27% (KDQT$^{512,1.0E-8}_{8192}$).
In general, the approaches being based on global space partitioning (UFS$_{n,cc}$, KDMBR$^{b}_{n}$, and KDQT$^{c,a}_{n}$), in relative terms, show better results compared to the T1 collection. For example, UFS$_{n,cc}$ is now significantly better than the MBR approach at 'MBRsize'—for the T1 collection, it was the other way round. Or, the Skylines of both KDMBR$^{b}_{n}$ and KDQT$^{c,a}_{n}$ show a significantly improved course versus the QTMBR$^{b}_{c,a}$ Skyline. This is easily explained: The number of resources is much lower for the F collection (only 5,951 resources) compared to the T1 collection (2,491,785 resources). Con-

Table 25: Comparison of the descriptive statistic *rds* values and the average number of occupied cells for $UFS_{32,cc}$ and the F respectively T1 collection. Remember that for the F collection, the results are averaged over four runs using different seeds, leading to the occuring *.25 respectively *.75 values. For both collections, 100% of the resources are summary-represented for $UFS_{32,cc}$.

$UFS_{32,cc}$	F collection	T1 collection
avg *rds* [in byte]	30.57	31.43
min *rds* [in byte]	29	29
25%-quant. *rds* [in byte]	29.75	31
median *rds* [in byte]	30.75	32
75%-quant. *rds* [in byte]	31.25	32
max *rds* [in byte]	33	33
avg number of occupied cells	1.69	1.06

sequently, the number of resources administering data points in a single cell of the global space partition is usually much lower. Hence, the global space partition by itself is simply much better suited for obtaining a small candidate set of potentially relevant resources. Nevertheless, $QTMBR^{b}_{c,a}$ is still significantly better than $UFS_{n,cc}$, and better than $KDMBR^{b}_{n}$ between ~44.5 B *rds* and ~54.0 B *rds* respectively better than $KDQT^{c,a}_{n}$ between ~48.0 B *rds* and ~56.0 B *rds*. This shows that also for the F collection, excellent results can be obtained by utilizing local space partitioning as a description base—even though the general setting (fewer, but spatially more spread resources) is less favorable for the resource-individual quadtree space partitioning. For $KDMBR^{b}_{n}$, once again, there is a phase between $KDMBR^{8}_{64}$ (*rds*: 39.83 B, *rfc*: 0.89%) and $KDMBR^{3}_{256}$ (*rds*: 48.81 B, *rfc*: 0.84%) where significant additional *rds* investments result in only marginal *rfc* improvements. At this point, we do not go into detail like for the T1 collection in section 7 and do not e.g. investigate which parameter values are present how often in the respective Skyline parameterizations. Nevertheless, this observation for $KDMBR^{b}_{n}$ shows that at least some approach-specific characteristics which are evident for the T1 collection also apply for the F collection.

All in all, the three best approaches for the F collection are $KDMBR^{b}_{n}$, $KDQT^{c,a}_{n}$, and $QTMBR^{b}_{c,a}$. Again, $KDMBR^{b}_{n}$ is best for very small *rds* values but falls off a little between 40.0 B *rds* and 50.0 B *rds*. Once more, the selectivity of the MBR approach can be significantly improved when utilizing the same amount of storage space—up to 82% ($KDQT^{32,1.0E-5}_{32}$, also see Table 26). Despite being constrained by its basic full-precision MBR, $MBRQT^{c,a}$ shows decent results and at the least is better than $UFS_{n,cc}$ between ~46.0 B *rds* and ~51.7 B *rds*. Nevertheless, it is less suitable than the other hybrid approaches in this comparison.

Table 26: Listing of various key figures alongside the resulting *rds* and *rfc* values for the F collection and the benchmarked techniques at 'MBRsize'.

technique → / key figure ↓	MBR	$UFS_{256,cc}$	$KDMBR^{8}_{64}$	$KDQT^{32,1.0E-5}_{32}$	$QTMBR^{4}_{64,1.0}$	$MBRQT^{16,1.0}$
overlap between summaries [in dsu^2]	275,814,821	119,781,542	613,718	151,838	46,080	18,307,724
data space coverage	56.41	49.73	1.83	0.56	0.23	10.63
surface area per summary [in dsu^2]	868.0	706.9	22.49	6.68	2.45	163.5
number of indexed areas	4,211	14,200	9,666	23,441	15,977	10,145
aspect ratio (mean) [:1]	7.63	-	2.23	1.46	2.18	5.98
aspect ratio (median) [:1]	2.00	-	1.67	1.00	2.00	2.07
directly represented resources [in %]	29.24	22.97	11.50	8.38	0.00	29.24
avg *rfc* [in %]	4.83	2.18	0.89	0.88	0.99	2.65
avg *rds* [in byte]	42.35	43.32	39.83	42.36	42.34	44.14

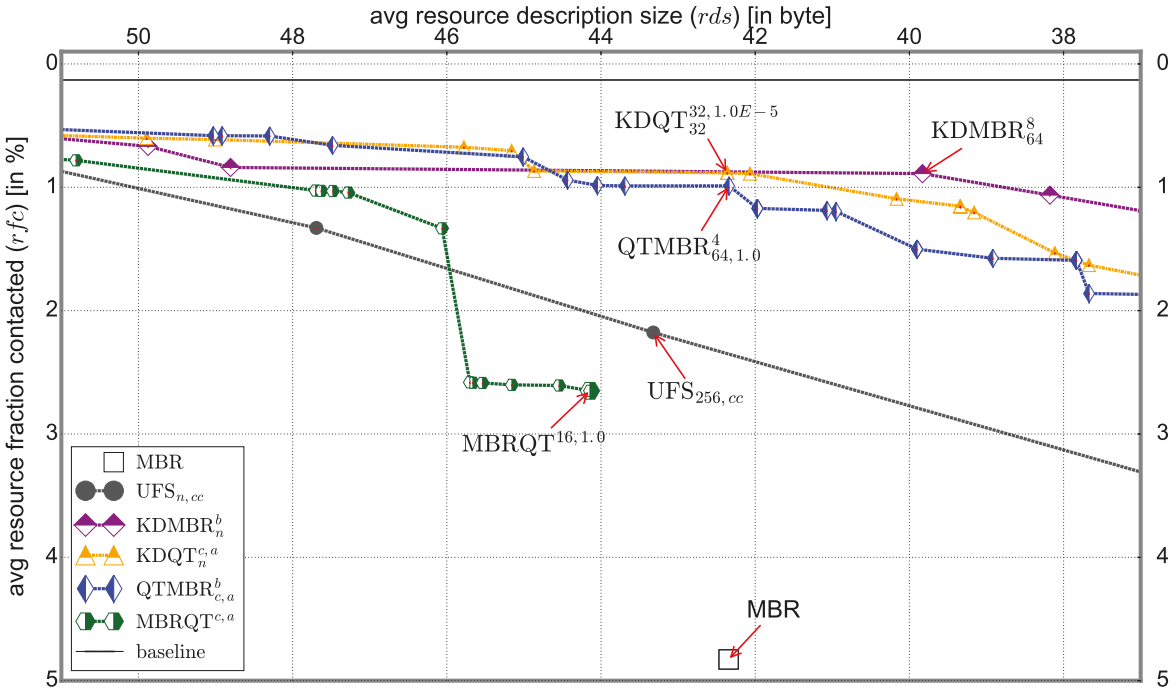


Fig. 59: Zoomed in version of Figure 57, showing the Skylines for the F collection around 'MBRsize'. The respective Skyline parameterizations of the different approaches at 'MBRsize' are marked with red arrows.

In Table 26, the key figures for the techniques at 'MBRsize' are depicted (also see Figure 59). Obviously, the numbers for the 'overlap between sum-

maries', the 'data space coverage', and the 'directly represented resources' are much lower compared to the T1 collection (see Table 12 on page 183). On the other hand, the 'surface area per summary'-values are much greater for the F collection. Once more, this proves that the resources of the F collection are spatially more spread on average. Concerning the area-related key figures, it is evident again that the 'overlap between summaries' and the 'data space coverage' do not necessarily coincide with the resulting *rfc* values when there are approaches involved which are based on global space partitioning: Despite featuring ~13.7 times less overlap, $\text{QTMBR}^4_{64,1.0}$ has a slightly worse *rfc* value than KDMBR^8_{64} (0.99% versus 0.89%). Once again, the reason for the significantly greater 'overlap between summaries' and 'data space coverage' for KDMBR^8_{64} (and $\text{KDQT}^{32,1.0E-5}_{32}$) as opposed to $\text{QTMBR}^4_{64,1.0}$ is the varying resolution of the basic k-d space partition which has already been discussed in section 8.1.2. Also see Figure 60 for a visualization of the basic k-d space partition resulting for the F collection when $n = 64$. Note that in contrast to the T1 collection, Europe and Japan have much more weight in the F collection (and consequently, the basic k-d resolution is now much finer in these regions).

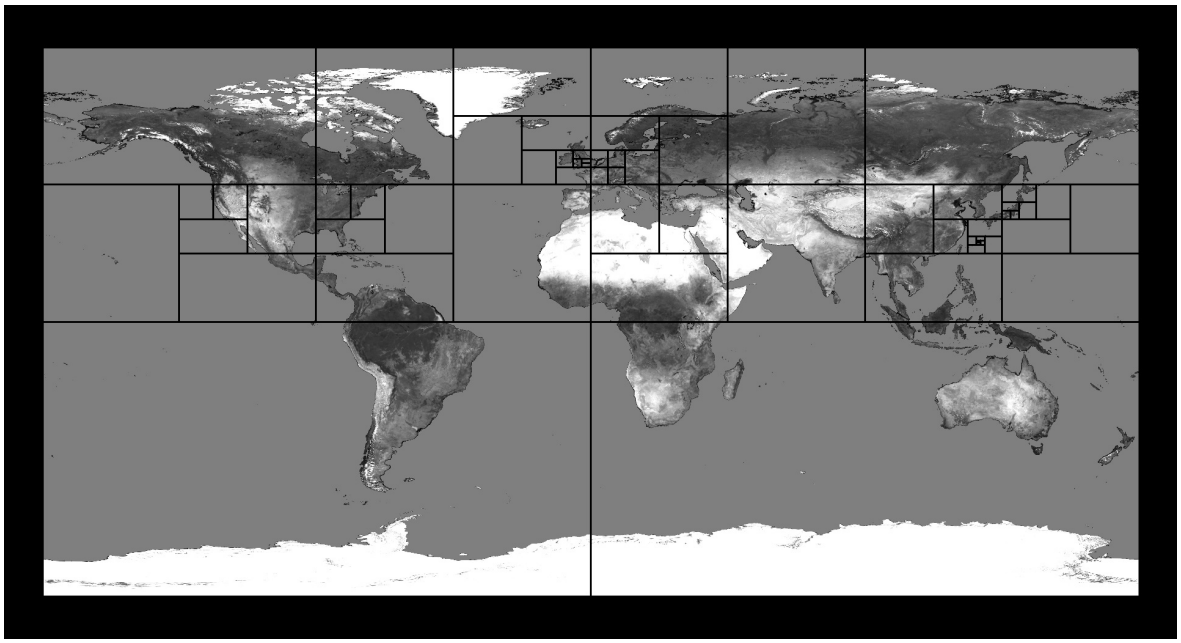


Fig. 60: Visualization of the resulting global k-d space partition for $n = 64$ (F collection).

Anew, we conduct a qualitative analysis to figure out the reasons behind the resulting *rfc* values for KDMBR^8_{64} and $\text{QTMBR}^4_{64,1.0}$. Generally, the area-related key figures (see Table 26) are tremendously in favor of $\text{QTMBR}^4_{64,1.0}$. Looking at the single queries, $\text{QTMBR}^4_{64,1.0}$ is better for 26 of the queries while KDMBR^8_{64} is better for 20 queries. For 4 queries, the performance is the same. In total, the *rfc* value averaged over all 50 queries is better for KDMBR^8_{64}. Again, it is striking that $\text{QTMBR}^4_{64,1.0}$ is consistently worse for queries where a lot of resources have to be contacted during query

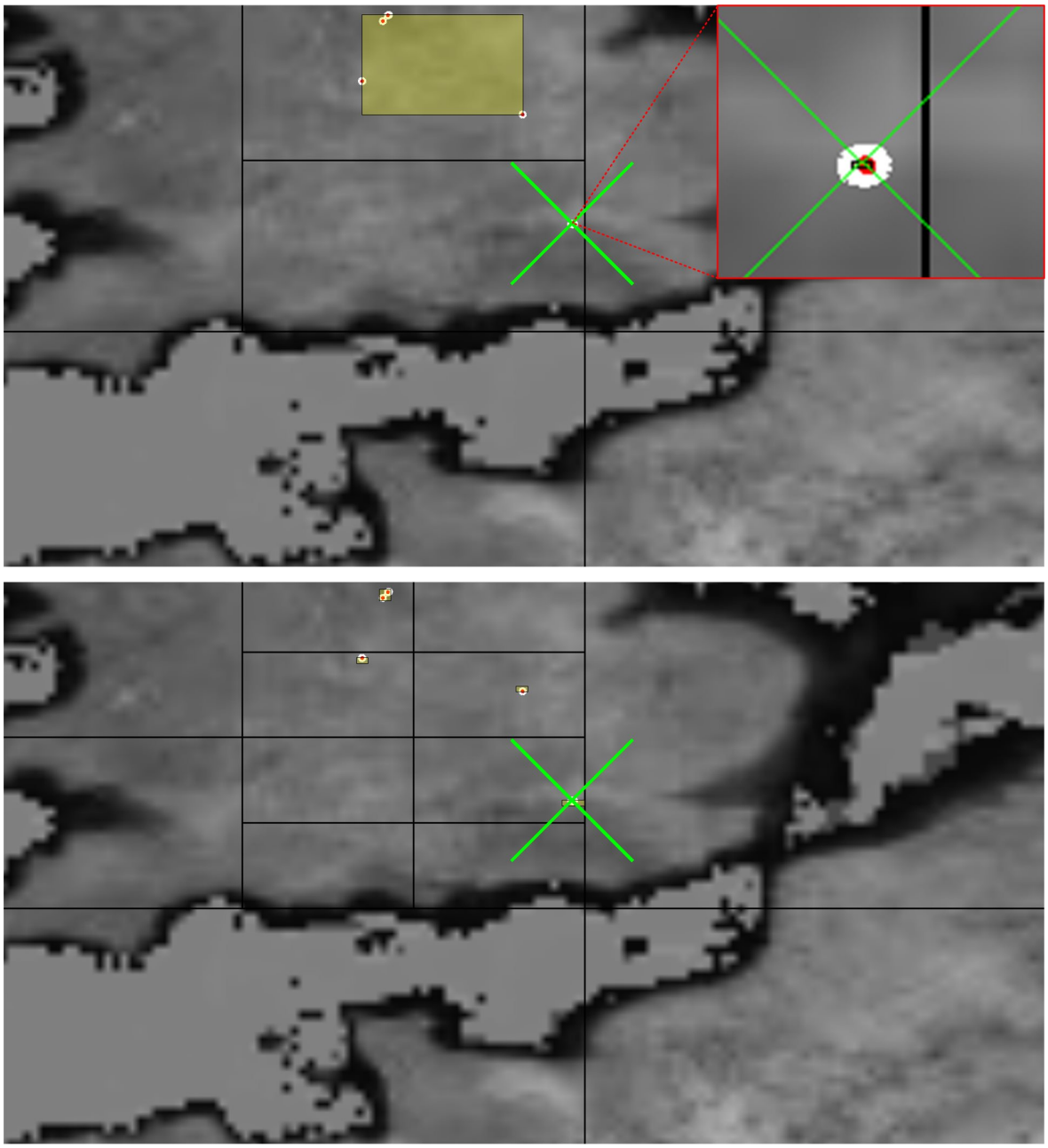


Fig. 61: Depiction of the KDMBR^{8}_{64} summary (top) and the $\text{QTMBR}^{4}_{64,1.0}$ summary (bottom) for resource 5,269 (F collection), visualization tool in its maximum zoom level. The green cross depicts the location of query 26 (London, UK).

processing. Table 27 lists the resource fractions contacted for three exemplarily depicted queries alongside the corresponding 'real-world' locations. Located in London, query 26 is the query where the second most resources are contacted for both KDMBR^{8}_{64} as well as $\text{QTMBR}^{4}_{64,1.0}$. Here, KDMBR^{8}_{64} is significantly superior. Likewise in New York City for the T1 collection, the maximum accuracy of the KDMBR^{8}_{64} summaries in London is significantly higher than for the corresponding $\text{QTMBR}^{4}_{64,1.0}$ summaries (see Figure 61).

[195] Remember that the basic quadtree space partition is resource-individual for $\text{QTMBR}^{b}_{c,a}$.

Table 27: Comparison of the results for KDMBR^8_{64} and $\text{QTMBR}^4_{64,1.0}$ for exemplarily selected query points (F collection).

query	x	y	'real-world' location	KDMBR^8_{64} *rfc* [in %]	$\text{QTMBR}^4_{64,1.0}$ *rfc* [in %]
$ID = 22$	-51.20	-30.03	Porto Alegre, Brasil	0.20	0.17
$ID = 26$	-0.10	51.52	London, UK	2.37	5.21
$ID = 37$	-121.75	38.58	Sacramento, California	0.62	0.24

This is due to the good basic k-d resolution in this region and the 8 bit per bound for the quantized MBRs. The usually resulting basic quadtree cells[195] are even smaller than the k-d cell in which London is located in. Nevertheless, the minimally resulting surface areas are bigger since only 4 bit are used per quantized bound. Generally, there are a lot of resources for which their data points are spatially very *concentrated* in London, leading to the superiority of KDMBR^8_{64} for query 26.

For query 22 (located in Porto Alegre, Brasil), it is the other way round concerning the surfaces of the indexed areas: the *maximum accuracy* of the $\text{QTMBR}^4_{64,1.0}$ summaries around the query point is significantly higher (see Figure 62). This is because the global k-d space partition, again, only features two cells for the southern hemisphere (see Figure 60). Nevertheless, this discrepancy is not reflected in the resulting *rfc* values which are almost the same. There are only few resources in the F collection administering their data points around Porto Alegre. Hence, at least as long as the spatial accuracy of their summaries is halfway decent, there are not many candidate resources which can erroneously be placed at top positions of the ranking. Thus, the number of unnecessarily contacted resources can be kept within limits despite suboptimal spatial accuracy. In fact, the absolute difference of 0.03% for query 22 corresponds to ~1.8 additionally contacted resources for KDMBR^8_{64}.[196] This has already been noted in conjunction with the T1 collection (see section 8.1.2) and shall be underlined once more: For regions which, on a global scale, are sparsely populated with data points, coarser summaries are acceptable since the impact on the resulting *rfc* values is modest. The less resources in the data collection, the more this rule applies.

Surprisingly (bearing in mind the qualitative analysis of the T1 collection at 'MBRsize-' and with regard to query 14, located in San José, *California*, see section 8.1.2), $\text{QTMBR}^4_{64,1.0}$ is now significantly superior for query 37 of the F collection—which is located in Sacramento, *California*. For KDMBR^8_{64}, the spatial resolution of the basic k-d space partition is significantly coarser around California (and for the North American continent, in

[196] For query 22 of the F collection, four resources contribute to the final query result. One of them (resource 5,281) is directly represented for both KDMBR^8_{64} and $\text{QTMBR}^4_{64,1.0}$. Hence, the ideally achievable *rfc* value for both is $(1 + 3 \cdot 2)/5,951 = 0.12\%$. The values which are achieved are 0.20 % for KDMBR^8_{64} and 0.17% for $\text{QTMBR}^4_{64,1.0}$ (see Table 27).

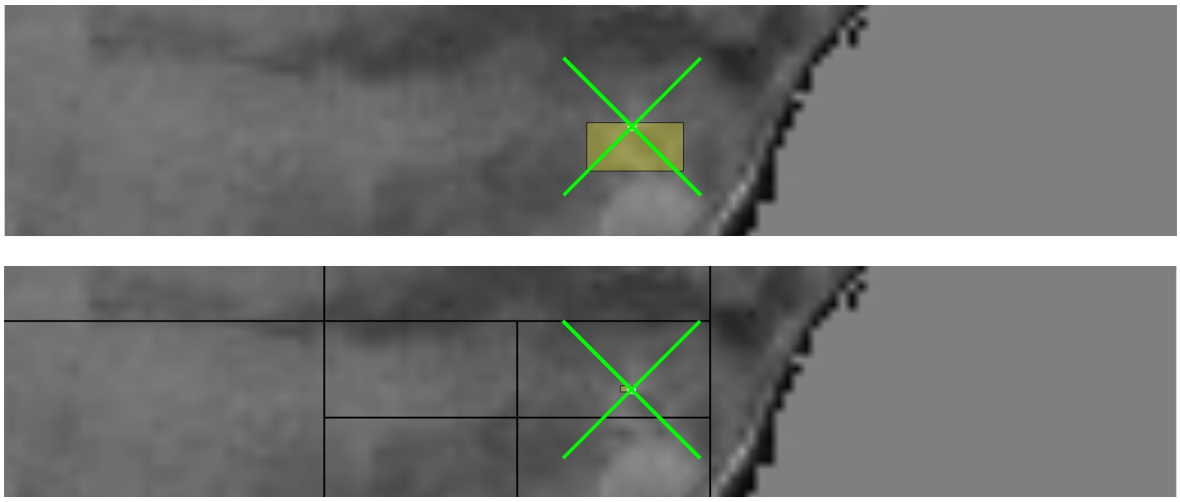


Fig. 62: Depiction of the KDMBR^{8}_{64} summary (top) and the $\text{QTMBR}^{4}_{64,1.0}$ summary (bottom) for resource 5,281 (F collection), visualization tool in its maximum zoom level. The green cross depicts the location of query 22 (Porto Alegre, Brasil).

general) compared to the T1 collection.[197] Nevertheless, when assessing resource 5,661 of the F collection (which is spatially very concentrated and on rank 1 for both the KDMBR^{8}_{64} and $\text{QTMBR}^{4}_{64,1.0}$ rankings), it shows that the indexed surface area for KDMBR^{8}_{64} is smaller compared to its $\text{QTMBR}^{4}_{64,1.0}$ equivalent (see Figure 63).

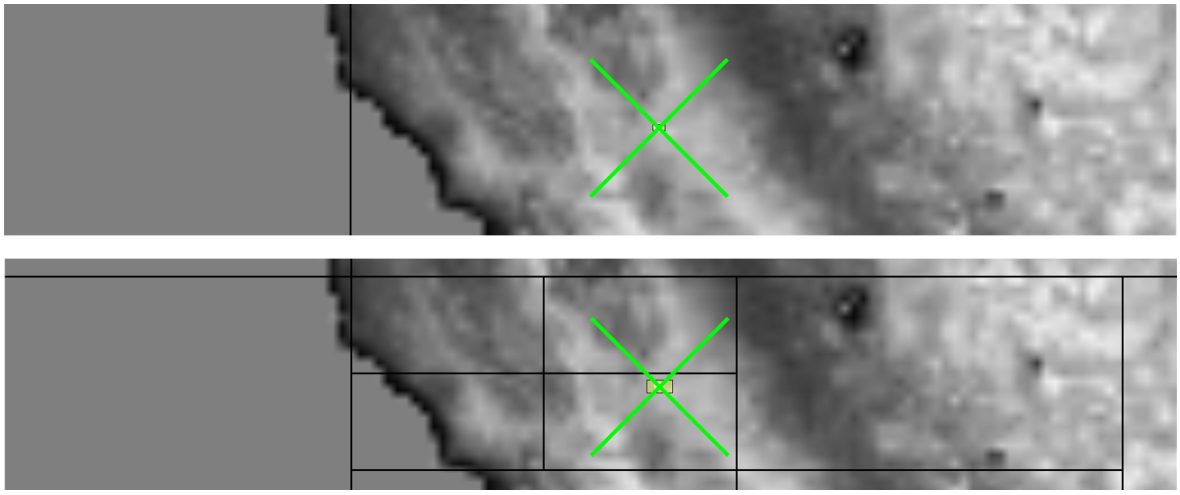


Fig. 63: Depiction of the KDMBR^{8}_{64} summary (top) and the $\text{QTMBR}^{4}_{64,1.0}$ summary (bottom) for resource 5,661 (F collection), visualization tool in its maximum zoom level. The green cross depicts the location of query 38 (Sacramento, California).

Hence, the problem of KDMBR^{8}_{64} cannot be that the *maximum accuracy* of its summaries is not high enough around Sacramento. The problem becomes evident when assessing the further ranking: The basic k-d cell in

[197] See Figure 60 for the basic k-d resolution of the F collection in comparison to Figure 45 (top) on page 192 for the basic k-d resolution of the T1 collection, both when $n = 64$.

which Sacramento is located in is too big and located unfavorably for the given spatial distribution of data points: it roughly extends from Portland in the north to San Diego in the south (see Figure 60 once again). The *basic k-d cell* for KDMBR^8_{64} in the region around Sacramento is much bigger than the *basic quadtree cells* which usually result for $\text{QTMBR}^4_{64,1.0}$. Obviously, many points of interest are located north (e.g. some national forests) and south (e.g. San Francisco and Los Angeles) of Sacramento. Hence, there are quite many resources which simultaneously administer data points both north and south of Sacramento. For these resources, the quantized MBRs of their KDMBR^8_{64} summaries cover large parts of Sacramento's basic k-d cell. As already noted in section 8.1.3 (for the T1 collection at 'MBRsize+'): oftentimes, large parts of an occupied basic cell are covered when quantized MBRs are used as a refinement. Since the resources are spatially more spread for the F collection, this problem is even more apparent here. In the very end, it results in that many KDMBR^8_{64} summaries overlap the query point unnecessarily due to the dead space contained in their quantized MBRs. See Figure 64 for a visualization of the two example resources 1,187 and 1,438 (which both do not administer any data points in the immediate vicinity of query 37): While for $\text{QTMBR}^4_{64,1.0}$, the indexed areas are delineated much tighter and are also far away from the query point, the dead-space-stricken indexed areas of the KDMBR^8_{64} summaries overlap the query point. Consequently, both resources are erroneously high ranked for KDMBR^8_{64} (see Table 28).

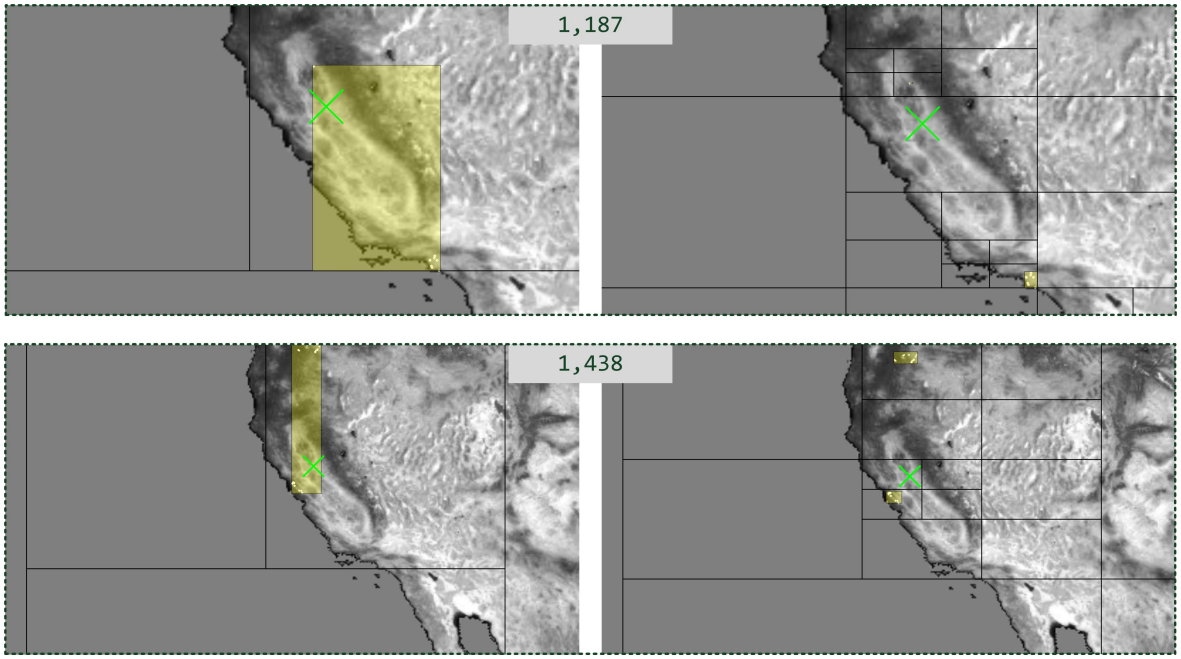


Fig. 64: Depiction of the KDMBR^8_{64} and the $\text{QTMBR}^4_{64,1.0}$ summaries of resource 1,187 (top) and resource 1,438 (bottom) of the F collection. The KDMBR^8_{64} ($\text{QTMBR}^4_{64,1.0}$) summary is always on the left (right) in each block. The green cross depicts the location of query 37.

Table 28: Listing of the ranks of exemplarily selected resources in the KDMBR^{8}_{64} respectively $\text{QTMBR}^{4}_{64,1.0}$ ranking for query 37 (F collection).

resource	KDMBR^{8}_{64}	$\text{QTMBR}^{4}_{64,1.0}$
$ID = 5,661$	1	1
$ID = 1,438$	14	59
$ID = 1,187$	20	215

In total, for the *rfc* value averaged over the 50 queries, the superiority of KDMBR^{8}_{64} for queries in regions with an extremely high global point density comes out favorably compared to the superiority of $\text{QTMBR}^{4}_{64,1.0}$ for queries in regions with middle to low global point densities. This is because the occurring differences with respect to the number of contacted resources are bigger for high density regions. Nevertheless, it has to be noted that even when the overall results are quite close, the suitability of the approaches can vary significantly, depending on the specific locations of the query points.

8.3. Evaluation of the T1 Collection (Disallowing a Direct Representation)

In this section, the approaches are evaluated for the T1 collection, again—but this time, the direct representation of resources is disallowed. Therefore, *all* of the resources are summary-represented. The resulting Skylines are depicted in Figure 65 and Figure 66. In general, the courses of the Skylines are very similar to the Skylines of the 'normal' configuration from section 8.1 (see Figure 42 on page 181). Of course, greater *rds* values result for specific techniques since opting for the storage-space-saving direct representation is not possible anymore. For example, KDMBR^{8}_{512} now requires 54.1 B *rds* instead of 46.2 B *rds*. These *rds* differences between both 'configurations' ('normal' and disallowing the direct representation) increase with 'higher' parameterizations. For 'lower' parameterizations, the *rds* differences are very small since the share of directly represented resources is generally small, too. For $\text{UFS}_{32,cc}$ as well as KDMBR^{3}_{32} and KDMBR^{4}_{32}, the resulting *rds* values are even exactly the same since the share of directly represented resources for these techniques is 0% in the 'normal' configuration.

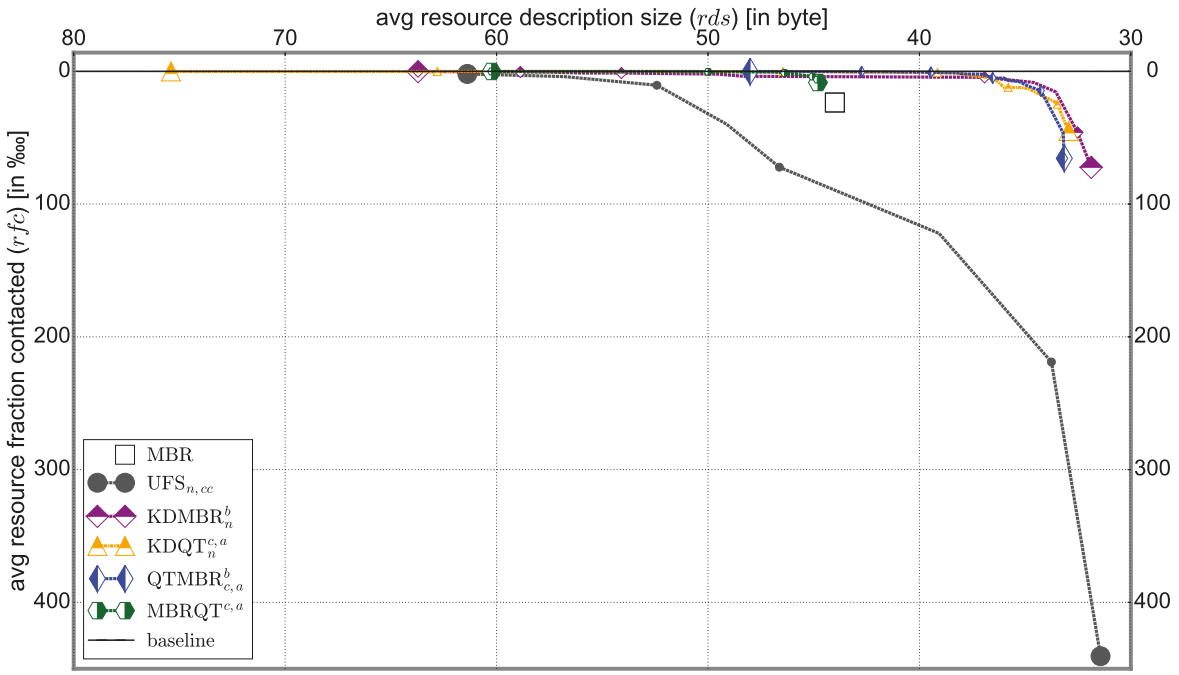


Fig. 65: Full Skylines of the six approaches selected for further evaluation and the T1 collection when disallowing the direct representation of resources.

By visual comparison, it is hard to spot any differences between the *relative* courses of the Skylines for both configurations. Only $\text{UFS}_{n,cc}$ is even significantly further behind. This is easily explained: Since no direct representation is allowed anymore, the spatial footprints of *all* resources are now described by the polygonal area(s) of the globally shared space partition for $\text{UFS}_{n,cc}$—even when a resource administers only one data point.

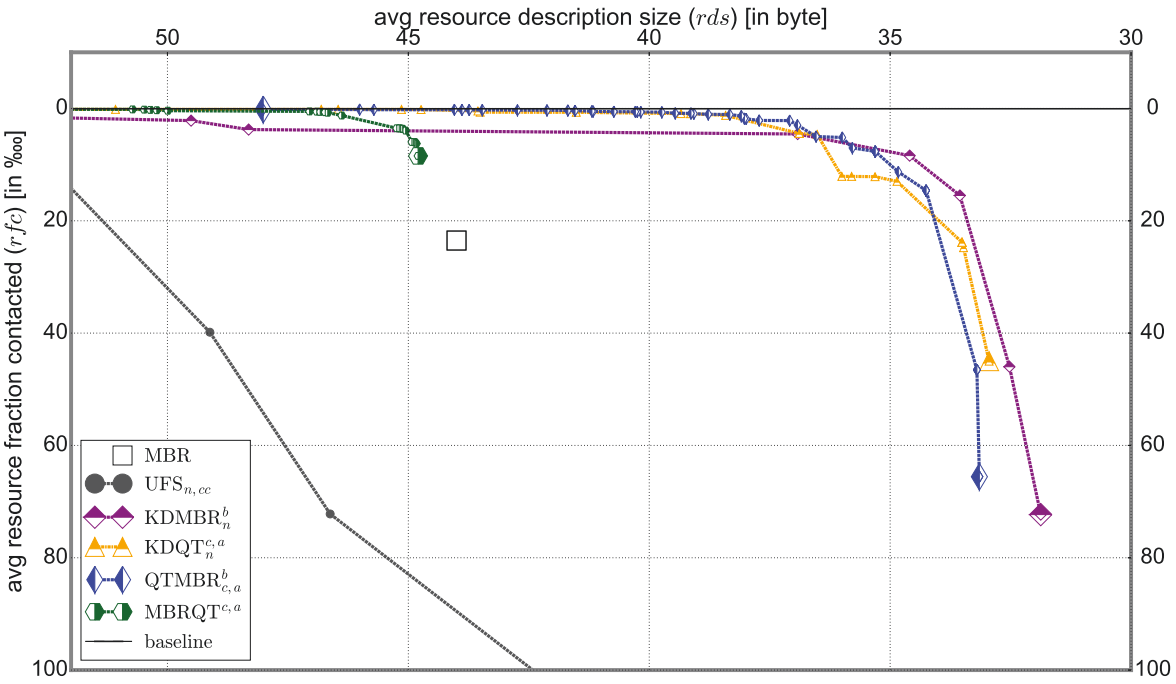


Fig. 66: Zoomed in depiction of the Skylines of the approaches selected for further evaluation and the T1 collection when disallowing the direct representation of resources.

Hence, when performing a query, at least all the resources administering data points in the query cell have to be contacted as the globally shared query cell polygon overlaps the query point (and thus, their distance to the query point is 0). In contrast, in the 'normal' configuration, only the summary-represented resources administering data points in the query cell and the directly represented resources *contributing to the final query result* have to be contacted at the very least. Depending on the parameterization, the share of directly represented resources can be very high for $\text{UFS}_{n,cc}$ and the T1 collection (up to 65.8% for $\text{UFS}_{16384,cc}$). Hence, the resulting *rfc* values for $\text{UFS}_{n,cc}$ (except for $\text{UFS}_{32,cc}$) are massively influenced by the omission of the direct representation. It is also intuitively comprehensible: As only small resources are directly represented, the directly represented resources are usually spatially very narrow. Such resources can be represented very accurately by all the further evaluated approaches except $\text{UFS}_{n,cc}$. Consequently, for the other approaches, the pruning of these resources works very well, and no big differences between the 'normal' configuration and disallowing a direct representation result.
For assessing the impact of the direct representation on the approaches, in Table 29, we list some results for the 'normal' configuration of the T1 collection (*T1 'normal'*) and the configuration disallowing a direct representation of resources (*T1 w/o direct representation*). For each parameterizable

[198]Note that $\text{QTMBR}^{6}_{512,0.5}$ and $\text{KDQT}^{512,1.0E-8}_{8192}$ are not part of the $\text{QTMBR}^{b}_{c,a}$ Skyline respectively the $\text{KDQT}^{c,a}_{n}$ Skyline anymore when disallowing the direct representation for the T1 collection.

approach, we list two parameterizations: The first parameterization is the one at 'MBRsize' for the *T1 'normal'* configuration. The second parameterization is the 'highest' parameterization for each approach, i.e. they are the respective spatially most accurate resource descriptions.[198]

Table 29: Comparison of the resulting *rds* and *rfc* values for selected techniques with respect to the T1 collection in its 'normal' configuration (light-green column) and when disallowing a direct representation (mid-green column).

setting → / technique ↓	T1 'normal'		T1 w/o direct representation	
	rds [in byte]	*rfc* [in ‰]	*rds* [in byte]	*rfc* [in ‰]
MBR	41.06	21.13	44.00	23.50
$\text{UFS}_{256,cc}$	41.24	38.62	46.62	72.20
$\text{UFS}_{16384,cc}$	48.61	1.00	61.38	2.00
KDMBR^{6}_{256}	42.96	2.97	48.31	3.76
KDMBR^{8}_{8192}	51.23	0.19	63.71	0.24
$\text{KDQT}^{64,1.0E-5}_{32}$	41.21	0.67	41.51	0.68
$\text{KDQT}^{512,1.0E-8}_{8192}$	61.71	0.09	78.07	0.12
$\text{QTMBR}^{6}_{512,0.5}$	41.04	0.39	41.80	0.41
$\text{QTMBR}^{8}_{1024,0.001}$	46.11	0.17	48.01	0.18
$\text{MBRQT}^{16,1.0}$	41.53	7.62	44.80	8.45
$\text{MBRQT}^{8192,1.0E-8}$	55.43	0.11	60.23	0.13

As just discussed, the impact of omitting a direct representation is clearly the highest for $\text{UFS}_{n,cc}$, roughly doubling the *rfc* values. For the other approaches, the changes are not that significant—as has been suspected at various passages of the detailed evaluation, before. For their 'highest' parameterizations, the differences are to large portions caused by the double-counting of summary-represented resources contributing to the query result. For e.g. $\text{KDQT}^{512,1.0E-5}_{8192}$, 6.13 *directly represented* resources contribute to the query result on average in the 'normal' configuration. The exact *rfc* value for $\text{KDQT}^{512,1.0E-5}_{8192}$ is 0.1342 ‰ in the 'normal' configuration (corresponding to ~33.4 contacted resources) and 0.1700‰ when disallowing a direct representation (corresponding to ~42.4 contacted resources). Hence, due to the double-counting, in fact only about three[199] additional resources are contacted for $\text{KDQT}^{512,1.0E-5}_{8192}$ when disallowing the direct representation.

At this point, we reiterate query 22 for $\text{KDQT}^{256,1.0E-5}_{128}$ and $\text{QTMBR}^{8}_{1024,0.001}$ (see section 8.1.3). Table 30 lists the results for query 22 as well as the aggregated overall results for both configurations. It can be seen that $\text{KDQT}^{256,1.0E-5}_{128}$ is still notably superior to $\text{QTMBR}^{8}_{1024,0.001}$ for query 22: In the 'normal' configuration, it is 0.81 ‰ for $\text{KDQT}^{256,1.0E-5}_{128}$ versus 1.44

[199] The given value arises as $(42.4 - 33.4) - 6.13 = 2.87$.

Table 30: Comparison of the resulting *rds* and *rfc* values for $\text{KDQT}_{128}^{256,1.0E-5}$ and $\text{QTMBR}_{1024,0.001}^{8}$ with respect to the T1 collection in its 'normal' configuration (light-green column) and when disallowing a direct representation (mid-green column). Both the results for query 22 as well as the overall results are depicted.

	T1 'normal'				T1 w/o direct representation			
	$\text{KDQT}_{128}^{256,1.0E-5}$		$\text{QTMBR}_{1024,0.001}^{8}$		$\text{KDQT}_{128}^{256,1.0E-5}$		$\text{QTMBR}_{1024,0.001}^{8}$	
	rds in [in byte]	*rfc* [in ‰]	*rds* [in byte]	*rfc* [in ‰]	*rds* [in byte]	*rfc* [in ‰]	*rds* [in byte]	*rfc* [in ‰]
query 22	47.16	0.81	46.11	1.44	50.71	0.89	48.01	1.45
overall	47.16	0.14	46.11	0.17	50.71	0.17	48.01	0.17

‰ for $\text{QTMBR}_{1024,0.001}^{8}$. Now, it is 0.89 ‰ versus 1.45 ‰. For the overall results, there are also no significant changes: $\text{KDQT}_{128}^{256,1.0E-5}$ is still minimally better than $\text{QTMBR}_{1024,0.001}^{8}$ for the 'T1 w/o direct representation' configuration (when assessing the non-rounded *rfc* values). Unquestionably, the $\text{KDQT}_{128}^{256,1.0E-5}$ *rfc* values worsen slightly more for query 22 and also in general. This was to be expected since in the 'normal' configuration, 51.22% of the resources are directly represented for $\text{KDQT}_{128}^{256,1.0E-5}$ (compared to 45.81% for $\text{QTMBR}_{1024,0.001}^{8}$), i.e. more resources lose their 'perfect' spatial representation for $\text{KDQT}_{128}^{256,1.0E-5}$. On the other hand, the overall difference for $\text{KDQT}_{128}^{256,1.0E-5}$ between both configuratons corresponds to 0.03 ‰ or ~7.5 additionally contacted resources on average. All in all, this shows that the option of directly representing resources has an impact on the results, but this impact is fairly low. As discussed, the sole exception is $\text{UFS}_{n,cc}$ (or pure global space partitioning approaches in general) which exhibits significantly worse results. Since global space partitioning approaches are generally unsuited for the T1 collection anyway, in total, disallowing the direct representation does not result in significant changes when comparing the suitability of the different approaches against each other.

For the sake of completeness, Table 31 shows the key figures for the respective approaches at 'MBRsize' (see Figure 67 for a zoomed in depiction of the Skylines around 'MBRsize'). Obviously, the suitability of the techniques differs so clearly that the area-related key figures show perfect congruence with the resulting *rfc* values. The previously observed effects of especially KDMBR_{n}^{b} outperforming $\text{QTMBR}_{c,a}^{b}$ despite tremendously worse area-related key figures as well as $\text{KDQT}_{n}^{c,a}$ obtaining a lower 'overlap between summaries' compared to $\text{QTMBR}_{c,a}^{b}$ (which both are caused by the varying resolution of the basic global k-d space partition) do not occur here. Consequently, $\text{QTMBR}_{1024,0.001}^{8}$ and (a little behind) $\text{KDQT}_{64}^{64,1.0E-5}$ prove to be the most suitable *summarization* approaches for the T1 collection at 'MBRsize'.

Note that in comparison with Table 12 on page 183 (which depicts the key figures at 'MBRsize' for the 'normal' configuration), the 'overlap between

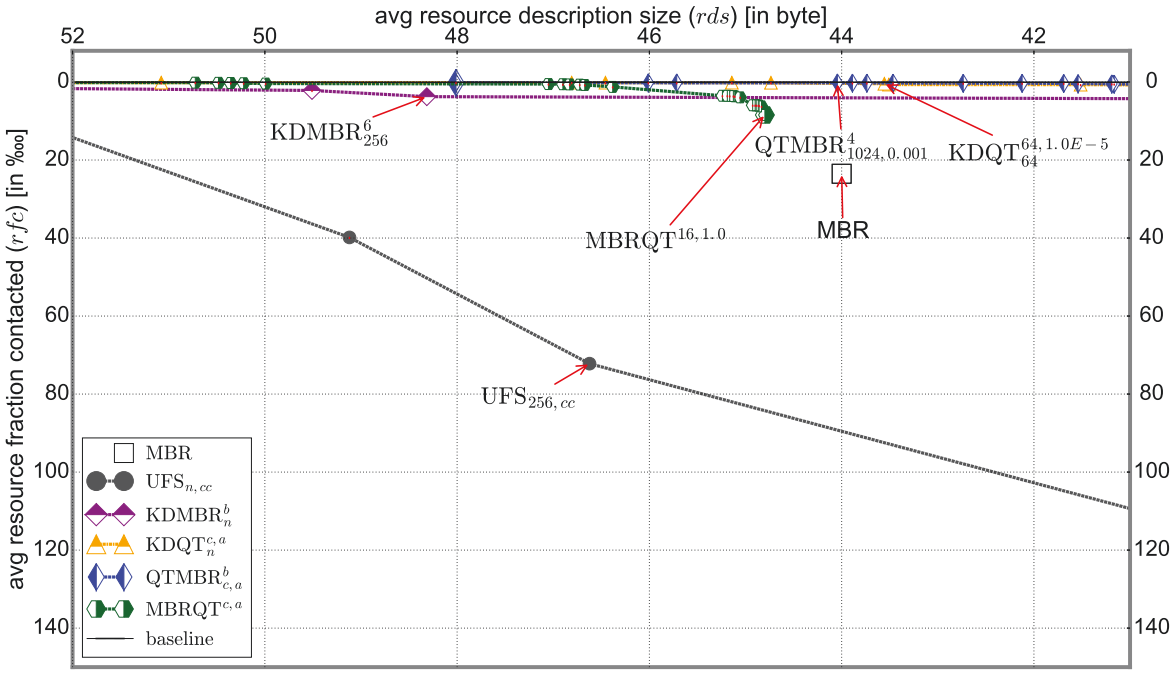


Fig. 67: Zoomed in version of Figure 65, showing the Skylines for the T1 collection around 'MBRsize' when disallowing a direct representation. The respective Skyline parameterizations of the different approaches at 'MBRsize' are marked with red arrows.

summaries' and the 'data space coverage' are greater whereas the 'surface area per summary' is smaller for each approach—in case the same parameters are used in both tables.[200] This is no surprise since the former two are key figures which are *accumulated* over the whole data collection—which now comprises of much more summary-represented resources and thus also a much higher 'number of indexed areas'. The 'surface area per summary', on the other hand, is a key figure which is *averaged* over the amount of summary-represented resources. Since a lot of small resources (which are usually spatially narrow) are now summary-represented, the *average* surface area per summary is significantly smaller when disallowing the direct representation. In total, the increase in both the 'overlap between summaries' as well as the 'data space coverage' is moderate (due to the small surfaces of the 'newly' summary-represented resources)—similar to the increase of the *rfc* values. Once again, this shows that the impact of the option to directly represent resources is not very significant.
These observations apply to the MBR approach, KDMBR$^{6}_{256}$, and MBRQT$^{16,1.0}$. For KDQT$^{c,a}_{n}$ and QTMBR$^{b}_{c,a}$, the parameterizations changed between both tables. For UFS$_{256,cc}$, the key figures (even the 'surface area per summary') as well as the resulting *rfc* value are significantly greater.

[200] Note that for KDQT$^{c,a}_{n}$ and QTMBR$^{b}_{c,a}$, there are different parameterizations in both tables: KDQT$^{32,1.0E-5}_{64}$ and QTMBR$^{4}_{1024,0.001}$ in Table 12 as opposed to KDQT$^{64,1.0E-5}_{64}$ and QTMBR$^{4}_{1024,0.001}$ in Table 31. Therefore, the key figures listed for these are not directly comparable.

Table 31: Listing of various key figures alongside the resulting *rds* and *rfc* values for the T1 collection (disallowing a direct representation) and the benchmarked techniques at 'MBRsize'.

technique → / key figure ↓	MBR	$UFS_{256,cc}$	$KDMBR^{6}_{256}$	$KDQT^{64,1.0E-5}_{64}$	$QTMBR^{4}_{1024,0.001}$	$MBRQT^{16,1.0}$
overlap between summaries [in dsu^2]	3.05E+10	1.8E+13	3.62E+08	392,878	4,187	7.02E+08
data space coverage	464.3	17,142.6	12.76	2.42	0.08	60.76
surface area per summary [in dsu^2]	12.07	445.8	0.33	0.06	0.002	1.58
number of indexed areas	2,491,785	2,916,578	2,895,618	6,012,697	5,010,367	2,684,237
aspect ratio (mean) [:1]	11.05	-	1.82	1.63	2.12	5.09
aspect ratio (median) [:1]	1.26	-	2.00	2.00	2.00	1.25
avg *rfc* [in ‰]	23.50	72.20	3.76	0.45	0.22	8.45
avg *rds* [in byte]	44.00	46.62	48.31	43.56	44.04	44.80

Looking at the massive increase of its 'data space coverage' (17,142.6 versus 6,994.6), obviously, a lot of the 'newly' summary-represented resources administer their data points in regions with a low global point density. Therefore, their data points are predominantly located in rather large Voronoi cells of the global space partition, resulting in the massive increase of the area-related key figures for $UFS_{256,cc}$.

8.4. Evaluation of the F Collection (Disallowing a Direct Representation)

In this section, we briefly evaluate the impact of disallowing a direct representation of resources for the F collection. Figure 68 and Figure 69 show the resulting Skylines. Similar to the T1 collection, the *relative* courses of the Skylines are almost identical for both configurations of the F collection (also see Figure 57 on page 214 and Figure 58 on page 215 for the 'normal' configuration).

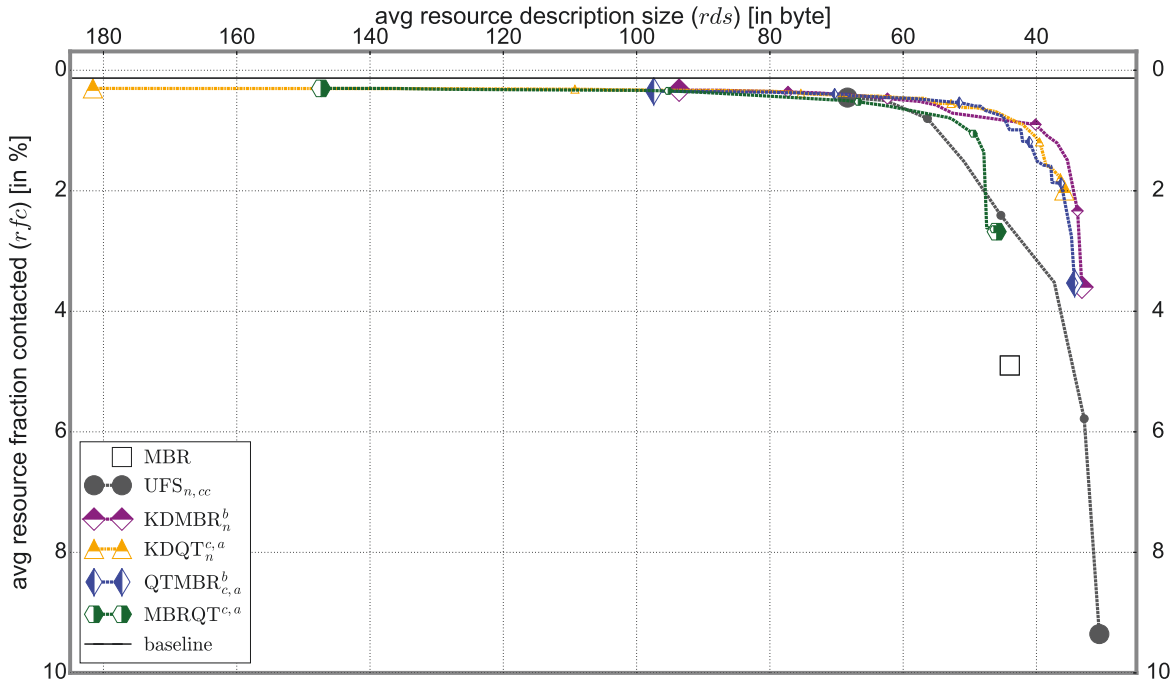


Fig. 68: Full Skylines of the six approaches selected for further evaluation and the F collection when disallowing the direct representation of resources.

The general observations made for the T1 collection also apply for the most part when assessing the key figures of the different approaches at 'MBR-size' (see Table 32): The 'overlap between summaries' and the 'data space coverage' increase moderately when disallowing a direct representation while the 'surface area per summary' decreases comparatively strongly.[201] Since the share of directly represented resources is significantly lower for the F collection, the changes—viewed relatively—are smaller compared to the changes for the T1 collection.

[201] Note that for KDQT$^{c,a}_{n}$ and QTMBR$^{b}_{c,a}$, the parameterizations at 'MBRsize' are different compared to the 'normal' configuration. Therefore, the listed values of both are not directly comparable to the values listed in Table 26 on page 217. Nevertheless, the remarks also apply for KDQT$^{32,1.0E-5}_{32}$ and QTMBR$^{4}_{64,1.0}$ (i.e. the parameterizations listed in Table 26). For example, the 'overlap between summaries' is now 151,842 dsu^2 for KDQT$^{32,1.0E-5}_{32}$ (as opposed to 151,838 dsu^2 in the 'normal' configuration, see Table 26).

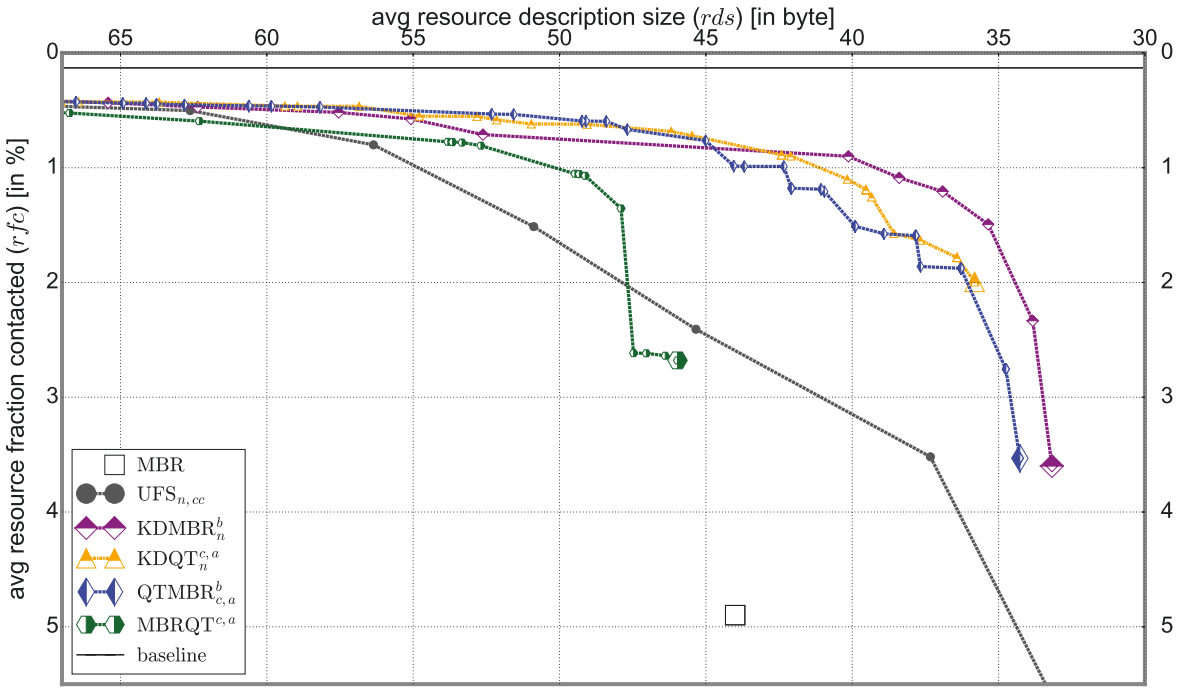


Fig. 69: Zoomed in depiction of the Skylines of the approaches selected for further evaluation and the F collection when disallowing the direct representation of resources.

For UFS$_{256,cc}$, the 'overlap between summaries' and the 'data space coverage' again increase significantly more than for the other techniques. Once more, this is because for UFS$_{n,cc}$, only the polygons of the global space partition are available for indexing. In contrast to the findings for the T1 collection, the 'surface area per summary' for UFS$_{256,cc}$ decreases when disallowing the direct representation, though. This is attributable to the greater spatial spread in the F collection which leads to that big resources often administer data points in several Voronoi cells. For small resources administering only one data point—which were previously most often directly represented ($\rightarrow$ depending on the bit vector clipping)—solely one cell can be occupied, obviously. Thus, the average surface area per summary decreases due to the newly summary-represented small resources.
Another interesting observation is that the MBR approach and UFS$_{256,cc}$ switch their ranks with respect to the 'data space coverage' and the 'surface area per summary' when comparing the 'normal' configuration and the configuration disallowing a direct representation. For the latter, UFS$_{256,cc}$ features the worst numbers for both key figures now (in the 'normal' configuration, the MBR approach has the worst numbers). This is easily explained: MBRs can be arbitrarily small (a point in the extreme case) whereas for UFS$_{n,cc}$, the polygonal areas of the Voronoi cells are the smallest indexable spatial units. Thus, the MBRs indexing the newly summary-represented (small and spatially narrow) resources are much more accurate. With regard to the 'overlap between summaries', UFS$_{256,cc}$ still features less overlap and also the resulting *rfc* value is better than for the MBR approach.

Table 32: Listing of various key figures alongside the resulting *rds* and *rfc* values for the F collection (disallowing a direct representation) and the benchmarked techniques at 'MBRsize'.

technique → / key figure ↓	MBR	$UFS_{256,cc}$	$KDMBR^{8}_{64}$	$KDQT^{32,0.001}_{64}$	$QTMBR^{3}_{512,1.0}$	$MBRQT^{16,1.0}$
overlap between summaries [in dsu^2]	284,858,335	170,488,306	613,797	80,547	8,780	18,399,643
data space coverage	57.33	59.06	1.83	0.42	0.02	10.67
surface area per summary [in dsu^2]	624.3	643.1	19.87	5.30	0.25	116.2
number of indexed areas	5,951	15,722	10,391	26,169	19,224	11,972.5
aspect ratio (mean) [:1]	6.33	-	2.17	1.32	2.10	5.90
aspect ratio (median) [:1]	1.58	-	1.62	1.00	2.00	1.88
avg *rfc* [in %]	4.90	2.41	0.90	0.72	0.98	2.68
avg *rds* [in byte]	44.00	45.34	40.13	45.48	44.05	45.97

Hence, it shows once again that the *rfc* value correlates most with the 'overlap between summaries' and not with the 'data space coverage'—which has already been observed in section 8.1.3.
Due to the general conformance of the results to already evaluated configurations, we close the assessment of the results for the F collection when disallowing a direct representation at this point.

8.5. Evaluation of the F Collection With an Altered Selection of the Query Points

In section 6.3, we describe how the query points are selected for the different data collections. For the F collection, it is a two-step process as standard: First, a random resource is selected. Second, a random data point from this resource is selected as query point. In the following, we refer to this standard selection process as *queryMode2* (similar to the denomination in [Henrich and Blank 2010] where it was applied first). Since there are much more small resources than big resources in the F collection, this implicitly prefers selecting the query points from small resources. For the evaluation of the F collection in section 8.2, *queryMode2* is applied.
In this section, we investigate if there are any changes when the selection process is more straightforward: the query points are simply randomly selected from the underlying data collection. Since most data points are administered by big resources, this implicitly prefers selecting query points from big resources. We refer to this selection process as *queryMode1* in the following (in compliance with the denomination in [Henrich and Blank 2010], again). Basically, this constitutes a slight change in the underlying application scenario: Now, it is assumed that users providing a lot of media items in the P2P network are also more active. Consequently, they issue more queries than users providing less media items (for *queryMode2*, it has been assumed that all users are equally active). As before, the users search for media items near the locations where certain of their own media items have been created. The locations of the query points of *queryMode1* are depicted in Figure 70. In comparison with *queryMode2* (see bottom image of Figure 31 on page 116), there are much more query points which are located in Europe and significantly less in North America.
For *queryMode1*, 2.90 resources contribute to the query result on average. Consequently, the ideal *rfc* value is $2.90/5,951 = 4.87$ ‰. In Figure 71 and Figure 72, the resulting Skylines are depicted. Overall, the resulting *rfc* values for *queryMode1* are significantly lower compared to those of *queryMode2* which is easily comprehensible: For the latter, more resources contribute to the query result on average (i.e. the selection task is more difficult) plus, in general, big resources are preferred in the ranking algorithm. In Table 33, we list the results for some selected techniques for both query modes.[202] Naturally, the *rds* values for a specific technique are the same (as the query modes only differ in their set of query points). By the values in the '*queryMode1* / *queryMode2*'-column, for each technique, the relative improvement of the *rfc* value which results from the switch from *queryMode2* to *queryMode1* can be assessed. For example, for the MBR ap-

[202] Note that for each approach, these are the parameterizations resulting at 'MBRsize-' (~36.5 B *rds*), at 'MBRsize' (~42.4 B *rds*), and at 'MBRsize+' (~60.0 B *rds*) with respect to the F collection in its 'normal' configuration. Since the resulting *rds* values are much greater for the F collection in comparison to the T1 collection, 'MBRsize+' of the F collection is also selected to be a much greater value than 'MBRsize+' of the T1 collection.

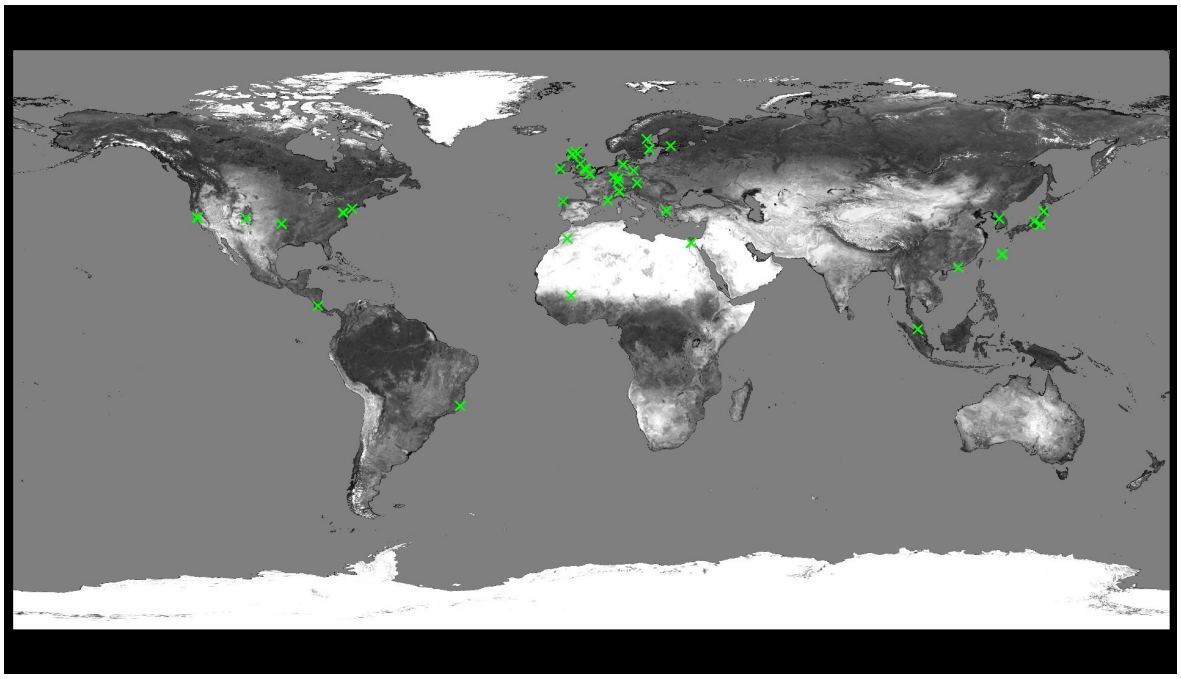


Fig. 70: Locations of the query points for *queryMode1* of the F collection.

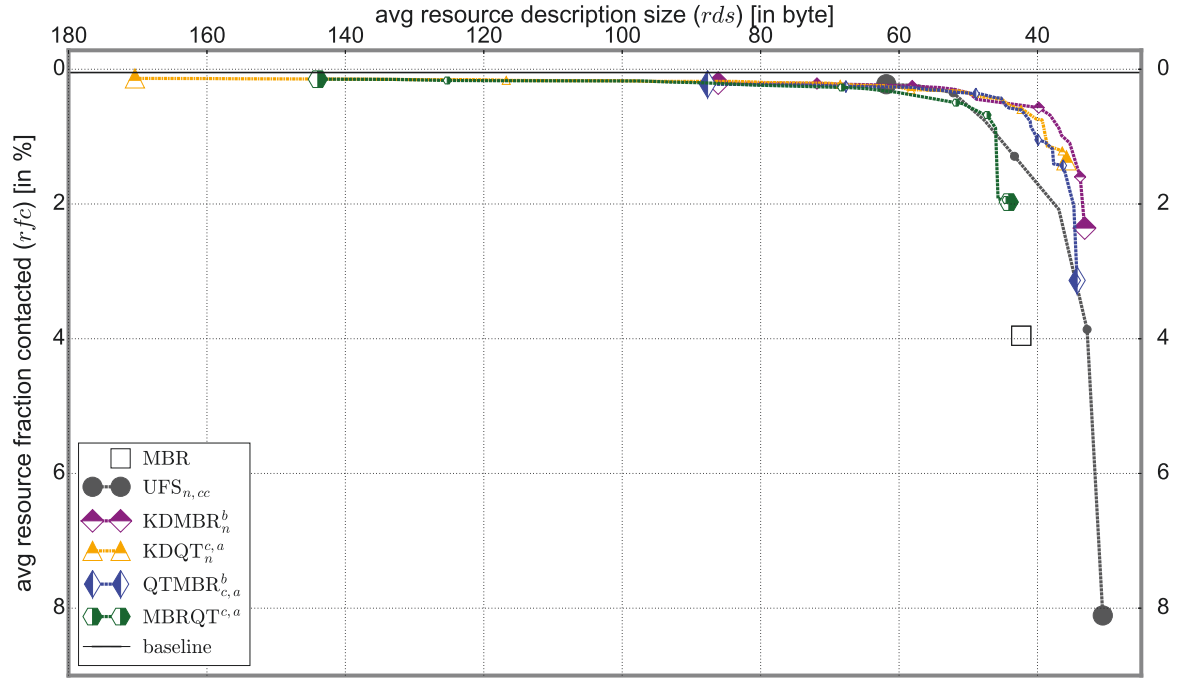


Fig. 71: Full Skylines of the six approaches selected for further evaluation and the F collection, selecting the query points by *queryMode1*.

proach, the *rfc* value of *queryMode1* divided by the *rfc* value of *queryMode2* results in $3.96\%/4.83\% = 0.82$. Hence, 18% less resources are contacted for the MBR approach in *queryMode1*. In general, the relative improvements are greater the more accurate the resource descriptions are (i.e. they are greatest at 'MBRsize+').

For all three target-*rds*-values, the relative improvements are the greatest for UFS$_{n,cc}$. Indeed, in a visual assessment of the Skylines for *queryMode1*

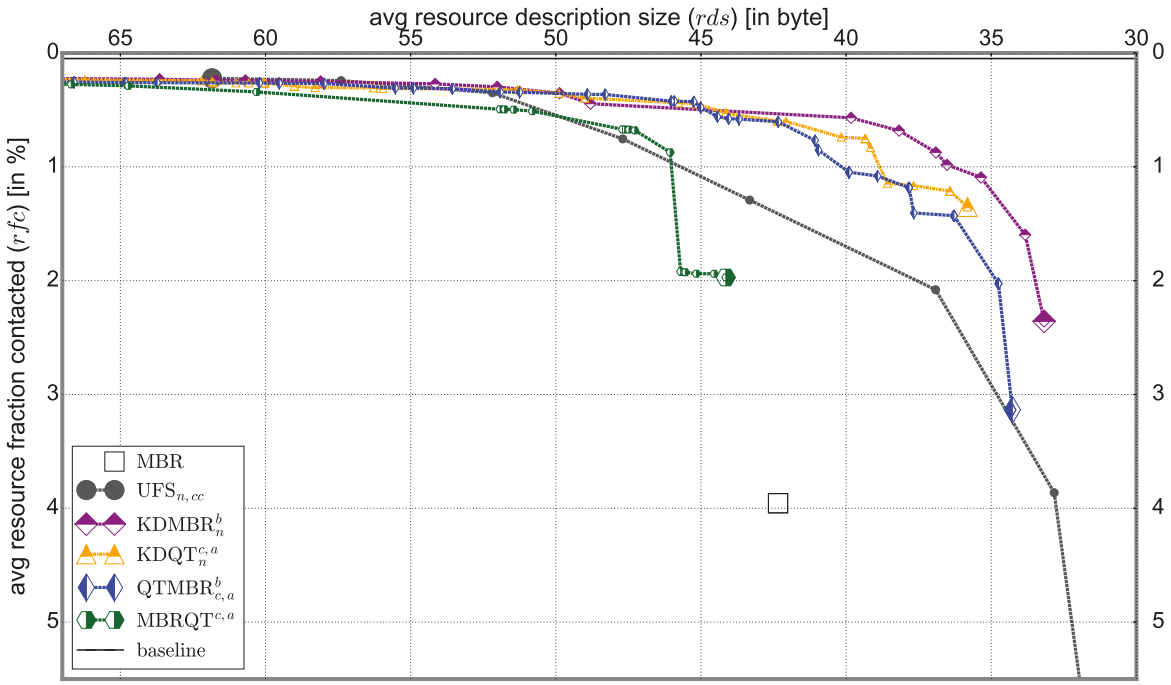


Fig. 72: Zoomed in depiction of the Skylines of the approaches selected for further evaluation and the F collection, selecting the query points by *queryMode1*.

Table 33: Comparison of the results for selected techniques when choosing the query points for the F collection by *queryMode1* respectively *queryMode2*.

	rds [in byte]	*queryMode1* *rfc* [in %]	*queryMode2* *rfc* [in %]	*queryMode1* / *queryMode2*	
UFS$_{128,cc}$	36.92	2.08	3.32	0.63	
KDMBR$^{4}_{64}$	36.52	0.98	1.44	0.68	MBRsize-
KDQT$^{16,0.1}_{32}$	36.42	1.21	1.78	0.68	
QTMBR$^{6}_{16,1.0}$	36.29	1.43	1.88	0.76	
MBR	42.35	3.96	4.83	0.82	
UFS$_{256,cc}$	43.32	1.29	2.18	0.59	
KDMBR$^{8}_{64}$	39.83	0.57	0.89	0.64	MBRsize
KDQT$^{32,1.0E-5}_{32}$	42.36	0.60	0.88	0.68	
QTMBR$^{4}_{64,1.0}$	42.34	0.60	0.99	0.61	
MBRQT$^{16,1.0}$	44.14	1.94	2.65	0.73	
UFS$_{16384,cc}$	61.84	0.22	0.41	0.54	
KDMBR$^{8}_{512}$	60.70	0.24	0.42	0.57	
KDQT$^{64,1.0E-8}_{64}$	58.30	0.30	0.45	0.67	MBRsize+
QTMBR$^{6}_{1024,0.1}$	60.21	0.26	0.45	0.58	
MBRQT$^{256,0.001}$	60.30	0.34	0.58	0.59	

(see Figure 71 and Figure 72) and *queryMode2* (see Figure 57 on page 214 and Figure 58 on page 215), the UFS$_{n,cc}$ Skyline seems to be a bit more com-

petitive for *queryMode1*. For example, it touches the $\text{QTMBR}^{b}_{c,a}$ Skyline at ~43.3 B *rds* or catches up again with the $\text{MBRQT}^{c,a}$ Skyline significantly earlier compared to *queryMode2*. Remember that the ranking algorithm works as follows: In case the distances to the query point and the surfaces of the indexed areas are *exactly* the same for two resources, the ranking algorithm assigns the better rank to the bigger resource. This leads to a *strong* preference for big resources in the $\text{UFS}_{n,cc}$ ranking: Due to the globally shared space partition as sole indexing tool, it often occurs that two or more resources are described by *exactly* the same areas for $\text{UFS}_{n,cc}$. In such cases, the biggest of these resources is ranked highest. For the other approaches, the preference for the big resources is not as pronounced since the resource descriptions are much more 'unique' (i.e. it is rare that *exactly* the same areas are indexed for two or more resources). As the query points for *queryMode1* mostly originate from big resources, the stronger preference for big resources in the $\text{UFS}_{n,cc}$ ranking is reflected in the resulting *rfc* values and is therefore also visible in the Skyline diagram.
Other than that, the relative courses of the Skylines and thus the general performances of the approaches are very similar to *queryMode2*. For example, KDMBR^{b}_{n} is still the best approach for low *rds* values and also falls off compared to $\text{KDQT}^{c,a}_{n}$ and $\text{QTMBR}^{b}_{c,a}$ in the range between 40.0 B *rds* and 50.0 B *rds*. Hence, we close the assessment of *queryMode1* for the F collection at this point.

8.6. Evaluation of the T2 Collection

In this section, we evaluate the T2 collection for the selected approaches. As mentioned in section 6.2, a different application scenario is assumed for the T2 collection: It is supposed that a media item provider grants users access to its media items. Internally, the media archive of the provider is organized as a distributed database management system. The distribution of media items to resources is much more balanced for a more even load distribution, and the assignment of media items to resources is not decided by the spatial properties as there are also other searchable properties (see section 1.2.1). In principle, such a scenario represents the worst case for a search based on our spatial resource descriptions: All resources administer a lot of data points and the data points of each resource are more or less scattered across the entire data space. As a result, the search task is significantly more difficult for the T2 collection since the spatial distributions of the different resources' data point sets are much more similar to each other and consequently, the 'data point clouds' of the resources are much more intermixed. Therefore, the demands on the spatial accuracy of the resource descriptions are very high in order to be able to make a clear distinction between relevant and irrelevant resources. It is therefore a test for our approaches under the most adverse conditions.
In all other evaluations, we used the values listed in Figure 32 (page 118) to parameterize the examined approaches. Here, we deviate from this procedure since otherwise, *rfc* values of ~80% or even more would be the best results achieved by some approaches. Hence, for some of them, additional parameter values are tested which result in more accurate resource descriptions. The tested parameters for the T2 collection are listed in Figure 73. The newly added values are highlighted in red. As always, all possible parameter combinations are tested for each approach.

MBR		no parameters
$UFS_{n,cc}$	n	32|64|128|256|512|2048|8192|16384|24576|32768
	cc	16|64|256|n
$KDMBR^{b}_{n}$	n	32|64|256|512|2048|8192|16384|24576|32768
	b	3|4|6|8
$KDQT^{c,a}_{n}$	n	32|64|128|256|512|2048|8192
	c	16|32|64|256|512
	a	1.0|0.1|0.001|1.0E-5|1.0E-7|1.0E-8
$QTMBR^{b}_{c,a}$	c	16|32|64|256|512|1024|2048|4096|8192|16384
	a	1.0|0.1|0.05|0.01|0.001|1.0E-5|1.0E-7|1.0E-8
	b	3|4|6|8
$MBRQT^{c,a}$	c	16|64|256|512|1024|2048|8192|16384|24576|32768
	a	1.0|0.1|0.001|1.0E-5|1.0E-7|1.0E-8

a = threshold surface **a**rea
b = # of **b**its
c = max. # of **c**ells of a quadtree
cc = # of **c**entroids **c**onsidered
n = **n**umber of subspaces

Fig. 73: Listing of the tested parameters for the different approaches and the T2 collection. The additionally tested parameter values are colored red.

The resulting Skylines are depicted in Figure 74 and Figure 75. In comparison to the F and T1 collections, both the resulting *rds* values as well as the

resulting *rfc* values are massively increased. Obviously, the high *rds* values are the result of the great number of data points per resource and their extensive spatial distribution. With 44.56 resources contributing to the query result on average, there are a lot more relevant resources compared to the other evaluations despite the small cardinality of the resource set (3,019). Thus, the ideal *rfc* value is $44.56/3,019 = 1.48\%$. Furthermore, the 'data point clouds' of the resources are much more intermixed with each other. In total, the selection task is much harder compared to the other evaluations, leading to the high *rfc* values. Note that for all tested techniques, the entirety of the resources is summary-represented—not in a single case, a direct representation is chosen.

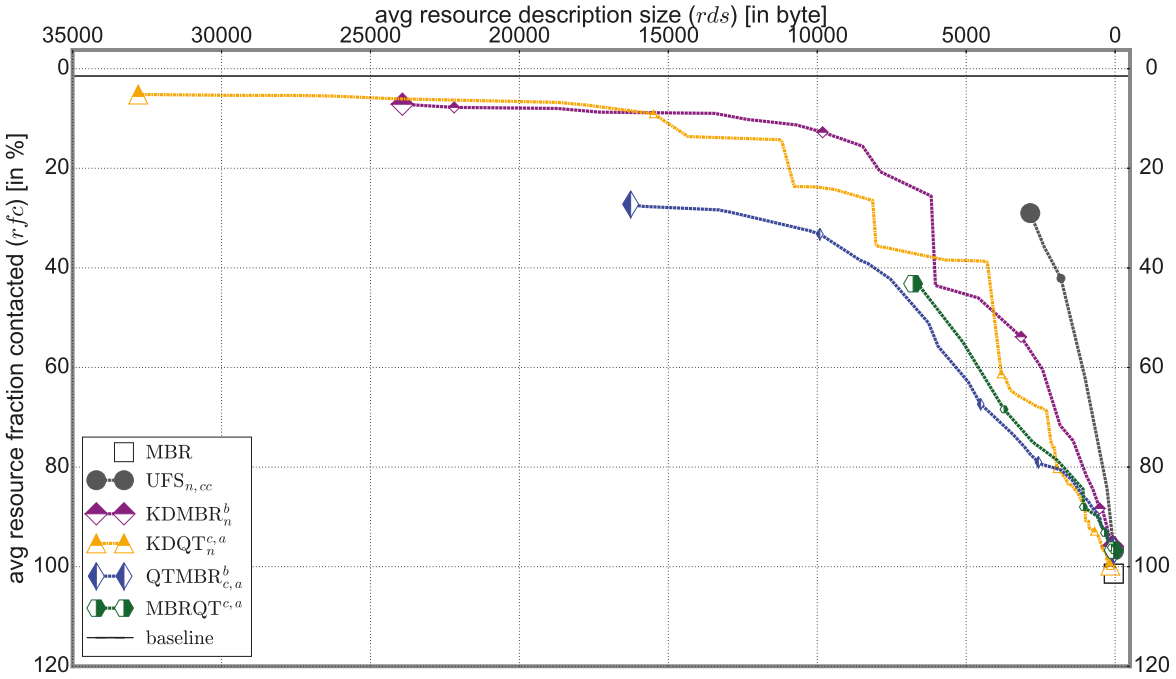


Fig. 74: Full Skylines of the six approaches selected for further evaluation and the T2 collection.

When assessing the Skylines, it shows that for the MBR approach, an *rfc* value greater than 100% results. This comes from doubly contacting summary-represented resources contributing to the query result and the poor spatial accuracy provided by the MBR summaries for the given data point sets. Also see Figure 76 for an example visualization. Averaged over the 50 queries, only about two resources can be pruned when processing a query—corresponding to contacting ~99.93% of the resources. Since the 44.56 result-contributing resources are contacted twice, this adds up to the final *rfc* value of ~101.40% for the MBR approach. For the other approaches, there are no parameterizations resulting in *rfc* values greater than 100% (KDQT$^{16,1.0}_{32}$ is the worst technique with an *rfc* value of 99.79%). Nevertheless, all of them require significant amounts of storage space to achieve even halfway decent *rfc* values.

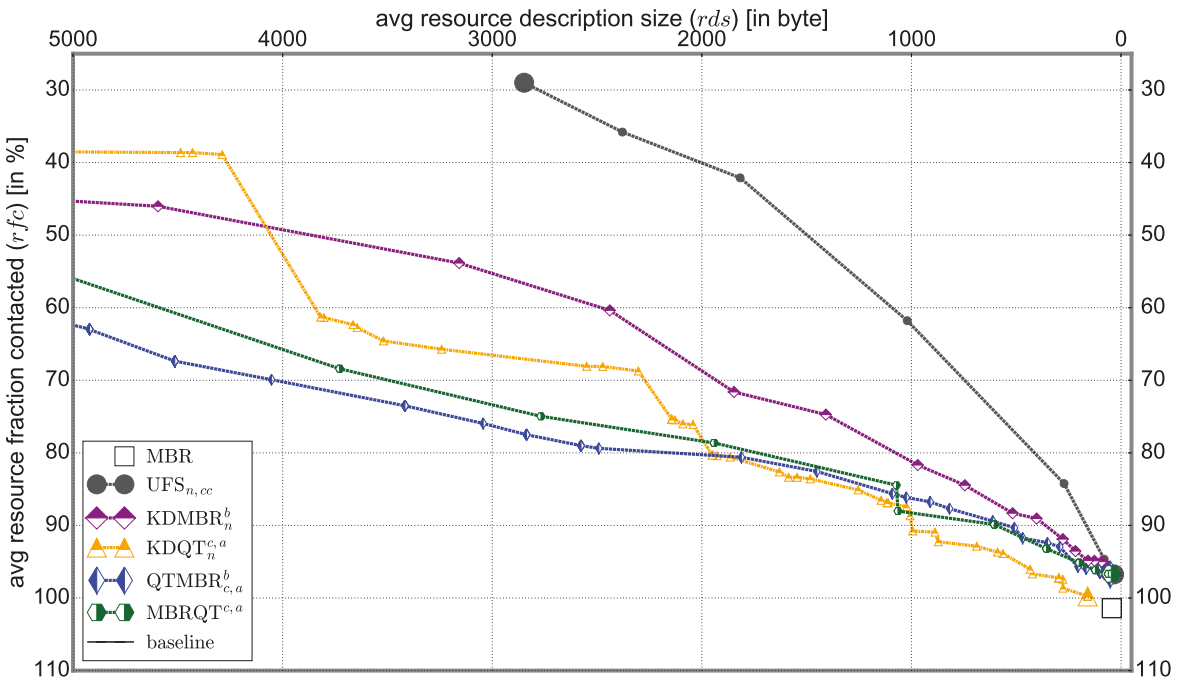


Fig. 75: Zoomed in depiction of the Skylines of the approaches selected for further evaluation and the T2 collection.

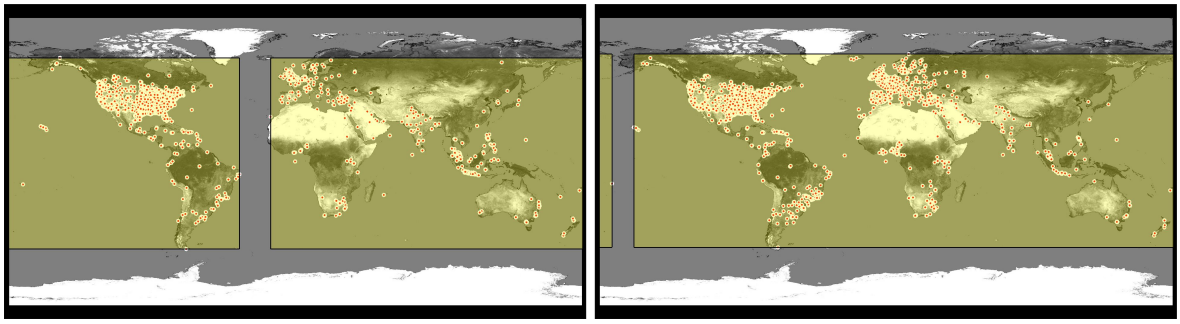


Fig. 76: Example visualizations of the MBR summaries for resource 0 (left, administering 7,437 data points) and resource 302 (right, administering 14,009 data points) of the T2 collection.

In contrast to all other evaluations, the pure global space partitioning UFS$_{n,cc}$ shows to be the most suitable approach in this scenario *by far*. None of the hybrid approaches can keep up. The approaches *based* on global space partitioning (KDMBR$^{b}_{n}$ and KDQT$^{c,a}_{n}$) are the best of the hybrid approaches, with KDMBR$^{b}_{n}$ being fairly superior for most parts of the *rds* range. Unfamiliarly, QTMBR$^{b}_{c,a}$ is the worst approach, surpassing MBRQT^{c,a} and KDQT$^{c,a}_{n}$ only temporarily for 'low' *rds* values until ~1,000 B respectively ~2,000 B *rds* but increasingly falling behind afterwards.
The superiority of UFS$_{n,cc}$ is easily explained: First, the number of resources in the T2 collection (3,019) is much lower compared to the T1 collection (2,491,785). Even though the resources of the T2 collection are spatially *very* spread ($\rightarrow$ each resource administers data points in a lot of Voronoi cells), this mitigates the greatest problem of UFS$_{n,cc}$ occurring for the T1 collection—namely that in regions with a high global point density,

Table 34: Listing of various key figures alongside the resulting *rds* and *rfc* values for the T2 collection and the benchmarked techniques around 3,000 B *rds*.

technique → / key figure ↓	MBR	UFS_{32768}	KDMBR^{4}_{2048}	$\text{KDQT}^{16,0.1}_{2048}$	$\text{QTMBR}^{4}_{8192,1.0}$	$\text{MBRQT}^{24576,0.1}$
overlap between summaries [in dsu^2]	1.05E+11	1.39E+10	7.58E+09	2.42E+09	3.50E+08	1.68E+08
data space coverage	1,874.4	506.7	159.55	69.44	6.99	4.79
surface area per summary [in dsu^2]	40,231.7	10,876.6	3,424.6	1,490.4	150.1	102.8
number of indexed areas	3,019	16,250,684	4,337,129	9,938,824	3,019,044	7,451,339
aspect ratio (mean) [:1]	2.94	-	2.08	1.53	2.22	2.93
aspect ratio (median) [:1]	2.98	-	1.75	2.00	2.00	2.97
avg *rfc* [in %]	101.4	29.00	53.85	65.72	75.95	74.96
avg *rds* [in byte]	44.00	2,847.6	3,158.1	3,241.4	3,043.3	2,767.7

too many resources administer data points in the query cluster. Second, the description of a single indexed area requires only 1 bit for pure global space partitioning approaches such as $\text{UFS}_{n,cc}$. Since the data points of each resource are widely scattered across the data space and hence a lot of data-point-containing areas need to be indexed to achieve acceptable selectivity, this is the most storage-space-efficient way to adequately describe the spatial footprints. For a local space partitioning quadtree, describing a data-point-containing area is much more expensive in terms of storage space. For example, an $\text{UFS}_{32768,cc}$ summary indexes $16,250,684/3,019 = 5,383$ areas on average whereas an $\text{QTMBR}^{4}_{8192,1.0}$ summary indexes only 1,000 areas on average. Nevertheless, the *rds* value of $\text{UFS}_{32768,cc}$ is even 6.4% lower (see Table 34). Furthermore, the quadtree's property of adapting its space partition to spatially narrow resources very storage-space-efficiently (which is its great edge for the T1 and F collections) is useless for the T2 collection—no spatially narrow resources exist.

The hybrid approaches based on global space partitioning (KDMBR^{b}_{n} and $\text{KDQT}^{c,a}_{n}$) also profit from the storage-space-efficient description base provided, obviously. Still, the refinement of occupied cells is not efficient due to its high storage space costs (in comparison to simply adding more cells to the global space partition) and the sheer amount of cells to refine. For example, $\text{UFS}_{32768,cc}$ results in ~2,848 B *rds* while KDMBR^{4}_{2048} results in ~3,158 B *rds*—despite the amount of global subspaces being 16 times

greater for $\text{UFS}_{32768,cc}$ and a rather coarse refinement for KDMBR^{4}_{2048} with $b = 4$ (also see Table 34). Due to the spatial spread of the resources, refining an occupied cell with a quantized MBR is suboptimal, once again: oftentimes, the MBR will cover large parts of its cell or even the entire cell (as an example, already see Figure 77, bottom). With regard to the spatial accuracy, a sufficiently detailed cell-interior quadtree can be a better description tool (see Figure 74, where $\text{KDQT}^{c,a}_{n}$ offers the best absolute *rfc* value)—at the cost of greater storage space expenses, obviously. Hence, both KDMBR^{b}_{n} and $\text{KDQT}^{c,a}_{n}$ cannot compete with $\text{UFS}_{n,cc}$.

Table 34 depicts the key figures for the approaches in their parameterizations closest to 3,000 B *rds*. By looking at the parameterization of $\text{KDQT}^{16,1.0}_{2048}$, it shows that now, a rather large amount of global subspaces is built ($n = 8,192$) and only rather small quadtrees are used ($c = 16$) for the refinement. Once again, this underlines the just discussed greater efficiency of the global space partitioning for the T2 collection. It is also the opposite from previous observations for the predominant parameterizations for $\text{KDQT}^{c,a}_{n}$ and the T1 collection (see section 7.3). Interestingly, despite exhibiting the worst results for the *rfc* values by far, $\text{QTMBR}^{4}_{8192,1.0}$ and $\text{MBRQT}^{24576,0.1}$ still feature the most favorable area-related key figures.[203] Nevertheless, their spatial accuracy in the regions with a high global point density is much worse compared to $\text{UFS}_{32768,cc}$. See Figure 77 as an example. Obviously, the importance of adjusting the summaries' 'level' of spatial accuracy to the global point density is even more important for the T2 collection than for the other collections. Simply optimizing for the minimization of the covered surface area is not sufficient here.

As a final remark, we want to point out the very high numbers of indexed areas for $\text{KDQT}^{16,1.0}_{2048}$ and $\text{MBRQT}^{24576,0.1}$ which are significantly greater compared to those of KDMBR^{4}_{2048} and $\text{QTMBR}^{4}_{8192,1.0}$. This exhibits that the initial ranking costs of $\text{KDQT}^{16,1.0}_{2048}$ and $\text{MBRQT}^{24576,0.1}$ are much greater—although in comparison, their results are rather mediocre. In this context, remember that for pure global space partitioning approaches, the 'number of indexed areas' does not linearly transfer into the costs for the initial ranking since there are only n different, globally shared indexable areas. Hence, the ranking costs of $\text{UFS}_{32768,cc}$ are not as high as the amount of indexed areas might suggest. All in all, $\text{UFS}_{n,cc}$ is clearly the most suitable approach for the T2 collection.

[203] Notably, it is the first time that $\text{QTMBR}^{b}_{c,a}$ (in the form of $\text{QTMBR}^{4}_{8192,1.0}$) does not feature the smallest 'overlap between summaries', 'data space coverage', and 'surface area per summary'—despite utilizing significantly greater amounts of storage space on average compared to $\text{MBRQT}^{24576,0.1}$. As the exterior MBR reduces the live space for $\text{MBRQT}^{c,a}$ and the T2 collection only marginally, this shows that additional storage space for $\text{QTMBR}^{b}_{c,a}$ would be better invested into additional quadtree cells instead of quantized MBRs.

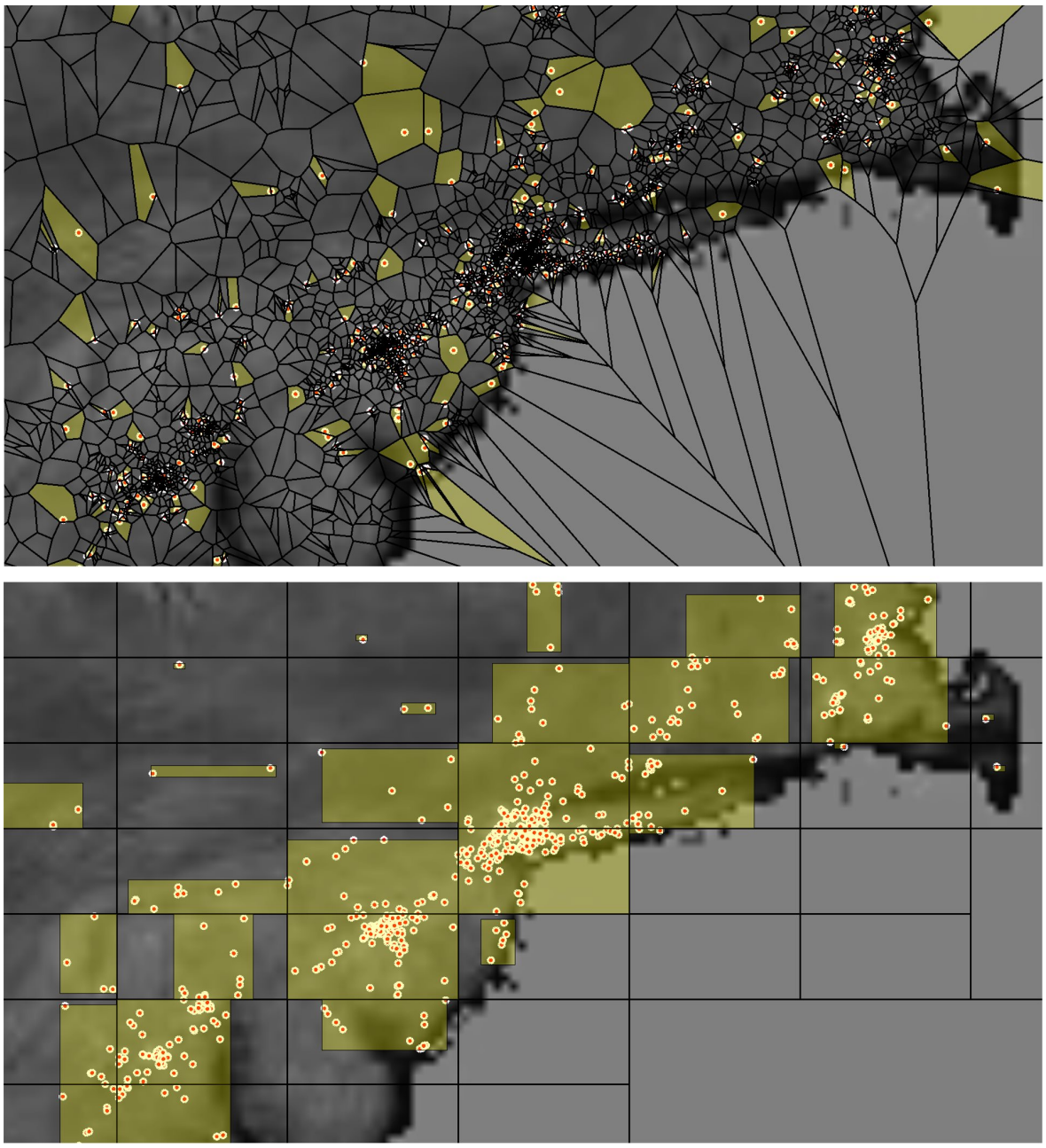


Fig. 77: Exemplary depiction of the $\mathrm{UFS}_{32768,cc}$ (top) and $\mathrm{QTMBR}^{4}_{8192,1.0}$ (bottom) summaries in the region around New York City for resource 0 of the T2 collection.

8.7. Evaluation of the Selected Approaches for Approximate Similarity Search

In this section, we briefly assess the suitability of the approaches for approximate similarity search (see for example [Zezula et al. 1998; Ciaccia and Patella 2000; Amato and Savino 2008]). As noted before, our search situation corresponds to a similarity search. Generally, an approximate similarity search is used to greatly improve the efficiency of the similarity search at the price of some imprecision in the results [Amato and Savino 2008, p. 1]: The results are only approximately correct but the query processing costs are much lower. Approximate similarity search is more common in high-dimensional, metric spaces where distance computations can be very costly and index structures offer only poor partitioning due to the 'curse of dimensionality' (i.e. due to the high number of dimensions, the distances between the data objects are so big that the distances of the nearest and the furthest neighbor to a given query object converge to each other). Hence, oftentimes, almost the entirety of the database has to be examined in order to retrieve the exact set of nearest neighbors *safely*. With an approximate search, these high costs can often be greatly reduced.
Both the high costs for distance calculations as well as the 'curse of dimensionality' do not apply to our search scenario with its two-dimensional spatial data. Thus, the suitability of our approaches for approximate similarity search is not as important as for approaches for metric spaces. Nevertheless, we briefly examine this issue here since also for spatial data, there are application scenarios in which approximated intermediate results are reasonably usable. As a generic example, such results can be utilized in any implementation of an *anytime algorithm*. In general, an anytime algorithm is an algorithm where the quality of the output (i.e. the query result in our case) is improved the longer the algorithm runs [Grass and Zilberstein 1996, p. 1f]. It can be interrupted at any time and returns the partial result that is calculated up to that point—which is then an approximated result, obviously. In our scenario, there are several reasons anytime algorithms are conceivable:

— The computational costs of the initial ranking can be very high for data collections with many resources. For example, with $\mathrm{KDQT}_{32}^{64,1.0E-5}$, the `T1` collection is described by almost 5 million rectangles for which the distances to the query point have to be calculated (see Table 12 on page 183).[204]
— For 'inconvenient' data collections such as the `T2` collection, the share of contacted resources to determine the top 50 data points *safely* can be very high.
— The latency in the network can delay the query processing.

[204] In section 8.8.2, the duration of the initial ranking is briefly evaluated for different techniques.

Thus, several factors can prolong the query processing to such an extent that in a 'real-world' system, it would be necessary to provide users with approximated results whilst the exact result has yet to be determined.
At this point, with respect to evaluating the selected approaches' suitability for approximate similarity search, we limit ourselves to assessing the *relative error* for the different approaches and the T1 collection at 'MBRsize' and at 'MBRselectivity'[205]. Other aspects (such as suitable algorithms for an approximated initial ranking) are out of the scope of this thesis. The same 50 query points as for the other evaluations of the T1 collection are used.

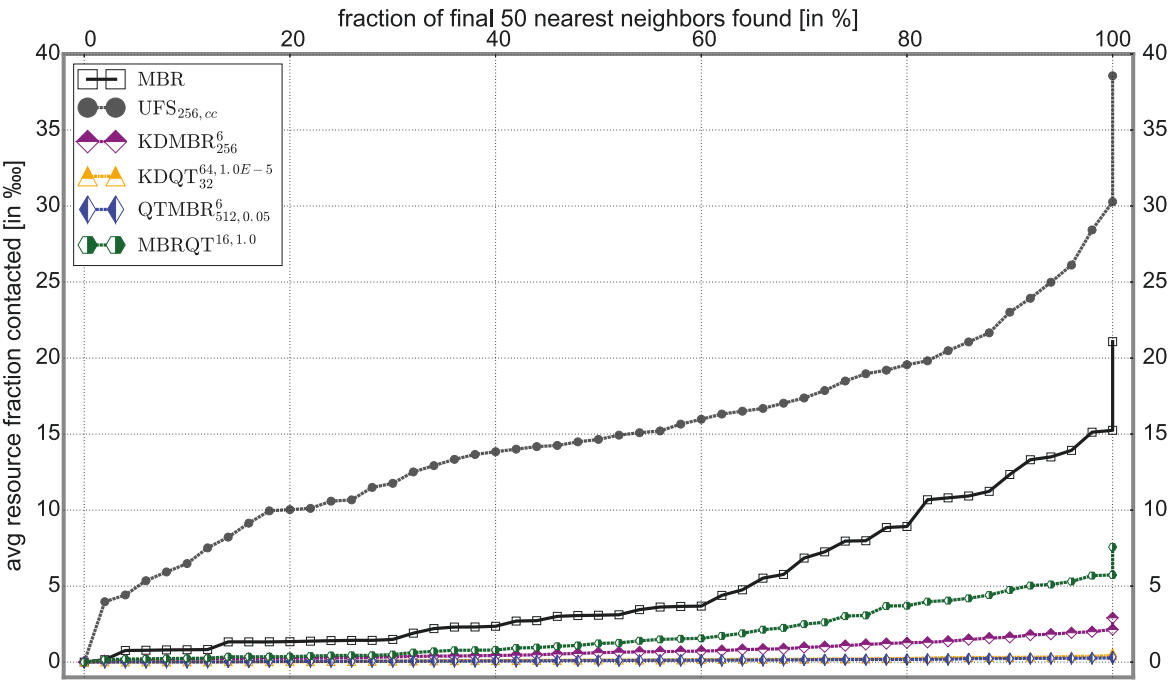


Fig. 78: Depiction of the relative error for the T1 collection and the six further evaluated approaches at 'MBRsize'.

Figure 78 depicts the relative error for the approaches in their respective parameterizations at 'MBRsize'. On the x-axis, the fraction of the final top 50 data points found so far is denoted. On the y-axis, the resource fraction contacted (averaged over the 50 queries) in order to retrieve the corresponding fraction of the top 50 data points is depicted. For example, for $UFS_{256,cc}$, 10.0 ‰ of the resources need to be contacted in order to retrieve 20% of the top 50 data points. At 100% on the x-axis, all curves exhibit two anchor points, resulting in a straight vertical climb of the curves. For example, for $UFS_{256,cc}$, the average resource fraction contacted jumps from 30.2 ‰ to 38.6 ‰. The first of these two anchor points (30.2 ‰ for $UFS_{256,cc}$) marks the resource fraction contacted after which all of the top 50 data points have actually been found. The second anchor point (38.6 ‰ for $UFS_{256,cc}$)

[205] At 'MBRselectivity' means that for each approach, the parameterization is assessed whose resulting *rfc* value is closest to the *rfc* value of the MBR approach.

denotes the resource fraction contacted after which the kNN algorithm terminates and thus, the top 50 data points have been determined *safely*, i.e. it is the final *rfc* value. Note that in the final *rfc* value, the final contact to the resources contributing to the final query result (for the media items associated with the relevant data points) is not included.
In general, at 'MBRsize', the results for the relative error comply with the final *rfc* values. For example, $\text{UFS}_{256,cc}$ exhibits the worst final *rfc* value and also requires contacting the most resources for retrieving arbitrary fractions of the top 50 data points. Comparing the courses of the curves, the curves are generally rather flat at the early stages (i.e. when rather low fractions of the top 50 data points have been found) and increasingly ascend at the later stages (when large fractions of the top 50 data points have been found). This means that the first few of the relevant data points are found quickly but finding the last few requires contacting a disproportionate amount of resources. The only exception is $\text{UFS}_{256,cc}$ whose curve is also notably steep at the early stages. Hence, $\text{UFS}_{256,cc}$ requires contacting fairly many resources to find even the first few of the final top 50 data points. Since only the globally shared Voronoi cell polygons are available as indexable areas, the information of the $\text{UFS}_{256,cc}$ resource summaries is not suitable for selectively exploring the immediate spatial neighborhood of the query point. In addition, as discussed previously, the areas indexed for different resources by $\text{UFS}_{256,cc}$ summaries are often enough exactly the same. This results in bigger resources to be ranked better. Since the query points of the `T1` collection are randomly selected from a separate training data set and the share of small resources is very high, there is no tendency that the relevant data points are primarily administered by big resources—unlike *queryMode1* for the `F` collection (see section 8.5). In total, the $\text{UFS}_{256,cc}$ resource ranking is rather random with regard to the immediate spatial neighborhood of the query point. Note that we also exemplarily looked into the results for *queryMode1* and the `F` collection at 'MBRsize'. The general courses of the curves are similar there, i.e. $\text{UFS}_{256,cc}$ still requires contacting disproportionately more resources in order to retrieve small fractions of the final top 50 data points compared to other techniques—even though the query points for *queryMode1* mostly originate from big resources which are strongly preferred in the $\text{UFS}_{256,cc}$ ranking. Thus, for the `T1` collection and its query points, the strong preference of big resources only amplifies the inherent shortcomings of the $\text{UFS}_{n,cc}$ approach. Overall, the *general* underlying problem for $\text{UFS}_{n,cc}$ is that too many resources administer data points in the single Voronoi cells. The consequential lack of fine-grained, 'distinct' spatial information concerning the immediate neighborhood of the query point is unique to $\text{UFS}_{256,cc}$ (respectively to pure global space partitioning approaches in general) and makes it (them) especially unsuited for approximate similarity search—at least for data collections with the characteristics of the `T1` and the `F` collection.

Concerning the other techniques, no specific anomalies are apparent from Figure 78. The better the final *rfc* value, the better the technique is also suited for approximate similarity search. The courses of the curves of $\text{KDQT}_{32}^{64,1.0E-5}$ and $\text{QTMBR}_{512,0.05}^{6}$ are not well perceptible in Figure 78. In a mutual comparison (not depicted in this work), it is observable that both curves exhibit almost identical courses, with slight advantages for $\text{QTMBR}_{512,0.05}^{6}$ in the main part. This advantage grows slowly as the 100% are approached.

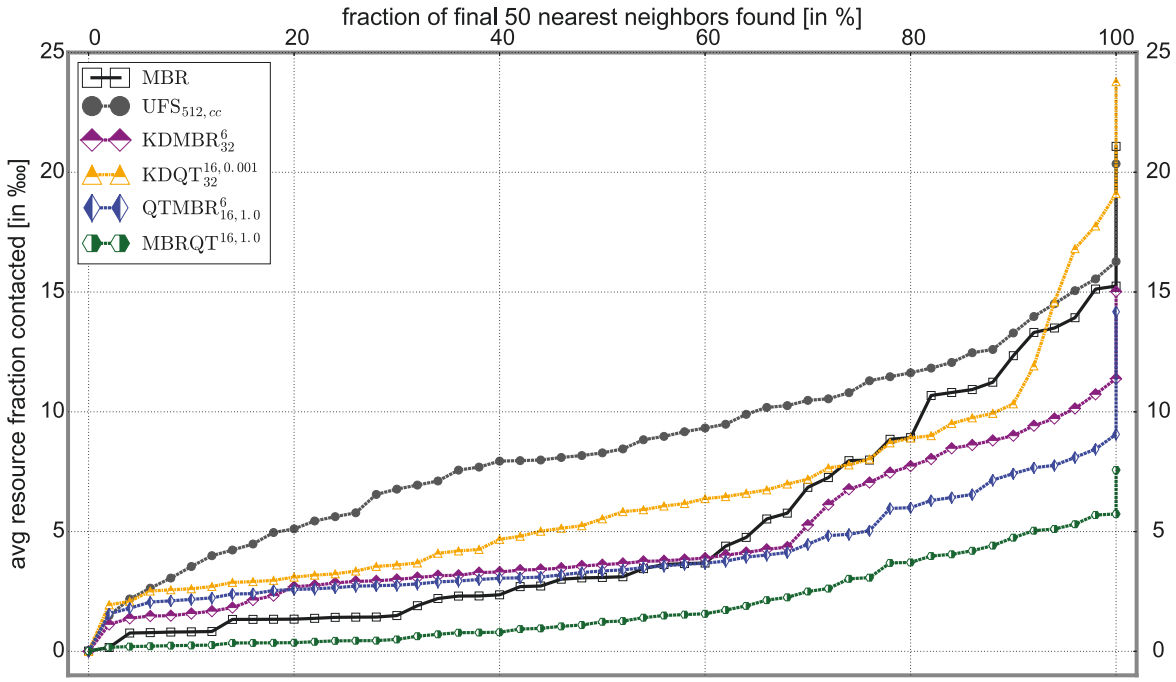


Fig. 79: Depiction of the relative error for the T1 collection and the six further evaluated approaches at 'MBRselectivity'.

Figure 79 shows the results for the approaches at 'MBRselectivity'. Since the final *rfc* values are now much closer together, there are also more intersections between the respective curves of the techniques (at 'MBRsize', only the $\text{KDQT}_{32}^{64,1.0E-5}$ and $\text{QTMBR}_{512,0.05}^{6}$ curves intersected each other). Once again, it is evident that $\text{UFS}_{256,cc}$ requires contacting more resources to retrieve the first few of the final top 50 data points. This comparative weakness of $\text{UFS}_{n,cc}$ in finding even small fractions of the final result is observable for all assessments of the relative error we have looked into. Thus, it is an inherent property of $\text{UFS}_{n,cc}$ (and equally for all other pure global space partitioning approaches in general) for which the obvious reasons have already been discussed.
Concerning the other techniques, both the curves of the MBR approach as well as $\text{MBRQT}^{16,1.0}$ are remarkably flat at the beginning, i.e. the first few of the final top 50 data points are found very soon. We also observed this phenomenon in other assessments of the relative error, for example for *queryMode1* and the F collection. Apparently, this is an effect of the uti-

lization of *full-precision* MBRs by the respective summaries. Full-precision MBRs can be arbitrarily small such that e.g. for a resource administering several data points at exactly the same location, the corresponding description of both the MBR summary as well as the $\text{MBRQT}^{16,1.0}$ summary is an MBR corresponding to a point (remember that for $\text{MBRQT}^{c,a}$, an internal quadtree is only built if the MBR's surface area is greater than the threshold surface area a). In contrast, for all other techniques, such resources are described by an *approximated indexed area* exhibiting a surface area greater 0 dsu^2:

— For KDMBR^{6}_{32} and $\text{QTMBR}^{6}_{16,1.0}$, this area is a cell of the quantization grid invoked into a cell of the basic space partition.
— For $\text{KDQT}^{16,0.001}_{32}$, it is a quadtree cell of the k-d-cell-interior quadtree (whose build is stopped when $a = 0.001$ has been undercut for all occupied cells).
— For $\text{UFS}_{512,cc}$, it is an entire, globally shared Voronoi cell polygon.

Hence, the MBR approach and $\text{MBRQT}^{16,1.0}$ find the first few of the top 50 data points fastest since they are the only techniques which depict spatially *extremely narrow* resources with 'infinite accuracy', facilitating the exploration of the query points' immediate neighborhood in the first place. For greater fractions of final top 50 data points found, we recurringly observed the curves of at least the MBR approach to be steeper than those of the other approaches. We attribute this to the relative coarseness of the MBR summaries for resources which are spatially *a bit* more spread than the extremely narrow resources. At least in case parameter a is set to a rather large value, $\text{MBRQT}^{c,a}$ seems to behave similarly.
Apart from the inappropriateness of $\text{UFS}_{n,cc}$ (respectively pure global space partitioning approaches in general) and the special suitability of the MBR approach and $\text{MBRQT}^{c,a}$ (respectively approaches based on full-precision rectangles) for retrieving the first few of the final top 50 data points, we could not find any constantly recurring phenomena in the data we looked into. From what we have assessed, the suitability of the remaining approaches is somewhere in between both other classes of approaches. *Within* the various classes, the suitability of the approaches seems to correlate fairly well with the resulting final *rfc* values, i.e. the better the final *rfc* value of an approach, the better the suitability at all 'stages' of the approximate search (such as when 50% or 70% of the final top 50 data points have been found). A more detailed analysis which could reveal new insights is left open for future work at this point, though.
As a final remark, let us emphasize that all of our approaches are well-suited for approximate similarity search in case the spatial domain ranker is used. For example, an approximate similarity search algorithm can be specified to terminate when a predefined threshold T (e.g. $T = 50\%$) of the final query result data points have been safely determined. As outlined in section 5.2, in the kNN algorithm, the current query result points are

maintained in a `topk[]` array while the remaining resources which still need to be assessed for relevant data points are maintained in a list L_r. For the spatial domain ranker, the resources in L_r are sorted in ascending order by the lower bound distances of the resources' spatial footprints to the query point.[206] The approximate similarity search can be terminated when $\geq T$ of the data points in the `topk[]` array exhibit a smaller distance to the query point than the lower bound distance of the first resource in L_r. This is easily determinable and therefore, all of our approaches are well-suited for approximate similarity search.

[206] Remember that for all approaches except the MBR approach, a resource can be described by several indexed areas. The lower bound distance of such a resource's spatial footprint is then the smallest MINDIST of any of these areas to the query point.

8.8. Excursus: Evaluation of Additional Properties

In this section, we briefly evaluate two additional properties which did not fit into the previous evaluations: the reduction of the query radius in advance of actually processing an kNN query (see section 8.8.1), and the duration of the initial resource ranking (see section 8.8.2).

8.8.1. EVALUATION OF THE QUERY RADIUS REDUCTION. One of the optimizations we conduct for our search system is to reduce the initial query radius for the kNN queries (see section 6.4.2). In this excursus, we briefly evaluate the query radius reduction achieved by the techniques at 'MBRsize' of the T1 collection (see section 8.1.1).
As described in section 6.4.2, three lists are managed for reducing the query radius of a kNN query:

— A list which captures the distances of the k nearest neighbors to the query point from the set of directly transmitted data points. We refer to the distance of the k-th nearest neighbor of this list as *dtRad* radius in the following.
— A list which captures the k smallest upper bound distances for the areas indexed by the summaries. It is guaranteed that the k nearest neighbors are within the distance of the k-th smallest upper bound distance as each indexed area contains at least one data point. We refer to this distance as *sumRad* radius in the following.
— A mutual top-k-list created from both previously mentioned lists. It contains the k smallest distances from both individual lists. We refer to the k-th smallest distance of the top-k-list as *finalRad* radius in the following. Generally, it is the smallest initial query radius which can be determined ($dtRad \geq finalRad \leq sumRad$).

For each technique, we record the three radii *dtRad*, *sumRad*, and *finalRad* occuring over the 50 queries of the T1 collection. The aggregated results are presented in Figure 80 (*dtRad*), Figure 82 (*sumRad*), and Figure 83 (*finalRad*). In each figure, the mean values (bar chart on the respective left) and the descriptive statistic values (boxplot on the respective right) are shown for each technique. The y-axis of the boxplots is always *log-scaled* due to the wide spread of the single occuring radii.[207] Some key figures relevant for the evaluation of the query radius reduction are listed in Table 35. In the following, we briefly assess the *dtRad* and the *sumRad* results before concluding with the *finalRad* results.

Assessment of the *dtRad* Results. In general, the intuitive expectation is that the more data points are directly transmitted, the lower the resulting *dtRad* radii should be. This intuition is based on the assumption of

[207] Therefore, the height of a *linear-scaled* mean-value-bar does not match the position of its boxplot's mean.

Table 35: Listing of several key figures for the techniques at 'MBRsize' of the T1 collection.

technique → / key figure ↓	MBR	$UFS_{256,cc}$	$KDMBR^{6}_{256}$	$KDQT^{64,1.0E-5}_{32}$	$QTMBR^{6}_{512,0.05}$	$MBRQT^{16,1.0}$
number of indexed areas	1,187,747	1,739,402	1,617,235	4,947,405	2,760,692	1,363,445
directly represented resources [in %]	52.33	48.82	48.24	34.21	42.00	52.33
number of directly transmitted data points	1,691,700	1,765,731	1,734,446	908,236	1,202,490	1,691,700
avg number of data points per directly represented resource	1.30	1.45	1.44	1.07	1.15	1.30
avg *rfc* [in ‰]	21.13	38.62	2.97	0.65	0.39	7.62

an *equal distribution*[208] of the directly transmitted data points in the data space. In Table 35, the techniques' shares of directly represented resources are listed, once again. Interestingly, the numbers of directly transmitted data points do not absolutely coincide with the shares of directly represented resources: For example, the MBR approach (52.33%) has a greater share of directly represented resources than $UFS_{256,cc}$ (48.82%) but 74,031 more data points are directly transmitted for $UFS_{256,cc}$. In general, the directly represented resources of $UFS_{256,cc}$ and $KDMBR^{6}_{256}$ administer more data points than the directly represented resources of the other techniques (see Table 35). Thus, on the basis of these numbers, the expectation is that the *dtRad* radii for $UFS_{256,cc}$ and $KDMBR^{6}_{256}$ are the smallest whereas for $KDQT^{64,1.0E-5}_{32}$, they should be the greatest. The results for the *dtRad* radii are displayed in Figure 80.

While the relative results for the MBR approach, $MBRQT^{16,1.0}$, and $QTMBR^{6}_{512,0.05}$ are within the expectations (i.e. the *dtRad* radii for the MBR approach and $MBRQT^{16,1.0}$ are lower than for $QTMBR^{6}_{512,0.05}$)[209], the three techniques involving global space partitioning ($UFS_{256,cc}$, $KDMBR^{6}_{256}$, and $KDQT^{64,1.0E-5}_{32}$) show strong deviations. For example, the number of directly transmitted data points is much larger for $UFS_{256,cc}$ (1,765,731) compared to $KDQT^{64,1.0E-5}_{32}$ (908,236). Nevertheless, the mean *dtRad* ra-

[208] By equal distribution, we mean that the spatial distribution of the directly transmitted data points is in accordance with the general spatial distribution of the data points (see Figure 30b on page 113) and that there are no specific regions for which e.g. preferably no data points are directly transmitted.

[209] In fact, the results for the MBR approach and $MBRQT^{16,1.0}$ are identical because for both, the exact same set of resources is directly represented.

dius is significantly greater for $\text{UFS}_{256,cc}$ (0.22 *dsu* as opposed to 0.11 *dsu* for $\text{KDQT}_{32}^{64,1.0E-5}$). In fact, the relative results for these three techniques are the exact opposite of the expectation (i.e. for example $\text{KDQT}_{32}^{64,1.0E-5}$ has the best mean *dtRad* radius although its share of directly represented resources is the lowest).

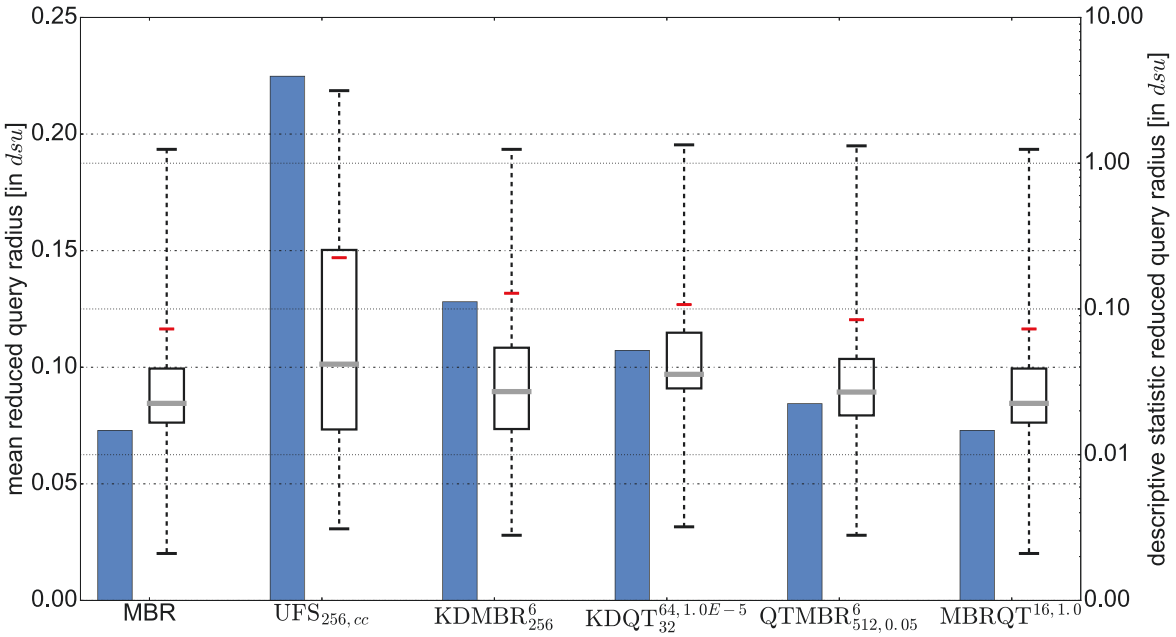


Fig. 80: Overview of the results for the *dtRad* radii occuring for the 50 queries of the T1 collection. The bar charts (linear-scaled, left y-axis) depict the mean values whereas the boxplots (log-scaled, right y-axis) depict the descriptive statistic values.

The most natural assumption is that for these techniques, the directly transmitted data points are not equally distributed in the data space. To verify this, we plotted the directly transmitted data points of the different techniques into the data space. Figure 81 exemplarily shows the plots for $\text{UFS}_{256,cc}$ (top) and $\text{KDQT}_{32}^{64,1.0E-5}$ (bottom).[210] The depictions are confined to the North American continent. Clear differences between both plots are visible, the distribution of data points displays anomalies for both techniques: For $\text{UFS}_{256,cc}$, the western half of the United States is very sparsely populated with directly transmitted data points while for $\text{KDQT}_{32}^{64,1.0E-5}$, there are two quadratic areas (around Lake Michigan and at the northern US East Coast) which are almost completely free of directly transmitted data points. Similar anomalies are observable for KDMBR_{256}^{6} (where e.g. the Iberian Peninsula is almost non-populated).
The reason behind these anomalies is the byte-aligned bit vector clipping: Since *always*, the most storage-space-efficient resource description is selected, the summary representation is possibly preferred over the direct

[210] Visualizations of the directly transmitted data points of all assessed techniques can be found in the appendix (see appendix A beginning on page 453).

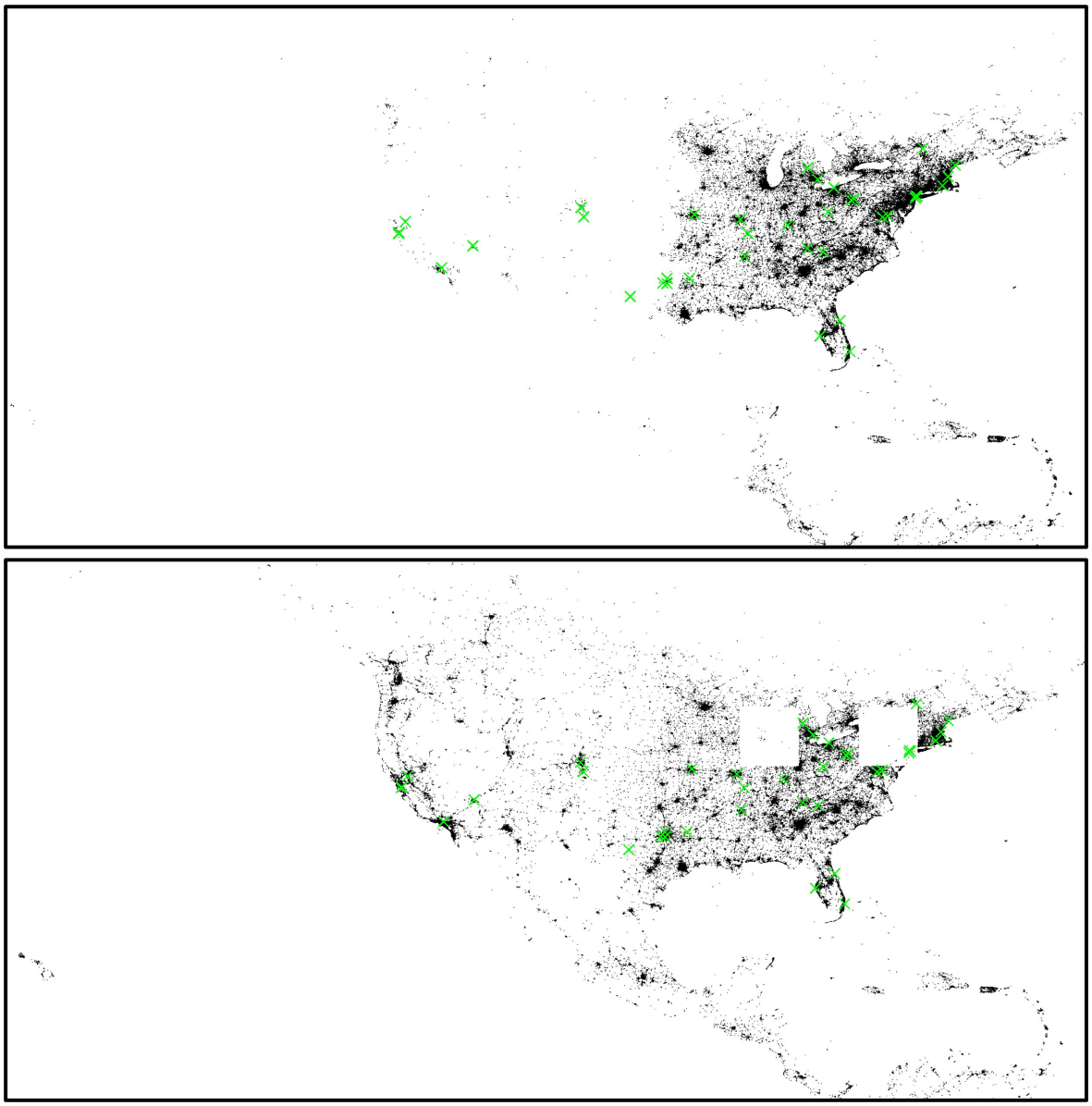


Fig. 81: Directly transmitted data points for $\text{UFS}_{256,cc}$ (top) respectively $\text{KDQT}_{32}^{64,1.0E-5}$ (bottom) and the T1 collection. The green crosses depict the locations of query points.

representation of a resource even if it administers only one or two data points—in case the clipped bit vector of the summary requires less storage space than the data point coordinates. For pure global space partitioning approaches (such as $\text{UFS}_{256,cc}$), this can happen if a resource administers

[211]The great presence of low-ID Voronoi cells in the western half of the United States is actually an effect from an optimization we conduct to speed up the calculation of the Voronoi-based approaches' summaries: As already mentioned in section 6.4.3, the Voronoi sites are sorted in ascending order by a) their *long* value and b) (in case of equal *long* values) their *lat* value. This allows for determining the Voronoi cell of a data point by considering way less than all of the n sites. Note that in Figure 81 (top), there are still some directly transmitted data points in e.g. Hawaii. In general, two reasons exist that directly transmitted data points with very low *long* values may occur for the Voronoi-based approaches:

its data points *exclusively* in subspaces with a very low ID (as the cell occupancy information of low-ID cells is first in the summary's bit vector). This also works for the hybrid techniques KDMBR^{6}_{256} and $\text{KDQT}^{64,1.0E-5}_{32}$ (which are *based* on a global space partitioning) as the refinement information immediately follows the '1' for its occupied subspace—in case the occupied cells' IDs are low and the refinement information is concise enough. In Figure 81, it can thus be seen that for e.g. $\text{UFS}_{256,cc}$, several low-ID Voronoi cells are located in the western half of the United States.[211] For $\text{KDQT}^{64,1.0E-5}_{32}$, two low-ID k-d cells are located at Lake Michigan and the northern US East Coast.

Consequently, the assumption of an equal distribution of directly transmitted data points does not hold for approaches utilizing global space partitioning. For these, the achieved *dtRad* radii are strongly dependent on the query location, the assignment of IDs to the cells, and the global data point density of these cells. That means a strong external influence is present for these approaches, explaining the observed anomalies. For example, the query points are obviously more favorably located for $\text{KDQT}^{64,1.0E-5}_{32}$ than for $\text{UFS}_{256,cc}$ which is evident from the results and also from the visual depiction in Figure 81. In contrast, the relative results for the MBR approach, $\text{MBRQT}^{16,1.0}$, and $\text{QTMBR}^{6}_{512,0.05}$ are in perfect congruence with the expectation as their directly transmitted data points are equally distributed.

Assessment of the *sumRad* Results. In Figure 82, the results for the *sumRad* radii are displayed. For these, the intuition is that the better the resulting *rfc* values of a technique (see Table 35), the more accurate its summaries in the neighborhood of the query points and therefore, the smaller the *sumRad* radii should be.

For the most part, the results are within the expectations. For $\text{UFS}_{256,cc}$, the *sumRad* radii are by far the greatest which is due to the coarse, globally shared Voronoi cell polygons being the only indexing tool: They do not allow the computation of small upper bounds. In fact, the *minimally* occurring *sumRad* radius of $\text{UFS}_{256,cc}$ (0.086 *dsu*) is even greater than the *mean sumRad* radii of $\text{KDQT}^{64,1.0E-5}_{32}$ (0.035 *dsu*) and $\text{QTMBR}^{6}_{512,0.05}$ (0.044 *dsu*). As a bit of a surprise, the *sumRad* radii for the MBR approach and $\text{MBRQT}^{16,1.0}$ are smaller compared to those of KDMBR^{6}_{256}—despite the latter being significantly more selective. Generally, the KDMBR^{6}_{256} sum-

— The site of these data points' Voronoi cell is located on the other side of the data space but its Voronoi cell crosses the International Date Line. This is *not* the case for the directly transmitted data points in Hawaii for $\text{UFS}_{256,cc}$—the Voronoi cell containing Hawaii does not cross the International Date Line (verified by means of our visualization tool).

— A resource e.g. administers two data points: One in Hawaii (in a cell with a very low ID), and one in a cell much further to the east (in a cell with a significantly greater ID) such that the bit vector clipping is not as effective or not applied at all for the resource's summary. In such cases, the resource will still be directly represented for $\text{UFS}_{256,cc}$.

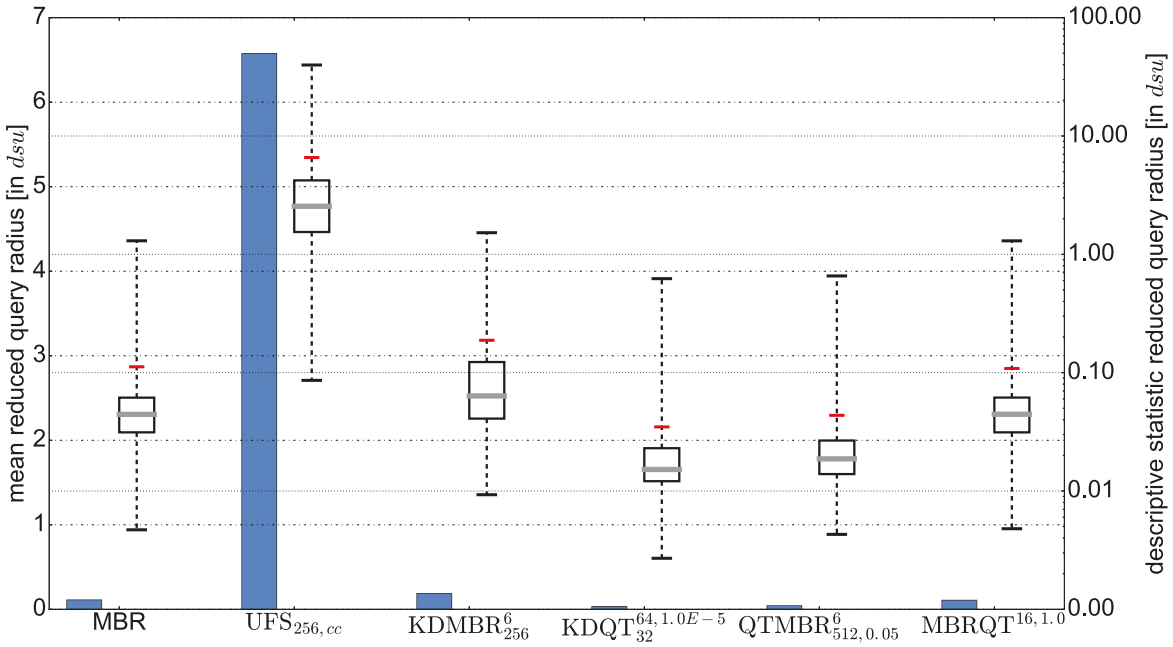


Fig. 82: Overview of the results for the *sumRad* radii occuring for the 50 queries of the T1 collection. The bar charts (linear-scaled, left y-axis) depict the mean values whereas the boxplots (log-scaled, right y-axis) depict the descriptive statistic values.

maries are limited in their maximum accuracy: the smallest indexable spatial unit is an MBR embedded into the quantization grid of an occupied cell. Furthermore, it has been shown before that quantized MBRs oftentimes cover considerable parts of their cell. In contrast, the full-precision MBRs of the MBR approach and $\mathrm{MBRQT}^{16,1.0}$ can be arbitrarily small. As only the $k = 50$ smallest upper bounds are required for the *sumRad* radius of a query, the combination of circumstances obviously works out better for the MBR approach and $\mathrm{MBRQT}^{16,1.0}$—even though the number of indexed areas is clearly higher for KDMBR^{6}_{256}. Generally, the results for the MBR approach and $\mathrm{MBRQT}^{16,1.0}$ are very similar: for example, the mean *sumRad* radius is only 3.2% smaller for $\mathrm{MBRQT}^{16,1.0}$ compared to the MBR approach. This was to be expected since not many MBRs are refined for the given parameterization of $\mathrm{MBRQT}^{16,1.0}$, though.

The smallest *sumRad* radii are offered by the two most selective techniques: $\mathrm{KDQT}^{64,1.0E-5}_{32}$ and $\mathrm{QTMBR}^{6}_{512,0.05}$. Interestingly, the mean *sumRad* radius of $\mathrm{KDQT}^{64,1.0E-5}_{32}$ is smaller compared to $\mathrm{QTMBR}^{6}_{512,0.05}$ despite worse selectivity. This is easily explained, though: Since only one data point can be assumed per indexed area, the significantly greater 'number of indexed areas' obviously has a positive effect on the *sumRad* radii of $\mathrm{KDQT}^{64,1.0E-5}_{32}$ ($\mathrm{KDQT}^{64,1.0E-5}_{32}$ has almost 2.2 million more indexed areas). In general, the spatial accuracy offered by the $\mathrm{KDQT}^{64,1.0E-5}_{32}$ and $\mathrm{QTMBR}^{6}_{512,0.05}$ summaries in combination with their significantly greater amount of indexed areas is obviously great enough to achieve better results for all descriptive statistic *sumRad* values (i.e. min, 25%-quantile, etc.) than the 'ar-

bitrarily accurate' MBR and $\text{MBRQT}^{16,1.0}$ summaries. This is in spite of the limitation that the smallest indexable spatial units for $\text{KDQT}_{32}^{64,1.0E-5}$ and $\text{QTMBR}_{512,0.05}^{6}$—similar to KDMBR_{256}^{6}—are approximated, quantized areas.

Assessment of the *finalRad* Results. The *finalRad* radius results from combining the k smallest distances of both lists from which the *dtRad* radius and the *sumRad* radius are determined. An *a priori* intuition for the *finalRad* results is not easy to obtain: For example, the absolute number of directly transmitted data points respectively indexed areas is certainly important. Then, the general accuracy and conciseness of the indexed areas plays a role. Furthermore, it has to be considered that from the summaries, only upper bounds can be determined (which are pessimistic estimates with regard to the distance *and* the number of data points contained in an area since only one data point can be assumed) while for the directly transmitted data points, it is the exact distance (and therefore the optimal 'estimate'). Thus, it is hard to generally foresee whether the *finalRad* radius is dominated by the directly transmitted data points or by the summaries' upper bounds.

Table 36: Overview of the diverse query radii resulting for the different techniques at 'MBRsize' of the T1 collection.

technique → / key figure ↓	MBR	$\text{UFS}_{256,cc}$	KDMBR_{256}^{6}	$\text{KDQT}_{32}^{64,1.0E-5}$	$\text{QTMBR}_{512,0.05}^{6}$	$\text{MBRQT}^{16,1.0}$
mean *dtRad* [in *dsu*]	0.0729	0.2248	0.1281	0.1072	0.0844	0.0729
mean *sumRad* [in *dsu*]	0.1121	6.5753	0.1878	0.0348	0.0436	0.1085
mean *finalRad* [in *dsu*]	0.0433	0.2171	0.0664	0.0309	0.0371	0.0433

This is also reflected in the diversity of the results. For $\text{KDQT}_{32}^{64,1.0E-5}$ and $\text{QTMBR}_{512,0.05}^{6}$, very accurate summaries exist which index a lot of areas. At the same time, the number of directly transmitted data points is comparatively low for them. Consequently, their mean *sumRad* radius is significantly smaller than their mean *dtRad* radius. In total, the result is that their mean *finalRad* radius is rather moderately smaller than the mean *sumRad* radius: The improvements of also taking the directly transmitted data points into account are only 11.2% ($\text{KDQT}_{32}^{64,1.0E-5}$) respectively 14.9% ($\text{QTMBR}_{512,0.05}^{6}$). Also see Table 36. For $\text{UFS}_{256,cc}$, it is the other way

round: Due to the coarse indexed areas of the summaries, the mean *sumRad* radius is very large while the mean *dtRad* radius is tremendously smaller (0.22 *dsu* as opposed to 6.58 *dsu*). Consequently, the mean *finalRad* radius is hardly an improvement over the mean *dtRad* radius for $\text{UFS}_{256,cc}$ (only 3.4% smaller). For the remaining techniques (the MBR approach, KDMBR^{6}_{256}, and $\text{MBRQT}^{16,1.0}$), the mean *dtRad* radii are significantly smaller than the mean *sumRad* radii (all between ~32% and 35%). This is also the case for KDMBR^{6}_{256} even though the numbers of indexed areas (upper bounds) and directly transmitted data points (exact distances) are very similar: there are only 7.2% more directly transmitted data points (this difference is much bigger for the MBR approach and $\text{MBRQT}^{16,1.0}$). Nevertheless, for these three techniques, the combination of both lists results in significant improvements of the mean *finalRad* radii compared to the mean *dtRad* radii: For the MBR approach and $\text{MBRQT}^{16,1.0}$, it is 40.6% while for KDMBR^{6}_{256}, it is even 48.2%. Hence, these techniques benefit greatly from the combination of the directly transmitted data points *and* the upper bounds provided by the summaries. In contrast, for $\text{UFS}_{256,cc}$, the consideration could in principle also be restricted to the directly transmitted data points. For $\text{KDQT}^{64,1.0E-5}_{32}$ and $\text{QTMBR}^{6}_{512,0.05}$, the consideration of the summaries would *possibly* be sufficient.

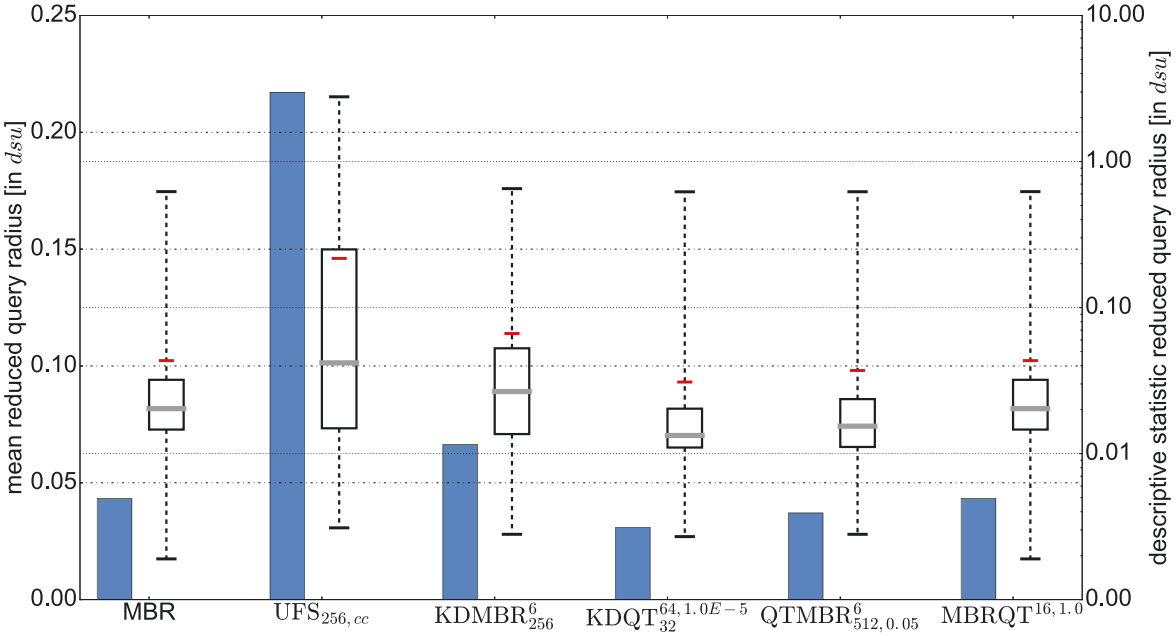


Fig. 83: Overview of the results for the *finalRad* radii occuring for the 50 queries of the T1 collection. The bar charts (linear-scaled, left y-axis) depict the mean values whereas the boxplots (log-scaled, right y-axis) depict the descriptive statistic values.

In total, the best query radius reduction is achieved for $\text{KDQT}^{64,1.0E-5}_{32}$ and $\text{QTMBR}^{6}_{512,0.05}$ (see Figure 83). The MBR approach and $\text{MBRQT}^{16,1.0}$ come surprisingly close (considering their comparatively poor *rfc* values), though. $\text{UFS}_{256,cc}$ is far behind. Note that the average distance of the 50th

nearest neighbor is 0.0056 *dsu*. Hence, the best achieved mean *finalRad* radius (0.0309 *dsu* for $\text{KDQT}_{32}^{64,1.0E-5}$) is still about 5 times greater. In general, it shows that the query radius reduction is primarily dependent on the summaries. In case the indexed areas are very accurately delineated and numerous, the smallest initial query radii can be achieved. For coarse summaries, the upper bounds are not small enough to contribute much to the *finalRad* radius—which is then almost completely dependent on the directly transmitted data points. As for the techniques at 'MBRsize', the amount of directly transmitted data points is rather small in comparison to the total amount of data points, the achievable *dtRad* radii are worse than the *sumRad* radii of the techniques indexing a lot of areas, though. In addition, the directly transmitted data points are non-equally distributed for the approaches involving global space partitioning which further impedes the conditions for those. Still, for 'medium-accurate' summaries, the combined consideration can significantly improve the effectivity of the query radius reduction.
All in all, it has to be stated that the achieved query radius reduction does not fully coincide with the resulting selectivity which might be the first, naive intuition.

8.8.2. EVALUATION OF THE DURATION OF THE INITIAL RESOURCE RANKING. In this section, we *briefly* assess the duration of the initial resource ranking.[212] Note that this section is intended to give a *general impression* of the relative runtimes, only. In no way, the time measurements depict the best achievable ranking duration performance: For the implementation, the focus was on the correctness, and no profiling with subsequent code optimization has been conducted.
Generally, we distinguish between two scenarios for the time measurements:

— In scenario a), the minimization of the network traffic is the sole objective. It is assumed that the storage and memory space which is occupied on the single resources is no critical factor. Consequently, for the time measurements, it is presumed that on the resources which issue queries, the data structures representing the indexed areas of the resources are cached and that the surfaces of each indexed area have already been computed. Thus, for each query, the time measurement only encompasses the update of the `R-Entries`[213] and the subsequent ranking of the resources.

[212] Note that the further processing of the kNN algorithm after the initial ranking is *not* part of the time measurements since it is dependent on the network latency, the response time behaviour of the resources, and so on. From the computational costs, the further kNN processing is very cheap, though.

[213] Since the surfaces of all indexed areas are already computed here, the update of a resource's `R-Entries` comprises of i.) the MINDIST calculation between each indexed area and the query point (and the subsequent update of the information held by the corresponding `R-Entry`), and ii.) the sorting of the resource's `R-Entries` (by MINDIST to the query point as the first criterion and minimum surface area as the second criterion).

— In scenario b), it is assumed that also the storage and memory space which is occupied on the single resources shall be minimized. Since then, only the binary resource description data is at hand on the resources which issue queries, for each query, the time measurements involve reconstructing the indexed areas from the summaries' bit vectors, creating the resources' `R-Entries`[214], and the subsequent ranking of the resources.

The experiments are executed on desktop computers equipped with 64-bit Windows 10 Enterprise LTSB, an Intel i7-7700@3.6 GHz, and 64 GB of DDR4 memory (@2,400 MHz). The time measurements are conducted for the techniques at 'MBRsize' of the `T1` collection, once again. For each of the 50 queries, the duration of the initial resource ranking is recorded.
For $\text{UFS}_{256,cc}$, the creation respectively the update of the `R-Entries` is implemented in the way described at several occasions before: For the 256 polygons of the Voronoi diagram, the distances to the query point and eventually the surface areas are calculated, first. Then, the `R-Entries` for the single resources are updated/built from the cached results. For scenario b), the $\text{UFS}_{256,cc}$ time measurements include the explicit reconstruction of the Voronoi diagram from the set of sites.
In Figure 84, the results for the techniques are shown. The blue boxplots to the respective left show the aggregated results for scenario a) while the red boxplots on the right show the aggregated results for scenario b). In general, it is evident that the runtimes for the initial ranking are very stable for all of the techniques. As for each ranking, only the query points differ but the set of resources and their descriptions are the same for a specific technique, significant divergences would have been surprising.
For scenario a), the runtimes of the different techniques conform quite well to the 'number of indexed areas', i.e. the more indexed areas, the larger the duration of the initial ranking. Also see Table 37 where the number of indexed areas and the mean duration of the initial ranking of the MBR approach are put as 100% while the results of the remaining techniques are depicted as relative percentages. Nevertheless, despite a certain degree of conformity, the number of indexed areas does not translate linearly into the duration of the initial ranking. For example, $\text{KDQT}_{32}^{64,1.0E-5}$ has 417% of the indexed areas but only 185% of the mean initial ranking duration of the MBR approach.[215] This shows that the amount of distance calculations is not the single factor deciding on the ranking runtime. Other time-relevant processes are the update of the `R-Entries` or the ranking process itself,

[214]The creation of a resource's `R-Entries` comprises of i.) the MINDIST calculation of each indexed area to the query point, ii.) the surface area calculation for each indexed area, iii.) the creation of the `R-Entry` objects (in conjunction with setting both attribute values), and iv.) the sorting of the resource's `R-Entries` (by MINDIST to the query point as first criterion and minimum surface area as second criterion).
[215]In the following, we refer to e.g. '417% of the indexed areas of the MBR approach' as '417% indexed areas'.

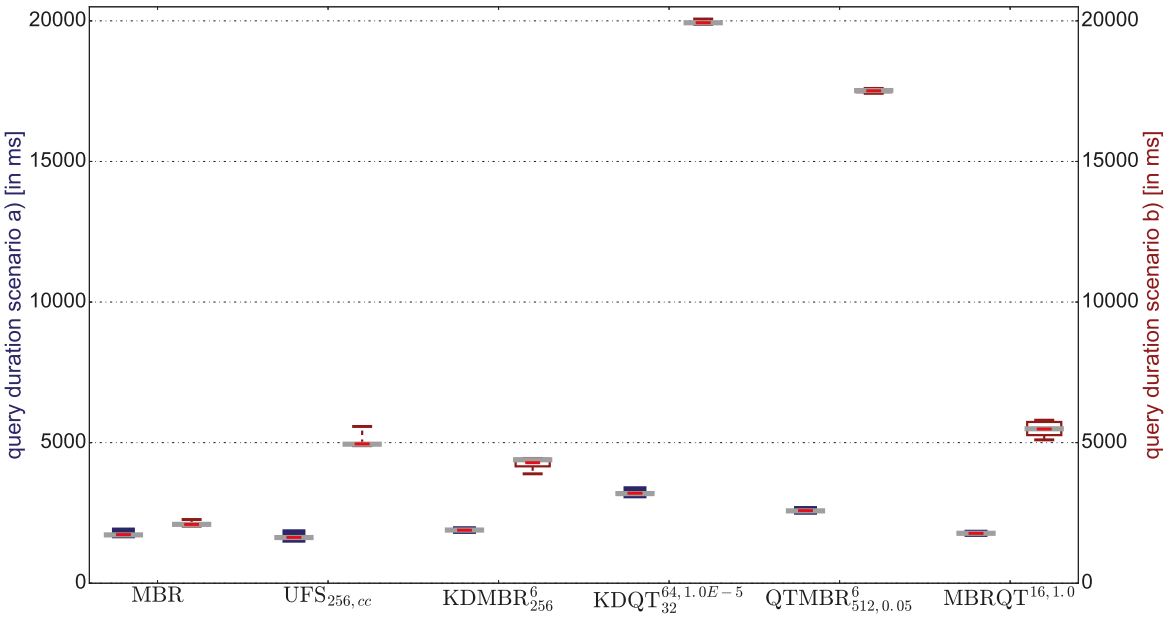


Fig. 84: Aggregated ranking duration results for the techniques at 'MBR-size' for the T1 collection. The blue boxplots on the respective left depict the results for scenario a), the red boxplots on the respective right depict the results for scenario b).

obviously. Further investigations on which process requires how much execution time and eventual subsequent optimizations are subject of future work, though. As discussed on several occasions before, the $\text{UFS}_{256,cc}$ results stand out as it is the only technique where the rule 'greater number of indexed areas $\rightarrow$ longer ranking duration' does not apply: It has 146% indexed areas but only 94% mean ranking duration. This demonstrates that the assumed reduced computational efforts for the $\text{UFS}_{n,cc}$ ranking can indeed be realized.

Table 37: Relative and absolute mean ranking duration results for the techniques in scenario a) alongside their relative numbers of indexed areas.

technique $\rightarrow$ / key figure $\downarrow$	MBR	$\text{UFS}_{256,cc}$	KDMBR^{6}_{256}	$\text{KDQT}^{64,1.0E-5}_{32}$	$\text{QTMBR}^{6}_{512,0.05}$	$\text{MBRQT}^{16,1.0}$
number of indexed areas [in %]	100	146	136	417	232	115
relative mean ranking duration [in %]	100	94	109	185	149	103
mean ranking duration [in ms]	1,729	1,631	1,887	3,201	2,582	1,773

For scenario b), the results are a bit different. The mean ranking duration of $\text{UFS}_{256,cc}$ is now 237% of the MBR approach (see Table 38). This is not too much of a surprise since for each query, the Voronoi diagram has to be explicitly reconstructed from the sites. Nevertheless, for 'higher' parame-

terizations (and therefore greater numbers of indexed areas), the—in this regard—conceptual advantage of $\text{UFS}_{n,cc}$ (or *pure* global space partitioning approaches in general) should pay off, again: The number of indexed areas grows disproportionately in comparison to the number of global Voronoi cells if more storage space is provided for the resource descriptions. Hence, in such situations, the costs of the construction of the Voronoi diagram should amortize better. For the remaining techniques, the ranking duration differences are significantly greater compared to scenario a) which is due to the higher decoding costs, obviously. Notably, the quadtree reconstruction plays an important role in this issue: the greater the quadtrees, the more disproportionate the prolongation of the ranking duration. For example, $\text{KDQT}_{32}^{64,1.0E-5}$ has 417% indexed areas while $\text{QTMBR}_{512,0.05}^{6}$ has only 232% indexed areas. On the other hand, the quadtrees of $\text{QTMBR}_{512,0.05}^{6}$ are generally greater than the $\text{KDQT}_{32}^{64,1.0E-5}$ quadtrees since the former ones build the description base while the latter ones are used for the refinement of occupied cells. Consequently, the ranking duration of $\text{KDQT}_{32}^{64,1.0E-5}$ is 953% whereas the ranking duration of $\text{QTMBR}_{512,0.05}^{6}$ is at a comparatively disproportionate 837%. The relative high costs of the quadtree reconstruction also show when comparing the results for KDMBR_{256}^{6} (which does not involve quadtrees) and $\text{MBRQT}^{16,1.0}$: the latter unveils significantly longer ranking durations despite featuring about 254,000 fewer indexed areas. Accordingly, the durations for $\text{MBRQT}^{16,1.0}$ are significantly greater than for the MBR approach (also in comparison to the relative results for scenario a)).

Table 38: Relative and absolute mean ranking duration results for the techniques in scenario b) alongside their relative numbers of indexed areas.

technique → / key figure ↓	MBR	$\text{UFS}_{256,cc}$	KDMBR_{256}^{6}	$\text{KDQT}_{32}^{64,1.0E-5}$	$\text{QTMBR}_{512,0.05}^{6}$	$\text{MBRQT}^{16,1.0}$
number of indexed areas [in %]	100	146	136	417	232	115
relative mean ranking duration [in %]	100	237	205	953	837	262
mean ranking duration [in ms]	2,092	4,958	4,286	19,935	17,514	5,484

Therefore, the quadtree reconstruction would be the starting point for a profiling and a check for possible optimizations. The following is an example for such an optimization: In the current quadtree implementation of the distributed application scenario, the number of the quadtree's internal nodes is not stored as a class attribute. Consequently, to find out its number

[216] In our implementation, a quadtree's number of internal nodes is e.g. required during the decoding of CBLQ-coded $\text{QTMBR}_{c,a}^{b}$ summaries.

of internal nodes, the quadtree's tree structure has to be traversed—which is suboptimal in terms of runtime in any case.[216] Nevertheless, also for the other approaches, there is certainly room for optimizations to significantly reduce the ranking duration runtimes. For example, for all approaches, the squared distances could be calculated instead of the actual distances which require to extract the square root. Furthermore, our implementation is single-threaded. The MINDISTs to the query point and the surface areas of the indexed areas (which are required to build the resources' `REntries`) could be calculated in parallel. This should greatly reduce the overall duration. All in all, we think the implementations are generally at a similar level of optimization for the different approaches. Therefore, the measurements should represent a fair depiction with regard to the computational efforts of the respective ranking costs since the results also reflect one's intuition quite well.

8.9. Summarization of the Results for the Distributed Application Scenario

In this section, we present a final overview of the results for the distributed application scenario. In particular, we summarize the key results of the entire evaluation (section 8.9.1), assess the results with regard to the degree of achievement of thesis objective ❶ (section 8.9.2), and give an overview of topics for future work (section 8.9.3).

8.9.1. KEY RESULTS OF THE EVALUATION. The evaluations show that the utilization of full-precision bounding volumes has to be performed with caution for the assumed distributed application scenario. It seems that at best, the utilization of one full-precision MBR *in combination with other approaches* is viable (such as for MBRQT^{c,a}). In terms of indexing multiple areas, multiple full-precision bounding volumes generally take up too much storage space compared to approaches utilizing space partitioning and/or quantization methods. Likewise, non-adapted space partitioning is unsuitable (even for hybrid approaches such as GridQT$_r^{c,a}$) in case of a non-uniform spatial distribution of the data points (which is usually the case). The pure global space partitioning approaches are de facto unsuitable for application in scenarios featuring a lot of resources (as simulated with the `T1` collection) since the amount of resources administering data points in the single cells is much too great. In such situations, even the MBR approach is vastly superior when spending similar amounts of storage space. In application scenarios featuring less resources (as simulated with the `F` collection), pure global space partitioning approaches are doing better but are still far behind the best hybrid approaches—as long as the allocation of data points to resources follows the typical long tail distribution. Nevertheless, the evaluation also reveals that the pure global space partitioning approaches are unconquerable in a specific situation: when the network consists of rather few, invariably big, and spatially very spread resources (as simulated with the `T2` collection). In such situations, the hybrid approaches show massive problems to keep up since a lot of areas need to be indexed for each resource—which is comparatively storage-space-intensive for them. Furthermore, the accuracy gained by means of the refinement is oftentimes fairly negligible since the data points are spread so widely within the subspace to refine. In contrast, for the pure global space partitioning approaches, only *a single bit* is required for each area. Of course, all these explanations assume that the global space partitioning is adapted to the spatial distribution of the data points.

The hybrid approaches prove to be best suited for the situation that fits the assumed distributed application scenario most (simulated by the `T1` and `F` collections)—the long tail distribution of data points to resources. KDQT$_n^{c,a}$ and QTMBR$_{c,a}^b$ stand out as particularly good. For hybrid approaches, the utilization of quantized MARs as opposed to a single quantized MBR generally does not pay off—at least for the way in which we have

applied them: Both $\text{KDMAR}_n^{b,k}$ as well as $\text{QTMAR}_{c,a}^{b,k}$ are no improvements compared to their MBR-utilizing counterparts. In fact, for the most part, $\text{KDMAR}_n^{b,k}$ respectively $\text{QTMAR}_{c,a}^{b,k}$ are even slightly inferior despite much greater computational costs. Although all in all, the quantized MBRs are the better choice compared to quantized MARs, the former are too coarse of a refinement in some cases: If the data points are unfavorably distributed within their cell, the quantized MBRs cover too large portions of the cell. An adapted approach—utilizing both quantized MBRs and quantized MARs—which could provide remedy is briefly discussed in section 8.9.3. In contrast, the quadtree refinement of occupied cells proves to be a very effective means which is unrivaled in reducing indexed dead space *efficiently*. Overall, the evaluation gives the impression—since all in all, e.g. $\text{KDQT}_n^{c,a}$ is more promising than KDMBR_n^b—that the quadtree is generally an even more suitable means for refinement than the quantized MBR. At least, it is significantly more robust.

Another interesting result is that the area-related key figures (overlap between summaries, data space coverage, and surface area per summary) for the specific techniques are not linearly transferable into their resulting selectivity. It has been shown that it is important to feature very accurate summaries in regions which—on a global scale—are very densely populated with data points while in sparsely populated regions, also coarser summaries might be acceptable. In fact, it is this particular property which makes (adapted) global space partitioning interesting as summarization tool for non-uniformly distributed data. From the surfaces of the indexed areas alone, global space partitioning is clearly inferior to local space partitioning and even data partitioning approaches.

For large amounts of storage space spent on average (i.e. greater than 'MBRsize+'), all approaches show converging, very selective *rfc* values for the data collections featuring a long tail distribution. That means that for all approaches (aside from the MBR approach), great selectivity can be achieved. This is not too much of a surprise since for high parameterizations, the summaries are becoming spatially very accurate and also the share of directly represented resources becomes very large. In consequence, in case the minimization of the total amount of storage space required has only a low priority, the approach to apply could be selected according to which of them allows for the fastest query processing.[217]

Generally, pure global space partitioning approaches oftentimes behave very different from hybrid approaches and pure data partitioning approaches. This is because the indexable areas are 'shared' between the

[217] Note that in section 8.8.2, only the duration of the initial resource ranking is evaluated. The query processing also involves the subsequent application of the kNN algorithm. Therefore, even though $\text{UFS}_{n,cc}$ (or pure global space partitioning approaches in general) appears to be promising on the basis of the initial ranking's duration, a separate evaluation including the execution time of the kNN algorithm is required in order to make an accurate statement regarding this issue.

resources for the pure global space partitioning approaches. In contrast, for approaches of the other categories, the indexed areas are much more adapted to the individual resource—even for the hybrid approaches *based* on global space partitioning.

In general, it has been shown that the suitability of the different approaches with respect to *specific queries* can vary widely even if the overall result is very similar: A lot depends on the data space region a query falls into. Usually, this involves approaches that differ in their conceptual methodology (such as approaches with a global space partitioning base as opposed to approaches with a local space partitioning base). Obviously, this is in particular the case if rather coarse summaries are utilized—as has been demonstrated in section 8.1.2 of this work. There, the relative selectivity of KDMBR_n^b, $\mathrm{KDQT}_n^{c,a}$, and $\mathrm{QTMBR}_{c,a}^b$ varies greatly for single queries even though overall, almost the same *rfc* values result. Consequently, for 'real-world' applications, one has to be very aware of the circumstances in which what approach is well-suited, and where the majority of the query points is expected to be located at. It also points out that there is more than one way to do it and that there is no specific approach which is the best solution in all circumstances. Generally, if the query point is located in a data space region with a high global data point density, hybrid approaches based on global space partitioning are best. For regions of low to middle global data point density, hybrid approaches based on local space partitioning are superior. Overall, it also depends on whether the cell boundaries of the global space partition are favorably positioned for the specific query point: In case the boundaries are favorably positioned, global space partitioning can be extremely promising with regard to the spatial accuracy of the areas which are indexed by the resource summaries (see e.g. Figure 61 on page 219). Otherwise, it can also turn out the other way round (see e.g. Figure 64 on page 222).

Furthermore, it has been shown that the option of a direct representation of resources, on the one hand, can significantly reduce the storage space requirements for 'high' parameterizations but, on the other hand, has no significant impact on the *relative performances* between the different approaches. As sole exception, this does not apply to pure global space partitioning approaches since the omission of the direct representation can lead to sharp drops in selectivity—depending on the general conditions: In case a large share of resources is directly represented for a network of many resources, large selectivity losses are to be expected when the direct representation is omitted or not permitted.

Alternative selection strategies for query points (which simulate different assumptions regarding the origin of queries) have also no significant impact on the relative performances between the approaches. This is not much of a surprise as the ranking algorithm is the same for all approaches (and therefore, big resources are generally and likewise preferred in all rankings). In this context, it has been shown that pure global space par-

titioning approaches hold a slightly stronger preference for big resources. Hence, in case the query points originate mostly from big resources, the selectivity of global space partitioning approaches increases a little more compared to the others. Nevertheless, all approaches increase their selectivity in such cases (as typically, the number of relevant resources is then also significantly lower).
Finally, it has been shown that for the ranking of resources, using the available coordinate information of the spatial domain is superior to using ranking schemes that are also suited for metric spaces (i.e. which are not using coordinate information). For pruning purposes, the superiority of utilizing the spatial information is self-evident and therefore, it has not been evaluated separately.

8.9.2. DEGREE OF THESIS OBJECTIVE ACHIEVEMENT AND CONTRIBUTIONS. In the following, we briefly assess the degree of achievement with regard to thesis objective ❶. Generally, it can be stated that the aim of significantly improving the MBR approach has been fully accomplished. At 'MBRsize', the newly developed approaches are vastly superior with regard to the resource selection performance in all evaluations. In total, a wide range of applicable concepts has been presented to improve the MBR. Especially $\text{QTMBR}^{b}_{c,a}$ and $\text{KDQT}^{c,a}_{n}$ turn out to be significant improvements—each by applying different concepts (local vs. global space partitioning, quantized MBRs vs. cell-interior quadtrees). It has also been shown that superior resource selection performance can be achieved even if significantly less storage space is utilized on average. Therefore, the aim of improving resource selection performance *and* resource description efficiency simultaneously has been fully accomplished, too.
With regard to the 2012 state-of-the-art approach $\text{UFS}_{n,cc}$, it has been shown that it is unsuited for application in case a great number of resources exist (see evaluations of the `T1` collection). But even in the environment which was previously dominated by $\text{UFS}_{n,cc}$—the `F` collection with its significantly smaller set of resources and the long tail distribution of data points to resources—the novel approaches greatly outperform $\text{UFS}_{n,cc}$ for almost the whole *rds* range. Hence, also the aim of significantly improving the resource description approaches developed until 2012 has been accomplished. The only environment in which $\text{UFS}_{n,cc}$ is still top-notch is the pathological case of the `T2` collection.
In accordance with the improvement of the MBR approach and $\text{UFS}_{n,cc}$, the development of approaches that enable a high spatial accuracy for the approximated representation of two-dimensional data point sets as well as their suitable, storage-space-efficient representation has also been thoroughly successful.
With respect to improving further properties of existing approaches (*besides* resource selection performance and resource description efficiency), with $\text{QTMBR}^{b}_{c,a}$ and $\text{MBRQT}^{c,a}$, we have succeeded to develop capable approaches which are neither dependent on additionally transmitted infor-

mation nor on extensive compression. Both are drawbacks of approaches relying on global space partitioning. In cooperative environments, the former might not be too much of a problem but for uncooperative environments, it can be much more of a hindrance. Furthermore, in either environment, additional communication costs are introduced by the need to collect data in the first place and then disseminate the information about the global space partition. Completely self-dependent approaches do not require any of this: For $\text{QTMBR}^{b}_{c,a}$ and $\text{MBRQT}^{c,a}$, all information on the indexed areas is encoded in the summaries and due to the high entropy of the captured data, the resource summaries are mostly not compressible. Hence, it can be refrained from applying compression. Both aspects make $\text{QTMBR}^{b}_{c,a}$ and $\text{MBRQT}^{c,a}$ easier to use than the approaches relying on global space partitioning. On top of it all, at least $\text{QTMBR}^{b}_{c,a}$ also shows a lower tendency to directly represent resources, i.e. it is capable of describing small or spatially narrow resources with very low storage space requirements.

Furthermore, it has been demonstrated that the indexed areas' surfaces are dramatically reduced by the novel approaches. Again, especially $\text{QTMBR}^{b}_{c,a}$ achieves tremendous reductions compared to the MBR approach. Due to its global space partitioning nature, $\text{UFS}_{n,cc}$ is unsuited for the aim of minimizing the indexed surface area—which might be important in certain applications. The novel approaches' reduction of the indexed areas' surfaces is even significantly greater than the improvements in resource selection performance (both in comparison to the MBR approach). This is an essential finding with regard to the aspect of conducting basic research for summarizations of two-dimensional spatial point data sets with this work. The very extensive evaluation—which elaborates the strengths and weaknesses of the approaches under different conditions in detail—can also be seen as a contribution to this basic research.

When indexing spatial data, an established method to reduce indexed dead space is to further refine occupied cells of a space partition with a quantized MBR. By exploiting the presence of their embedding cell in combination with the quantization, the storage space requirements of these MBRs can be kept low whilst still offering reasonable spatial accuracy in comparison to full-precision MBRs. Nevertheless, as noted at several occasions throughout this thesis, quantized MBRs potentially suffer from covering large parts of the cell they are quantized into which severely impedes the aim of reducing dead space. In this thesis, two methods have been developed to conquer this drawback while still remaining storage-space-efficient: the utilization of quantized MARs (including a rectangle number reduction procedure for minimizing the storage space requirements), and the utilization of cell-interior quadtrees. With regard to the reduction of dead space, the in-depth evaluation shows that the use of cell-interior quadtrees generally leads to a much greater reduction of the indexed areas' surfaces. Comparisons of the corresponding approaches KDMBR^{b}_{n} and

KDQT$_n^{c,a}$ tend to be won by the latter, especially at 'MBRsize'. Therefore, we think that the cell-interior quadtrees are a very valuable tool for refinement purposes, especially since the calculation is also rather cheap. With regard to the use of quantized MARs, despite the rectangle number reduction procedure, no real gain is achieved compared to the approaches that only use quantized MBRs. This is all the more disillusioning as both the recursive MAR calculation as well as the rectangle number reduction procedure are quite runtime-intensive. Clarification on the extent to which the quantized MARs succeed in reducing dead space compared to quantized MBRs is still to be investigated. However, with careful use, we generally assume that the method can be brought in profitably for the distributed application scenario (a brief explanation follows in section 8.9.3). For more specific applications, it is conceivable that the utilization of quantized MARs (including the rectangle number reduction procedure) might also be beneficial as a standalone method—although the question arises whether the quadtree refinement approach might not be the most preferable option in any situation. Overall, we consider the MAR method and the rectangle number reduction procedure to be interesting procedures which do not pay off for the distributed application scenario in their current form of use by a narrow margin.

A further contribution of this work is the identification and the application of two suitable methods which have not yet played a major role in the research concerning the indexing of spatial vector data: the utilization of linear quadtree encodings and the rectilinear polygon decomposition. While the application of the former has already been extensively studied in this work, the latter has only been applied in the context of the rectangle number reduction procedure. By the creation of the link between the fields of spatial indexing and polygon decomposition, additional suitable application scenarios and approaches may be discovered in the future.

As a final aspect, we want to mention the utilization of the Skyline operator as a means for a meaningful comparison of the different approaches' performances. Prior to the introduction of the Skyline operator, these comparisons were rather complicated and also non-intuitive as it is difficult to generally determine a valuation of the trade-off between storage space efficiency and selectivity—which is inherent to the distributed application scenario, however—on the basis of the results for specific *techniques*. Even though it has been shown that also the assessment via Skyline operator has some shortcomings (e.g. when the Skylines' anchor points are too far apart), we think that its application is a big step forward with regard to the evaluation of the distributed application scenario's results.

All in all, taking the achieved effectiveness, the achieved efficiency, the minimization of the indexed areas' surfaces respectively the indexed dead space, the independence from additional information and compression, the not too large number of parameters and their relative ease of setting, and

the limited computational complexity into account, $\text{QTMBR}^{b}_{c,a}$ appears to be the most promising approach overall.

8.9.3. STARTING POINTS FOR FUTURE WORK. Finally, we briefly outline some aspects for future research. As already stated, the utilization of quantized MARs instead of a single quantized MBR for refinement did not result in improvements, it was rather even the opposite of the case. A more detailed investigation of the reasons behind this is an interesting starting point for future work. A conceivable approach integrating both methods could be to use the MAR refinement only if the quantized MBR reduces the indexed surface area of an occupied cell just marginally. For example, "if the quantized MBR does not reduce the indexed surface area of its cell by at least p%, then utilize quantized MARs for the refinement". Similar to $\text{RecMAR}_{k,sl}$, a recursive procedure could be applied until the reduction of the dead space is satisfactorily. Of course, p would be an additional parameter which requires adjustment and thus reduces the ease of use. With regard to the introduction of such a parameter p, $\text{QTMBR}^{b}_{c,a}$ would also provide another option for remedy: The basic quadtree cell in question could simply be further partitioned by the basic quadtree space decomposition (and then again, each occupied cell is refined by a quantized MBR). In case the indexed area can be significantly reduced, the more detailed description replaces the former one. In general, the aforementioned approach integrating MBRs and MARs would be a more complex solution as also a novel structure for the summaries' bit vectors would have to be designed.
Since neither $\text{KDMAR}^{b,k}_{n}$ nor $\text{QTMAR}^{b,k}_{c,a}$ have been selected for consideration in the in-depth evaluation, no evaluation of the rectangle number reduction procedure's impact has taken place. Therefore, we take a quick look at it here. Table 39 showcases some key figures recorded for selected techniques. For both $\text{KDMAR}^{b,k}_{n}$ and $\text{QTMAR}^{b,k}_{c,a}$, we select the techniques at 'MBRsize' and the respective highest parameterizations. The data collection is the `T1` collection.
Obviously, it is impossible to reduce the number of refining rectangles if there is only one rectangle within an occupied subspace. Therefore, the amount of occupied subspaces for which a reduction is possible is depicted in the '# of possible reductions'-row (row 5). Note that in case a quantized MAR is completely overlapped by another quantized MAR, the overlapped MAR is removed *before* the rectangle number reduction procedure starts. Therefore, the value specified in row 5 depicts the number of occupied subspaces containing two (or more) MARs which are not completely congruent or overlapped. Consequently, for these occupied subspaces, an actual reduction can be obtained. The results show that the greater parameter k (i.e. the more MARs are maximally calculated for an occupied subspace) and the lower parameter b (i.e. the coarser the quantization), the more often a rectangle number reduction is achieved. For $\text{KDMAR}^{4,2}_{256}$, $\text{KDMAR}^{6,4}_{8192}$,

Table 39: Key figures depicting the impact of the rectangle number reduction procedure for $\text{KDMAR}_n^{b,k}$ and $\text{QTMAR}_{c,a}^{b,k}$ (T1 collection).

key figure ↓ technique →	$\text{KDMAR}_{256}^{4,2}$	$\text{KDMAR}_{8192}^{6,4}$	$\text{QTMAR}_{256,0.1}^{6,2}$	$\text{QTMAR}_{1024,0.001}^{6,4}$
# summary-represented resources	1,320,527	799,684	1,482,510	1,353,109
# occupied subspaces	1,654,160	1,902,101	2,428,840	3,570,059
# occupied subspaces with 1 MAR	1,168,819	974,671	1,242,130	2,107,759
# of possible reductions	485,341	927,430	1,186,710	1,462,300
no reduction	328,668 67.7%	669,126 72.1%	1,071,199 90.3%	982,823 67.2%
reduction	154,898 31,9%	255,959 27.6%	112,541 9.5%	476,724 32.6%
increase	1,775 0.37%	2,345 0.25%	2,970 0.25%	2,753 0.19%
# indexed areas before reduction	2,139,501	3,600,144	3,615,550	5,932,066
# indexed areas after reduction	1,984,603 -7.2%	3,328,901 -7.5%	3,503,009 -3.1%	5,428,900 -8.5%

and $\text{QTMAR}_{1024,0.001}^{6,4}$, reductions are achieved for about 30% of the occupied subspaces for which a reduction is possible. For a fine quantization in combination with a lower maximum number of MARs, the reduction succeeds relatively rarely—as can be seen for $\text{QTMAR}_{256,0.1}^{6,2}$ (only in 9.5% of the possible cases). In rare cases, the rectangle number reduction procedure even results in more rectangles (see 'increase'-row). This can happen if its input is a polygon cover which requires less rectangles than the optimal polygon decomposition. A simple example is depicted in Figure 85. As noted in the '$\text{KDMAR}_n^{b,k}$'-paragraph[218] of section 4.3, in these cases, the input is used as the occupied subspace's refinement. In case the input and the decomposition solution have the same amount of rectangles, the decomposition solution is used.[219] In general, in rather favorable circumstances (see $\text{KDMAR}_{256}^{4,2}$, $\text{KDMAR}_{8192}^{6,4}$, and $\text{QTMAR}_{1024,0.001}^{6,4}$), a reduction of about 7% to 9% can be achieved for the total amount of indexed areas while in rather unfavorable circumstances (see $\text{QTMAR}_{256,0.1}^{6,2}$), a reduction of about 3% is obtainable in our preliminary experiments. For even more favorable circumstances (i.e. a high maximum amount of quantized MARs

[218] This paragraph begins on page 84.

[219] This is done because the decomposition solution's surface area is guaranteed to be smaller or at most equally large as the input's surface area (due to the former's guaranteed absence of overlap).

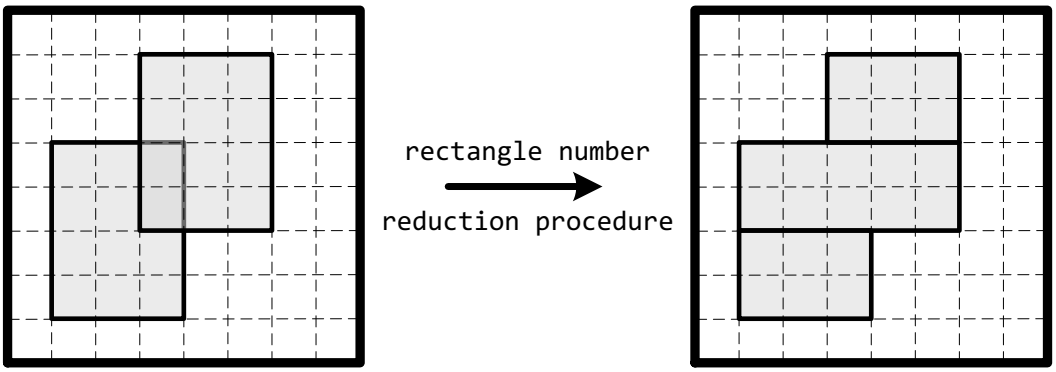


Fig. 85: Exemplary depiction of a rectangle number reduction procedure which results in an increase of the number of rectangles.

and a coarse quantization, e.g. using only $b = 3$ bits), we expect the total share of successful reductions to be significantly greater than 10%. The extent to which a) the solution found then does not mostly simply correspond to a quantized MBR and b) an improvement with regard to the spatial accuracy of a quantized MBR can be achieved has to be determined in future work. However, we think that the rectangle number reduction procedure can be valuable in appropriate circumstances (such as maybe for the aforementioned combined approach with quantized MBRs and MARs) although no measurable advantage resulted for the distributed application scenario and its use within the $\text{KDMAR}_n^{b,k}$ respectively $\text{QTMAR}_{256,0.1}^{6,2}$ approaches.
The utilization of the covering radii information did not turn out favorably for $\text{DFS}_{n,cc}^b$. A more suitable approach for refining a basic Voronoi space partition could be to calculate the MBR for each Voronoi cell's polygon and then utilize this MBR as potential data region for the calculation of quantized MBRs as refinement.[220] The structure of the corresponding summaries' bit vectors could be exactly as for KDMBR_n^b. Since Voronoi cells can be of rather unfavorable forms with regard to being approximated by an MBR (see e.g. Figure 77 on page 242), here, the use of quantized MBRs for refinement purposes may not be quite as promising as for e.g. KDMBR_n^b. However, it should certainly lead to better results than $\text{DFS}_{n,cc}^b$.
Space-filling curves have not been applied in this work since its foundation, the uniform grid, has been shown to be unsuitable in the distributed application scenario in prior work (see for example [Henrich and Blank 2010]). Nevertheless, they could be utilized in the context of global space partitioning. For both the k-d-based as well as the Voronoi-based approaches, the assignment of IDs to cells (and therefore the indices of their corresponding binary occupancy information in the summaries' bit vectors) is more or

[220] Of course, also the utilization of MBR-interior quadtrees, quantized MARs, or any combinatory approach using both quantized MBRs and MARs is conceivable. For convenience, we limit ourselves to quantized MBRs in the following notes, though.

less random.[221] In general, it is likely that resources administer their data points in adjacent cells of the space partition. Hence, if the centers of the k-d-cells respectively the sites are ordered by means of a space-filling curve, longer zero runs (respectively one runs) might arise for the summaries, increasing compressibility. In the 'Space-Filling Curve'-paragraph[222] of section 2.4.3, a set of conceivable space-filling curves has been presented. The examination of which space-filling curve is most suitable for which approach is also a point for future work.

An idea for a new category of summarization approaches within the context of global space partitioning is to use two levels of global space partitions. The first level is a rather coarse basic space partition while the second level is a very fine space partition which further decomposes the cells of the first level's space partition. In case a cell of the first level's space partition is occupied, the summary additionally contains occupancy information for the corresponding finer cells of the second level's space partition (the bit vectors can be designed similar to those of hybrid approaches based on a global space partition). For the definition of the second level's space partition, different approaches are conceivable. It can e.g. be the simple continuation of the space partition of the first level or, specifically with regard to Voronoi diagrams, the utilization of e.g. hierarchical Voronoi diagrams (see section 2.4.2).

Another starting point for future work is the evaluation of additional query types such as range queries or more complex query types such as reverse kNN queries. In comparison to the MBR and UFS$_{n,cc}$ approaches, especially for range queries, massive selectivity improvements should be achievable.

A further interesting field for future work is the design of alternative ranking algorithms which allow for a faster initial ranking and thus a sped-up query processing. The time measurements for the precise kNN queries in the current implementation (see section 8.8.2) showed that depending on the technique, between 2 and 20 seconds are required for the initial ranking of a query if no information is cached. Although in the given context, these runtimes do not appear to be unacceptable and furthermore, several starting points for optimizations have already been identified, there are certainly application scenarios for which these runtimes are too long. Hereby, the use of appropriate resource indexes could provide remedy.[223] Adapted ranking algorithms which speed up the ranking procedure at the cost of giving up on the preciseness of the result (i.e. the results are only approximately correct) could also be of interest for such applications where the results have to be determined very quickly.

With regard to enhancing the evaluation, the development of a cost model taking the separately transmitted information about basic global space

[221] This also applies to the Voronoi-based approaches even though the sites are sorted by 1) their *long*-coordinate and 2) their *lat*-coordinate (see footnote[211] on page 253).

[222] This paragraph begins on page 39.

[223] Note that this has already been briefly discussed in footnote[9] on page 10.

partitions (for resource description approaches being based on global space partitioning) into account is of interest. This cost model could either aim for a consideration with regard to the communication costs in the network, or for an adapted calculation of the resulting *rds* values. In the evaluations in this work, this separately transmitted information has been ignored so far.

Part III:

Centralized Application Scenario

As a second proof of concept, our summarization approaches are applied in a multidimensional data structure—the R-tree. Consequently, this point marks the beginning of Part III of the thesis: the centralized application scenario. It is organized as follows: In section 9, we briefly introduce the idea behind the utilization of summarization approaches in the classical R-tree and also explain why the R-tree is a suitable target for doing so. In section 10, the R-tree and important aspects of the integration of our summarization approaches into an R-tree are outlined. It is followed by a presentation of algorithms for range and kNN queries in section 11. After showcasing the experimental setup (section 12), the evaluation of the selected approaches is conducted (section 13) which concludes Part III.

9. PRELIMINARIES AND MOTIVATION FOR THE CENTRALIZED APPLICATION SCENARIO

For the centralized application scenario, we assume that a set of geotagged media items is administered in an R-tree. An R-tree is a centralized multidimensional data structure. It means that the media items are administered on a single machine. This is in contrast to the distributed application scenario in which the media items are administered by a multitude of machines which build a network. Consequently, in the centralized application scenario, the 'resources' do not correspond to peers in a P2P network but to the internal and leaf nodes of the R-tree's hierarchical tree structure. All this has been outlined in section 1.2.2. We assume a primary index R-tree, i.e. the geotagged media items are administered directly in the leaf nodes of the tree structure. Similar to the distributed application scenario, the spatial footprints of the media items are Plate-Carrée-projected geocoordinates (i.e. two-dimensional data points for which x corresponds to *long* and y corresponds to *lat*). Nevertheless, the International Date Line is *not* taken into account for the centralized application scenario, i.e. also in the x-dimension, the data space boundaries are now closed. This is because during the evaluation of the distributed application scenario, we noticed that the consideration of the International Date Line has at most marginal effects on the results but considerably complicates the implementation. Similar to the distributed application scenario, we assume a rather static database, i.e. the integration of our approaches is tailored for conducting many queries on a relatively unchanged data point set.

The R-tree has been briefly discussed in section 2.4.3 and is outlined in more detail in section 10.1. The main property which makes it suitable for the integration of our summarization approaches is its data partitioning feature: In general, in case a node must be split, the node's set of data objects is divided into two groups by only considering the data objects themselves. It is independent of previously made decisions on the path from the root to the node. Consequently, also the node's spatial footprint is only dependent on its associated data objects. This simplicity provides an ideal framework for integrating our sophisticated summaries—which are made to depict a given set of data objects (respectively a set of data points, to be more specific).

As mentioned in conjunction with thesis objective ❷, the field of multidimensional data structures has been intensively researched over a period of more than 30 years. Besides a plethora of other optimizations, alternatives for the classical summary describing the nodes' spatial footprints—the MBR—have also been proposed. The suggestions range from alternative full-precision bounding volumes such as spheres (see e.g. the sphere tree) or polygons (see e.g. the cell tree), to hybrid combinations of full-precision bounding volumes such as MBRs and spheres (see e.g. the SR-tree), or the utilization of quantized MBRs (see e.g. the QRMBR technique). This has

been showcased in section 2.4.3. Nevertheless, to the best of our knowledge, there has been no research with respect to the strategy that makes our summarization approaches so successful in the distributed application scenario: divide the data point set to describe into groups, and describe each group tightly and storage-space-efficiently. Hence, the integration is an attempt to evaluate if this neglected strategy can still lead to improvements in this intensively optimized domain. As a consequence, we want to keep the R-tree that integrates our summaries as close to the classical R-tree as possible.

Obviously, only 'self-organizing' summarization approaches are suitable for integration, i.e. approaches which solely require the data point set to summarize as input. Hence, the approaches based on global space partitioning are not an option as they leave too many questions unanswered: Where does the global space partition come from? How and when is the space partition updated? Where is the information on the space partition stored in the R-tree? And so on. Thus, from our 13 approaches besides the MBR approach, only RecMAR$_{k,sl}$, QT$_{c,a}$, GridQT$_{r}^{c,a}$, QTMBR$_{c,a}^{b}$, QTMAR$_{c,a}^{b,k}$, MBRQT^{c,a}, and MARQT$_{k,sl}^{c,a}$ remain. In the evaluation of section 7, it has been shown that for hybrid approaches, the utilization of quantized MARs tends to lead to slightly worse results in comparison to the usage of quantized MBRs—despite much greater computational efforts. Similar results can be expected for the centralized application scenario which on top requires far greater computational efficiency than the distributed application scenario. Hence, QTMAR$_{c,a}^{b,k}$ and MARQT$_{k,sl}^{c,a}$ are excluded. In principle, the RecMAR$_{k,sl}$ approach corresponds to the application of the exhaustive node split algorithm of Guttman. Nevertheless, it has to be applied for *each summary (re-)calculation*—which is much more frequent than a node split. Due to its high computational costs (even with the polynomial time algorithm), an application of RecMAR$_{k,sl}$ is therefore unsuitable. For the distributed application scenario, the results for GridQT$_{r}^{c,a}$ were very similar to those of QT$_{c,a}$ but mostly tended to be worse. This can also be expected here. QT$_{c,a}$ for its part has consistently been outperformed by QTMBR$_{c,a}^{b}$ for the `T1` collection. Since QTMBR$_{c,a}^{b}$ has also shown to be the most promising approach for the distributed application scenario overall, it is an obvious candidate for use in the centralized application scenario. MBRQT^{c,a} did not perform quite as well as QTMBR$_{c,a}^{b}$ (except for the pathological case of the `T2` collection) but is a very natural extension of an MBR. Therefore, we integrate both QTMBR$_{c,a}^{b}$ and MBRQT^{c,a} into an R-tree. Both summarization approaches also fulfill the general requirements for summaries in R-trees (see section 10.3).

For our R-tree, we use a Java/Kotlin main memory implementation which simulates a disk-resident R-tree implemented in a lower-level programming language such as C++. Kotlin[224] is a statically typed programming

[224] https://kotlinlang.org/, last visit: 08.08.2018.

language that runs on the Java Virtual Machine (JVM) and is 100% compatible with Java. In our implementation, the general R-tree features are implemented in Kotlin whereas the summary-calculation-related parts are implemented in Java.[225] Overall, this makes no difference in terms of performance as both types of source code are compiled to Java byte code.

[225]The Kotlin part of the project has been implemented by Felix Engl in the course of two student projects and a master's thesis. For the master's thesis, see [Engl 2018].

10. INTEGRATING SUMMARIES INTO AN R-TREE

In this section, the aspects which are important for the integration of our summarization approaches into an R-tree are outlined. The classical R-tree and some of its most recognized optimizations are described in section 10.1. As mentioned before, we want to keep the classical R-tree as unchanged as possible to assess the potential for improvement by integrating our MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summaries. In more than 30 years of research, a lot of R-tree optimizations have been tailored considering MBRs as the nodes' summaries. Therefore, it is not adequate to apply many of these optimizations to our R-tree as we want to assess the potential of improved spatial footprints. The applied optimizations are outlined at the appropriate places in the following sections. Nevertheless, it does not make sense to e.g. employ evidentially suboptimal split algorithms. Furthermore, the utilization of our summarization approaches requires adjustments to the R-tree. The necessary modifications are showcased in section 10.2. In section 10.3, the general requirements for summaries in R-trees are outlined. In addition to the modification of the R-tree itself, an efficient application of our summarization approaches necessitates a calculation of 'upper-level' summaries from 'lower-level' summaries. Approaches for calculating MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summaries from corresponding 'lower-level' summaries are presented in section 10.4. There, also the storage structures for writing the MBRQT^{c,a} respectively QTMBR$^{b}_{c,a}$ summaries to disk are showcased. Finally, the expected trade-offs with regard to the integration of more sophisticated summaries than MBRs into an R-tree are discussed in section 10.5.

10.1. The Classical R-tree

The classical R-tree was introduced in 1984 by Antonin Guttman and (alongside its many variations) has established itself as the predominant (family of) multidimensional data structure(s). In its classical form, the R-tree strongly resembles the B$^+$-tree [Comer 1979] and is a quite simple structure [Manolopoulos et al. 2006, p. 7]: Generally, it is a height-balanced tree which organizes a set of d-dimensional spatial objects. The spatial objects themselves are stored in the leaf nodes. Internal nodes consist of entries pointing to the child nodes and an associated d-dimensional MBR for each entry, spatially bounding the data contained in the subtree below. Besides the spatial objects, the leaf nodes contain pointers to the associated data objects.[226] Therefore, the classical R-tree logically corresponds to a nested hierarchy of MBRs. From our point of view, the MBRs are the *summaries* of the *resources* (i.e. the nodes of the tree structure) and—similar to keys in a classical one-dimensional data structure such as the B-tree—serve as 'signposts' when traversing paths in the R-tree. See Figure 86 for a

[226] Hence, the classical R-tree is designed as a secondary index.

visualization of the classical R-tree assuming two-dimensional rectangles as spatial data objects.

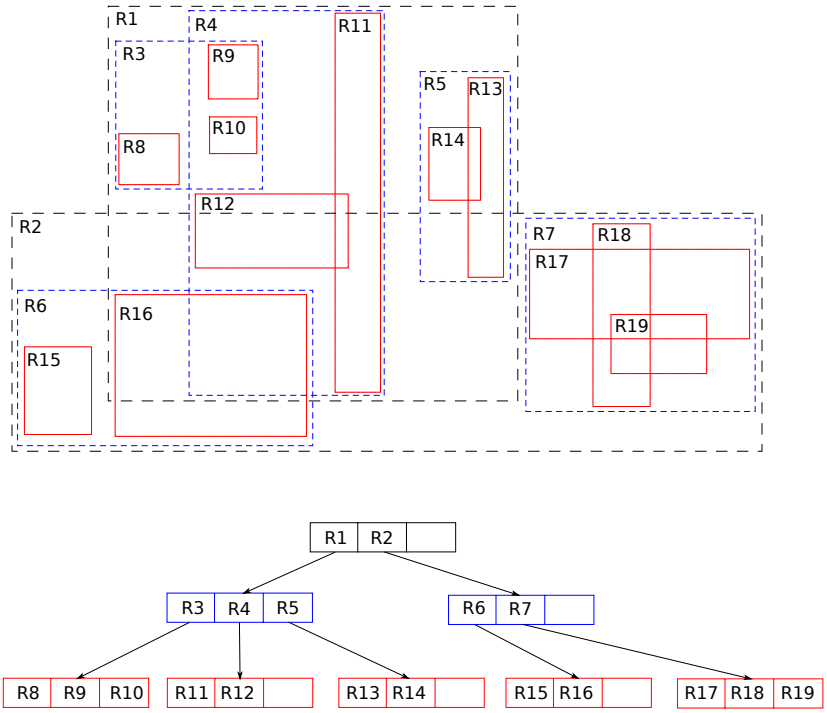


Fig. 86: 'Classical' R-tree (bottom) alongside its corresponding spatial representation (top). In this depiction, it is assumed that the spatial objects are two-dimensional rectangles. The root node is at the 'uppermost level' (0), the leaf nodes are at the 'lowest level' (2).[227]Image taken from [de Konink and Baca 2010].

Physically, each node is implemented as a disk page (i.e. the classical R-tree is disk-resident). Hence, a fixed, limited amount of storage space *bpp* (bytes per page) is available for each node of an R-tree. The classical MBR summaries occupy a fixed amount of storage space which depends on the dimensionality and the precision of the data. Generally, by use of MBRs as summaries, the maximum node capacity of an internal node is $M = (bpp - metadata\text{-}size)/(summary\text{-}size + pointer\text{-}address\text{-}size)$ entries. The metadata-size usually accounts for 16 B whereas the pointer-address-size is 8 B on a 64-bit system. For leaf nodes, M arises from the size of the node's metadata block and the indexed data objects (spatial footprints plus the pointers to the data objects). Considering data points (*dp*) as the spatial footprints of the data objects, the maximum capacity M of the leaf nodes can be determined as $M = (bpp - metadata\text{-}size)/(dp\text{-}size +$

[227]Note that the terms 'upper level' and 'lower level' refer to the tree structure as it manifests itself in a logical visualization, i.e. the root is at the uppermost level (or top) of the tree structure, and the leaf nodes are at the lowest level (or bottom). Index-wise, the tree level indices increase from the root towards the leaf nodes, i.e. the root has level index 0 whereas the leaf nodes have the greatest level index (2 in this example).

pointer-address-size).[228] See Figure 87 for a depiction of the disk pages' corresponding structures.

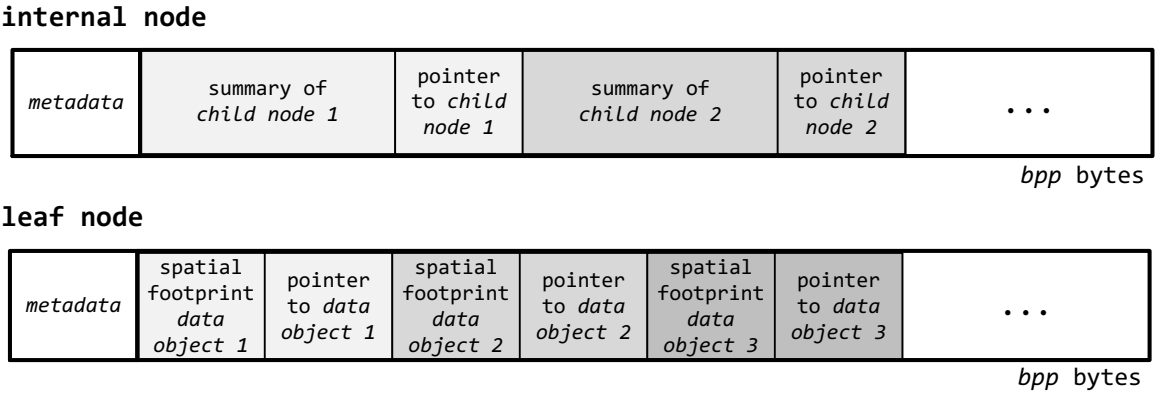


Fig. 87: Depiction of the disk pages' generic structure for internal and leaf nodes. In our specific case, the spatial footprints of the data objects in the leaf nodes are data points.

With M as the maximum number of entries in a node, $m \leq \frac{M}{2}$ as the minimum number, and considering MBRs as summaries, we can define that a 'traditional' R-tree has the following characteristics:

(1) Except it is the root, each leaf node hosts between m and M entries. The entries' spatial extents are represented by a bounding MBR.
(2) Internal nodes hold between m and M child nodes. They spatially bound their children's MBRs with their own MBR.
(3) The root as a leaf node has between 0 and M entries. As an internal node, the root holds between 2 and M entries.
(4) All leaf nodes of the R-tree are at the same level.

Let *data* be the data object[229] to be inserted into an R-tree, *sf* the spatial footprint of *data*, *selectedNode* the currently selected node in the traversal of the R-tree, and *query* the query object specifying the query type and its necessary parameters. In terms of minimum functionality, an R-tree requires two methods: INSERT(*data*, *sf*) and SEARCH(*query*).
For the insertion of *data*, an 'appropriate' leaf node to store *data* has to be determined. In particular, the INSERT(.)-method requires two procedures in which the critical decisions for good retrieval performance are made [Beckmann et al. 1990, p. 324]: the selection of the most appropriate subtree to store *data*, and the split of nodes in case the node capacity M is exceeded. With SELECTBEST(*selectedNode*, *sf*), the INSERT(.)-method selects the child node of *selectedNode* which is most 'adequate' to store *data* based on *sf*

[228] For convenience, we assume the same value M to be the maximum capacity of both internal and leaf nodes in the 'classical' R-tree for the following explanations. The explanations are easily transferable, though.
[229] More precisely, assuming a secondary index R-tree (as which the R-tree orginally was introduced), *data* actually has to be the reference to the data object.

and the child nodes' summaries. After adding *data* and its *sf* to the finally selected leaf node, INSERT(.) retraces the path towards the root node, splitting nodes containing more than M entries (SPLIT) and updating the nodes' summaries if necessary (UPDATE). An abstracted INSERT(.)-algorithm is depicted in algorithm 4.

ALGORITHM 4: An abstracted INSERT-algorithm for the classical R-tree.

Data: *data* the data object to insert
sf the spatial footprint of *data*
selectedNode the currently selected node in the traversal of the R-tree
tree the R-tree instance

```
1  function INSERT(data, sf)
2      selectedNode = tree.root
3      while selectedNode.level ≠ tree.maxLevel do
4          selectedNode = SELECTBEST(selectedNode, sf)
5      end
6      selectedNode.ADD(data, sf)
                                                  // backtracking ↓
7      while selectedNode.level ≠ 0 do
8          if selectedNode.NEEDSPLIT() then
9              selectedNode.SPLIT()
10         else if selectedNode.NEEDUPDATE(sf) then
11             selectedNode.UPDATE(sf)
12         end if
13         selectedNode = selectedNode.parent
14     end
15     if selectedNode.NEEDSPLIT() then
16         selectedNode.SPLIT()          // root has no summary to update
17     end if                                    // end of backtracking
18 end
```

In the classical R-tree, the most adequate node is determined by SELECTBEST(.) as follows: Select the node whose MBR requires the least enlargement of its surface area to include *data* (respectively *sf* associated with *data*). Resolve ties by choosing the node whose MBR has the smallest surface area [Guttman 1984, p. 50].
With regard to SPLIT, in general, a split algorithm should minimize the probability of invoking both created nodes for the same query [Manolopoulos et al. 2006, p. 9ff]. In his original paper, Guttman proposed three split algorithms which we briefly recapitulate at this point:

— **Linear Split**: Select two data objects as seeds for the two nodes. Take those objects that are as far apart as possible. Consider the remaining objects in random order and assign them to the node requiring the smallest enlargement of its MBR. The linear split requires linear runtime ($\mathcal{O}(M)$).

— **Quadratic Split**: Select two objects as seeds for the two nodes. Take those objects that create the most dead space if put together. The remaining objects are then assigned to the nodes, one at a time: At each step, the surface area expansion required to add each remaining object to each group is calculated. The object resulting in the greatest difference between the two groups is assigned. The quadratic split requires quadratic runtime ($\mathcal{O}(M^2)$).
— **Exhaustive Split**: All possible groupings are tested. The grouping covering the least overall surface area is selected. The exhaustive split as outlined by Guttman requires exponential runtime ($\mathcal{O}(2^M)$). Nevertheless, for example the algorithm of Becker et al. [Becker et al. 1991] finds the solution in polynomial time (as has already been discussed at various points of this work).

In practice, all of Guttman's algorithms lead to poor results since they only account for the minimization of the surface areas indexed by the MBRs [Manolopoulos et al. 2006, p. 18]. This often results in degenerated (i.e. thin and elongated) MBRs or heavy overlap between the MBRs of the different nodes. Consequently, a lot of alternative split algorithms have been suggested. One of the most noticed proposals in this context was introduced with the R*-tree. In contrast to Guttman's algorithms, it applies multiple optimization criteria at once:

— **Minimization of the surface area covered by each MBR**: It is the same criterion as utilized in Guttman's split algorithms, aiming at minimizing the indexed surface area.
— **Minimization of overlap between MBRs**: This criterion aims at minimizing the overlap between the MBRs of different nodes.
— **Minimization of MBR perimeters**: This criterion aims at preventing the degeneration of the MBRs, i.e. it seeks to create more quadratic rectangles. Obviously, this is beneficial for query types operating with spherical or quadratic query regions. Furthermore, quadratic objects can be packed more easily. Hence, it can also lead to the minimization of the indexed surface areas of upper-level nodes (similar to the other two aforementioned criteria) even though it is more of an indirect effect here.
— **Maximization of storage utilization**: Less nodes tend to be invoked during query processing when the storage utilization is high. Also, the tree height decreases.

The fourth criterion is basically addressed with the forced reinsertion procedure of the R*-tree which tries to reinsert selected objects of an overflowing node to prevent a node split. It is not further considered at this point. The other three criteria are approached by an adapted split algorithm. The algorithm works as follows [Beckmann et al. 1990, p. 326]:[230]

[230]Note the following explanations are tailored for MBRs but are easily transferable to point data. This is relevant for splitting *leaf nodes* administering point data.

Along each axis, the objects are sorted by a) the lower value and b) the upper value of their rectangles. For each sort, $M - 2m + 2$ different distributions of the $M + 1$ objects are determined. The k-th distribution ($k = 1, \ldots, M - 2m + 2$) is defined as follows: The first group contains the first $(m-1)+k$ objects, the second group the remaining objects. The sum S of all perimeter values of the different distributions is calculated. The axis with the minimum S is selected as split axis. Along the selected split axis, the distribution with the minimum overlap value is selected. Ties are resolved by choosing the distribution with the minimum surface area value.

In the R*-tree split algorithm, m can be set to different percentage values of M. Beckmann et al. claim that values of 40% yield the best results in their experiments. In literature, it is generally acknowledged that the R*-tree split algorithm is a significant improvement over Guttman's split algorithms. For that reason, although we try to keep the classical R-tree as unchanged as possible, we apply the R*-tree split algorithm for our R-tree implementation since the utilization of evidentially far from optimum split algorithms does not make sense. With the discussion of the SPLIT-procedure, the INSERT(.)-method has been fully considered.
Basically, a large number of different query types is applicable for the SEARCH(.)-method of the R-tree. Conceivable query types are e.g. point, window, range, or kNN queries. The corresponding query algorithms are usually optimized to utilize the MINDISTs from the nodes' summaries to the query points (see [Roussopoulos et al. 1995] as an example for kNN-queries). The concrete query algorithms for the query types assessed in our evaluation of the centralized application scenario are showcased in section 11.
At this point, the discussion of the classical R-tree and its important modifications from *previous research* is closed. In the subsequent section 10.2, the necessary modifications for integrating *our summarization approaches* into an R-tree are presented.

10.2. Modifications to the Traditional R-tree

Our MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summaries replace the MBRs which are traditionally used to describe the spatial footprints of the R-tree nodes and therefore, they have the same basic properties: They spatially bound the data in the subtree below and thus facilitate the calculation of lower bound distances to the 'spatial content' of a node. This allows for an unmodified utilization of the SEARCH(.)-function.
Nevertheless, they also feature two important differences: First, the spatial contents of a node may not be indexed by only a *single area* (as in the case of an MBR) but possibly by *several areas*. Second, our summaries require variable amounts of storage space. Hence, it is necessary to apply

changes to the functions and methods called inside the INSERT(.)-method (see algorithm 4).

10.2.1. SELECTBEST-MODIFICATIONS. As outlined in section 10.1, the 'classical' R-tree's SELECTBEST(.)-method selects *selectedNode*'s 'most appropriate' child node for inserting a data point *dp* by measuring the surface area enlargements of their MBRs and resolving ties on the basis of the covered surface areas. For an MBR, such enlargement calculations are simple: If necessary, enlarge a copy of the affected MBR to contain *dp*.
However, both MBRQT^{c,a} as well as QTMBR$^{b}_{c,a}$ summaries require a *complete recalculation* to measure a possible surface area enlargement.[231] Since this procedure is rather inefficient, the most appropriate child node is selected as follows: Select the node for which *dp* is contained within its summary's boundaries. Resolve ambiguous (more than one node is found) or inconclusive (no node is found) scans by selecting the node with the smallest distance between its summary's center point and *dp*. To define the center point, we first determine the MBR of the areas indexed by the summary. Then, we utilize this MBR's midpoint as center point. If there are still several candidate nodes, select a random candidate node to insert *dp*.

10.2.2. BACKTRACKING OPTIMIZATION. Updating an MBR with *dp* comes down to enlarging the MBR to also enclose *dp*. This applies for all MBRs on the retraced path from the leaf node storing *dp* up to the root node (i.e. the path which is ascended in the backtracking step of the INSERT(.)-method, see algorithm 4 and section 10.1). It is independent of whether the lower-level node has been split or its MBR has been updated. Hence, updating the MBRs of the nodes on the insertion path of a traditional R-tree at the end of an INSERT(.)-procedure is very cheap from a computational point of view.
However, this is not applicable for MBRQT^{c,a} and QTMBR$^{b}_{c,a}$: If *dp* is not covered by the selected leaf node's summary, its *complete recalculation* is inevitable. Recalculating the summary of a leaf node necessitates an 'UPDATE-ascension' from the leaf node up to the root node, enforcing a *complete recalculation* of the summaries associated with all nodes on the corresponding path.[232] To minimize the frequency of this event, the NEEDUPDATE-method is modified so that leaf and internal nodes are handled differently. For a leaf node, the NEEDUPDATE-method only returns *true* if *dp* is not covered by the summary. For internal nodes on the retraced path, it leads to returning *true* if the leaf node taking *dp* was split or up-

[231] In general, the necessity of a *complete recalculation* or the sufficiency of cheaper options for determining the surface area enlargement is decided by the properties of a summarization approach.

[232] Note that the complete recalculation of internal nodes' summaries from all the data points administered in their subtree below can be very expensive. See section 10.4 for how we handle this issue.

dated. Otherwise, no summary recalculations need to be conducted. This is feasible because the summaries of internal nodes are calculated in such a way that their indexed areas completely cover the areas indexed by the summaries of all subordinated child nodes (see section 10.4).

10.2.3. SPLIT-MODIFICATIONS. When the available *bpp* of a node are exceeded after inserting *dp*, the corresponding node has to be split. As outlined in section 10.1, we apply the R*-tree split algorithm (R*-split).
Unlike MBRs (which can be described by two corner points and thus allocate a fixed amount of storage space), the storage space consumption of $\text{MBRQT}^{c,a}$ and $\text{QTMBR}^{b}_{c,a}$ summaries can vary. To handle this, there are two options: preallocation or postallocation of a summary's storage space. Preallocation means that the summary calculates its 'worst case page size' and allocates it right from the start. For simple summaries like MBRs, preallocation is usually the appropriate choice. In contrast, the size of more sophisticated summaries (like $\text{MBRQT}^{c,a}$ and $\text{QTMBR}^{b}_{c,a}$ summaries) may depend on multiple criteria which make it difficult to assess the worst case page size in advance. For $\text{MBRQT}^{c,a}$ and $\text{QTMBR}^{b}_{c,a}$, four different summary description types are available: `lq-z`, `lq-nz`, `cblq-z`, and `cblq-nz`.[233] From these, the most storage-space-efficient option is selected. In conjunction with the parameterizability of our summarization approaches, precalculating a worst case page size is very difficult. Furthermore, the result value is most likely very inappropriate for realistic data sets (which are usually erratic) since the worst case page size should occur in conjunction with a regular spatial distribution of the data points to summarize. Therefore, we use postallocation for our $\text{MBRQT}^{c,a}$ and $\text{QTMBR}^{b}_{c,a}$ summaries. If the insertion of dp requires an UPDATE of the *dp-storing leaf node*, first, its summary is calculated and the most storage-space-efficient representation is selected. Then, it is checked whether the *bpp* of the *summary-storing internal node* are exceeded. The procedure is recursively propagated towards the root. This results in a dynamic reallocation of the summaries' storage footprints in the course of an 'UPDATE-ascension' on the path towards the root during which the split of *internal nodes* may be forced—even if the *leaf node* storing *dp* was not split. Consequently, a second invocation of the NEEDSPLIT(.)-method is required after the recalculation of a summary for internal nodes.

10.3. Requirements for Summaries in an R-tree

In the following, we briefly discuss the requirements for summaries utilized in an R-tree. Generally, the summaries fulfill the role of 'signposts' when performing operations on the R-tree. If the summarized node is an internal node, all the data points administered in the leaf nodes of its subtree need to be 'described' by its summary. If the node is a leaf node, only the

[233] Note that the direct representation is not an option in the centralized application scenario.

data points administered in this very leaf node must be described. Based on these summaries, algorithms select the appropriate 'pathway' in the tree structure for the further processing of their specific tasks. Hence, the summaries' qualities of describing the 'contents' of the nodes are of utmost importance. Generally, a summary is either a single or a set of geometric form(s) which spatially enclose all the data points in or below the corresponding node. In the centralized application scenario, the spatial quality of a summary is essentially determined by its ability to delimit the extents of this 'data point cloud' as accurately as possible. The spatial accuracy of a summary depends to a large extent on the indexed surface area: The less surface area is covered, the less dead space is contained within the summaries.
Of course, it is not suitable to optimize summaries solely by aiming at minimizing the indexed surface area. This can lead to very thin, elongated (or degenerated) indexed areas—such as iso-oriented rectangles which have a very small extent in one dimension but a very large extent in the other dimension (assuming $d = 2$). Such rectangles result in very small surface areas but are completely unsuitable for e.g. kNN queries due to their great extent in one of the dimensions.[234] Furthermore, it has to be taken into account that summaries have to be storage-space-efficient to prevent that the fanout of an R-tree decreases too much. Also, the required calculations on the summaries (such as distance calculations) have to be computationally cheap. Finally, summaries at the same level of the R-tree which overlap each other complicate the selection of the most appropriate subtree. Hence, the requirements for summaries in an R-tree (which are partially conflicting) can be subsumed as follows:

— The summaries have to be an accurate description of the nodes' spatial footprints, covering little surface area respectively enclosing little dead space.
— The summaries should have a small storage space footprint.
— The areas indexed by the summaries have to enable cheap calculations for required operations such as distance or containment tests.
— The degeneration of indexed areas to thin, elongated geometric forms has to be prevented.
— The overlap between summaries of different nodes at the same level of the R-tree has to be minimized.

The latter two requirements are in large parts solvable by utilizing suitable split algorithms to divide the data point set of the split node into two appropriate groups. As stated in section 10.2.3, we apply the R*-split which has been developed with both goals in mind. The other requirements have to be met by the corresponding summarization approach itself, though.

[234] Hence, the summary calculation faces similar problems as the split algorithm. Ultimately, the increased occurrence of degenerated rectangles in the classical R-tree was one of the reasons leading to the development of the R*-tree.

With respect to cheap calculations, an iso-oriented rectangle is an obvious choice for the geometric shape of indexed areas. This rectangle is the most commonly used geometric shape in the spatial indexing domain. Polygonal shapes such as the convex hull are much more costly with regard to distance and containment calculations, and are also suboptimal with regard to small storage space footprints. Spheres have an even smaller storage space footprint than rectangles and also offer cheap calculations. On the other hand, they can be much harder to calculate (for example in the case of a minimum bounding sphere compared to an MBR) and are also significantly coarser descriptions in most cases.
The MBR is a suitable summary with regard to all listed requirements except the minimal inclusion of dead space: The MBR of a data point set is usually a very coarse approximation of the 'data point cloud' and can contain lots of dead space. Most problematic, this coarse granularity makes it hard to explore the immediate neighborhood of a query point. This insight is essentially the motivation to utilize the $\text{MBRQT}^{c,a}$ and $\text{QTMBR}^{b}_{c,a}$ summarization approaches within an R-tree.

10.4. Summary-from-Summaries Calculation and Storage Structures

The subtrees of the upper-level nodes in an R-tree can quickly contain a lot of data points. Hence, the calculation of upper-level summaries *not* from data points but from lower-level summaries is indispensable for an efficient calculation of the internal nodes' summaries in an R-tree. As noted in section 10.2.2, the $\text{MBRQT}^{c,a}$ respectively $\text{QTMBR}^{b}_{c,a}$ summaries of a node need to be recalculated in case a newly inserted data point is assigned to this node and the point is not contained within the bounds of the node's summary. This is because we do not have a definition of how to add a new point to an already existing summary: The input for calculating a summary is always the entire data point set to describe. Furthermore, it would be difficult to define an appropriate procedure which works in such a way that the summaries do not distort over time, i.e. completely lose their advantage of greater spatial accuracy in comparison to an MBR summary. Nevertheless, especially for upper-level nodes, in case the summaries would always be calculated from the associated data points, this would result in the need to collect enormous amounts of data points from the leaf nodes below. Furthermore, the greater the set of data points to summarize, the more expensive the summary calculation. Thus, the summary calculation from data points is unfeasible for internal nodes of the R-tree as the construction of the R-tree would take too long. A more practicable approach (both with regard to preserving the spatial accuracy over time as well as computational efficiency) is to utilize the child nodes' summaries as input for calculating the parent node's summary.
According to our R-tree modification desribed in section 10.2.2, the summary-from-summaries calculation is conducted for *all* summarization

approaches. In the following, we outline how to calculate an MBRQT^{c,a} respectively QTMBR$^{b}_{c,a}$ summary from other summaries. The calculation of an 'upper-level' MBR for a set of input MBRs is trivial and therefore not explicitly outlined at this point.[235] Hence, in contrast to the summary calculation from a set of data points (like for the leaf nodes), the input for creating an internal node's summary is a set of rectangular areas. For an internal node N which is at level i of the R-tree, the input is the set of rectangular areas indexed by the summaries of N's *direct* child nodes (at level $i + 1$). Due to our backtracking optimization, it is also required that the child nodes' spatial footprints are completely covered by the parent node's spatial footprint (see section 10.2.2). Furthermore, the respective storage structures for writing MBRQT^{c,a} respectively QTMBR$^{b}_{c,a}$ summaries to disk pages are outlined in the following.

10.4.1. MBRQT^{c,a}. We start with the depiction of the summary-from-summaries calculation process. At first, the full-precision MBR of the set of input rectangles is calculated (a). If the MBR covers a surface area greater than the threshold surface area a, the MBR-interior data space is partitioned by a *full quadtree* of the target depth *td* (b). This means that each internal node of the quadtree structure has four children with the leaf nodes being at level *td*. We calculate *td* in dependence of parameter c as $\lfloor \log_4\ c \rfloor$.[236] Initially, the space partition corresponds to a regular grid and all quadtree leaf nodes are colored white.
Then, each subspace of the quadtree space partition is tested whether it is overlapped by any of the input rectangles. If this is the case, the corresponding leaf node is colored black (c). After testing all subspaces of the space partition, the quadtree is condensed (d). This process condenses both four black sibling nodes as well as four white sibling nodes into their similarly colored parent node.[237] The condensation is applied recursively, starting at level *td* (i.e. the initial leaf level) and ascending the quadtree structure until reaching the root. Thus, in extreme cases, the quadtree is condensed into a single black cell.

[235] Note that an MBR summary calculation from other MBR summaries would in principle not necessarily be required: The coverage of the newly added data point and all data points in the corresponding node's subtree can be ensured by simply resizing the MBR in case the newly added data point is not contained within the original MBR's bounds. However, we want to keep the R-trees as similar as possible to each other for the evaluation. Also, a corresponding specification of the calculation of the internal nodes' MBR summaries allows to utilize the same interfaces for all summarization approaches. Consequently, a summary-from-summaries calculation is also conducted for internal nodes with MBR summaries.

[236] The applied formula ensures that the initial quadtree space partition does not feature more than c cells. Thus, also the final partition cannot feature more cells—even in the worst case with regard to the resulting number of cells.

[237] Note that four white sibling nodes cannot occur for a quadtree construction from a data point set, see procedure described in section 3.2.

After the condensation has been applied, the calculation of an MBRQT^{c,a} summary from other MBRQT^{c,a} summaries is completed (e). See Figure 88 for an exemplary depiction of the steps (a) to (e).

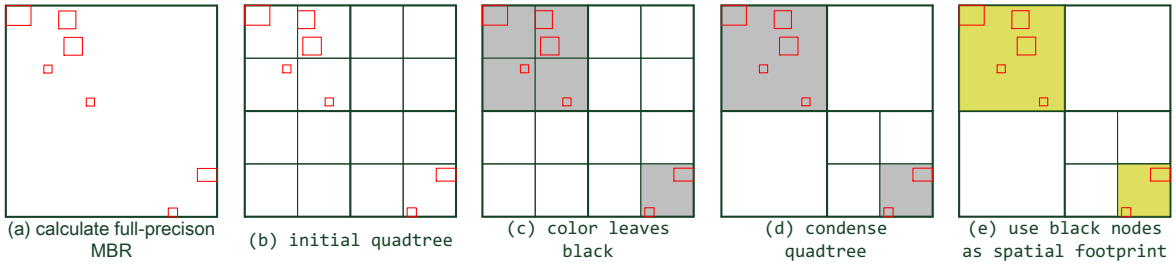


Fig. 88: Exemplary depiction of the summary-from-summaries calculation process for MBRQT^{c,a}, assuming $td = 2$. The subspaces of the 'black leaves' are colored light grey for a better visibility. The red-stroked rectangles are the input rectangles, the yellow areas in (e) constitute the indexed areas of the hypothetical upper-level node for which the summary was calculated.

Concerning the structure of the MBRQT^{c,a} summaries' bit vectors, there are slight adaptions compared to the corresponding bit vectors of the distributed application scenario. The entire bit vector structure of an MBRQT^{c,a} summary is depicted in Figure 89 on the left.[238] Again, both the LQ scheme and the CBLQ scheme are available for encoding the quadtree information. In general, the MBRQT^{c,a} summary bit vector is subdivided into a metadata and a payload component. The metadata component contains the information about the summary description type. As mentioned in section 10.2.3, due to the omission of the direct representation, there are four encoding options left: `lq-z`, `lq-nz`, `cblq-z`, and `cblq-nz`. Hence, we require 2 bits of metadata information for the summary description type.[239] The payload component contains the correspondingly encoded actual summary information. It has the same structure as the payload of the MBRQT^{c,a} summaries in the distributed application scenario, i.e. the full-precision MBR bounds are followed by the quadtree metadata and the raw quadtree data. In contrast to the distributed application scenario, no bit vector clipping is applied: The bit vector is byte-aligned via bit stuffing. The stuffed byte concludes the bit vector of the MBRQT^{c,a} summary.
Furthermore, the storage space is allocated word-wise. As we assume a 64-bit system, this means that the storage space is allocated in chunks of 8 B. Consequently, the byte-aligned summary is finally word-aligned before

[238] Remember that in Figure 27 (page 97) outlining the bit vector structures for the distributed application scenario, not the entire bit vector structures of the summarization approaches are depicted: The summarization overhead (due to the Java implementation) and the metadata information (containing the resource description type and the resource size) are not depicted (see Table 3 on page 121).

[239] Note that in case the basic MBR is not refined with a quadtree, both bits of the metadata component are set to '0' which corresponds to the metadata code of `cblq-nz` in the centralized application scenario.

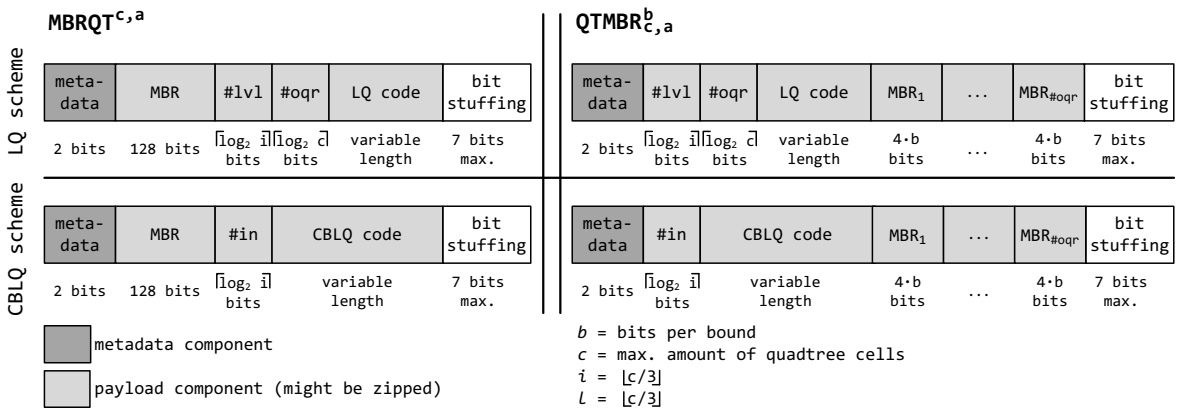


Fig. 89: Depiction of the bit vector structures for MBRQT^{c,a} (left) and QTMBR$^{b}_{c,a}$ (right) summaries in the centralized application scenario.

being stored on disk. See Figure 90 for a conceptual depiction. For example, presume that a summary requires 303 bits for storing all of its information. At first, the summary is byte-aligned via bit stuffing (which is depicted in Figure 89), i.e. $303/8 = \lceil 37.875 \rceil = 38$ B are required after byte-aligning the summary. Then, this byte-aligned summary must be word-aligned, one word accounting for 8 B. From 38 upwards, the next multiple of 8 is 40. Hence, 40 B are allocated in total for a summary which requires 303 bits of information.

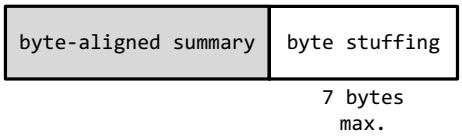


Fig. 90: Conceptual depiction of the word-aligned storage structure of the MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summaries.

10.4.2. QTMBR$^{b}_{c,a}$. Anew, we start with describing the summary-from-summaries calculation. First, the entire universe U is partitioned by means of a full quadtree of depth *td* (a). Again, all leaf nodes of the quadtree structure start out as white nodes. Then, each leaf node whose corresponding subspace is overlapped by any of the input rectangles is colored black (b). Next, *only* the white sibling nodes are condensed into their parent nodes (c). Hereafter, the basic quadtree structure has been determined and the quantized MBRs for the occupied subspaces have to be calculated.
For this purpose, the following procedure is executed for *each* occupied subspace: First, determine all areas of overlap of the occupied subspace with any of the input rectangles. Then, calculate the full-precision MBR for this set of overlap areas (d). Finally, adjust this full-precision MBR to the quantization grid of the occupied subspace (e). See Figure 91 for an exemplary depiction of the steps (a) to (e).
After processing all occupied subspaces in this way, the QTMBR$^{b}_{c,a}$ calculation from other QTMBR$^{b}_{c,a}$ summaries is completed.

Again, there are slight adaptions to the structure of the bit vectors of the $\text{QTMBR}^b_{c,a}$ summaries in comparison to the distributed application scenario. The $\text{QTMBR}^b_{c,a}$ summary is divided into a metadata and a payload component, too. The metadata component is the same as for $\text{MBRQT}^{c,a}$ since $\text{QTMBR}^b_{c,a}$ offers the same four encoding options. The payload component contains the correspondingly encoded summary information which has the same structure as the $\text{QTMBR}^b_{c,a}$ summary information in the distributed application scenario, i.e. the quadtree metadata and the raw quadtree data are followed by the data of the quantized MBRs. Again, the bit vector is byte-aligned by bit stuffing, and the stuffed byte concludes the bit vector of a $\text{QTMBR}^b_{c,a}$ summary (see Figure 89 on the right). Finally, the byte-aligned summary is word-aligned.

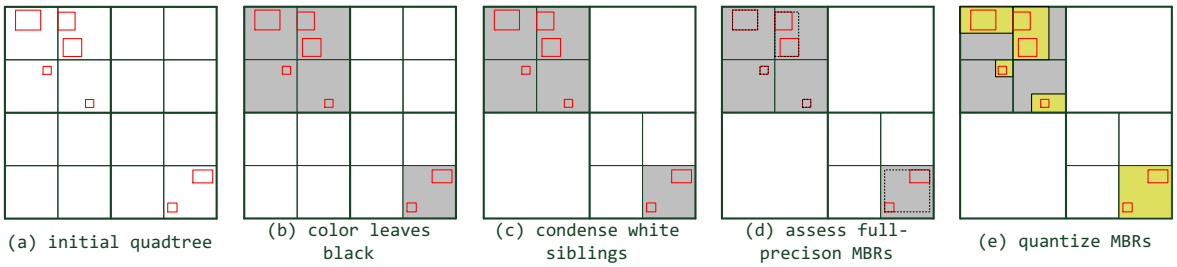


Fig. 91: Exemplary depiction of the summary-from-summaries calculation process for $\text{QTMBR}^b_{c,a}$, assuming $td = 2$ and $b = 2$. The quantized MBRs from (e) are used as the hypothetical upper-level node's spatial footprint.

For both $\text{MBRQT}^{c,a}$ and $\text{QTMBR}^b_{c,a}$, the integration of the #lvl and #oqr information into the LQ codes respectively of the #in information into the CBLQ codes is required for the unambiguous decoding of a node N's child node entries (each of which consists of a pointer to a child node and the child node's word-aligned summary). In contrast to the MBR summaries, the $\text{MBRQT}^{c,a}$ and $\text{QTMBR}^b_{c,a}$ summaries may allocate variable amounts of storage space. Therefore, one would usually require meta-information about the word length of each summary to be able to access specific entries. This information would be best stored in N's metadata section for which we specified that 16 B are available (see section 10.1). Of course, 16 B are not sufficient to store the summary sizes of all child nodes of N when e.g. page sizes of 4,096 B are assumed. Nevertheless, this information is not even necessary as *no specific* entries of N have to be accessed: All operations on the R-tree require the deserialization of all of N's entries because the spatial footprints of all child nodes need to be considered for each decision. Hence, it is sufficient to deserialize the individual entries sequentially. For this, we iterate byte-wise over N's page data while simultaneously decoding the binary stored pointer and summary data. The aforementioned meta-information of the encoding schemes make it possible to determine when the words for child node cn_i end and when the words for child node cn_{i+1} begin. Since we do not store any byte length information in the linear quadtree codes, the byte-wise iteration has to be applied

anyways for decoding the MBRQT^{c,a} respectively QTMBR$^{b}_{c,a}$ summary information.

10.5. Effects and Trade-Offs of the Integration of Sophisticated Summarization Approaches

In the following, we discuss the effects of the integration of our summarization approaches into an R-tree.

Obviously, the implementation becomes more difficult: the summary calculation, the just discussed more complex decoding procedure (due to the variable summary sizes), the backtracking optimization (see section 10.2.2), and the 'UPDATE-ascension' as mentioned in connection with the split modifications (see section 10.2.3) all complicate the implementation. Nevertheless, we think that the additional complexity is on a manageable level.

The introduced effects on the operation of the R-tree are more interesting. It makes sense to consider the impacts on a) the R-tree construction and b) the execution of queries separately.

The construction of an R-tree definitively takes significantly longer with our summarization approaches as both the MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ calculation are significantly more expensive with regard to CPU time than the MBR calculation. This is in particular because for both of them, four different encoding options are tested (which include the zipping of summaries) in the course of *each* summary calculation—and a lot of summaries need to be calculated while constructing an R-tree. Furthermore, for both MBRQT^{c,a} and QTMBR$^{b}_{c,a}$, the complete recalculation of a *leaf node*'s summary might be triggered which can be avoided for the MBR due to its simplicity (see section 10.2.2).[240] We accept the significantly longer construction times as a trade-off for faster query times. As outlined in section 9, we assume a rather static database, i.e. the set of data points to be queried does not grow and shrink at high rates, and the presumed use case is focused on frequently querying the existing database.

With regard to the query times, R-trees and related structures are usually optimized and evaluated for the minimization of disk accesses as it is regarded to be the time-dominant operation for disk-resident multidimensional data structures (similar to the number of distance calculations in the metric domain, see section 2.6.3). We think that by utilizing our summarization approaches, the number of disk accesses can possibly be reduced by significant extents due to the greater spatial accuracy of their summaries in comparison to the MBR. Nevertheless, there are also other aspects which have an impact on the query times—and not all are favorably influenced by the integration of our summarization approaches.

[240] As outlined before, for the internal nodes, the summary-from-summaries calculation is also conducted for the MBR approach. However, also this calculation should be significantly less expensive for the MBR approach in comparison to MBRQT^{c,a} and QTMBR$^{b}_{c,a}$.

Firstly, this concerns CPU time. Since our MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summaries might index multiple areas containing data points, the calculations for determining the MINDIST between the spatial footprint of a node and the query point are more expensive. Furthermore, there are four different encoding schemes for both MBRQT^{c,a} and QTMBR$^{b}_{c,a}$, and the reconstruction of the indexed areas from the bit vectors is much more complex. Consequently, the decoding of the MBRQT^{c,a} respectively QTMBR$^{b}_{c,a}$ summaries as a whole is significantly more expensive. The application of suitable caching strategies can mitigate the decoding overhead, though.[241] Another aspect is the fanout of the R-tree. Depending on the storage space consumption of the summaries, the fanout of the internal nodes changes as only a fixed amount of *bpp* bytes is available. In case the MBRQT^{c,a} respectively QTMBR$^{b}_{c,a}$ summaries consume more storage space on average than an MBR, the fanout of the tree decreases. Consequently, the height of the tree increases by tendency. This is disadvantageous as the data points are administered in the leaf nodes, i.e. the path to the leaf nodes is longer and a greater amount of disk pages has to be accessed to reach the leaf node level. In case the summaries consume less storage space than an MBR summary, the internal nodes' fanout increases. As already discussed several times, the aims of storage space efficiency and great spatial accuracy are conflicting. In general, we expect our MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summaries to consume more storage space on average than MBRs, i.e. it can be expected that the fanout of the corresponding R-trees is lower. MBRQT^{c,a} in fact has to spend more storage space as it is based on a full-precision rectangle and additionally requires the metadata block, see section 10.4.1. In total, we think that the trade-off between reduced I/O (enabled by greater spatial accuracy) on the one hand, and CPU time (caused by the greater number of indexed areas and the more complex decoding procedure) as well as the presumably unfavorable impacts on the R-tree's fanout (causing more I/O) on the other hand might turn out favorably for our summarization approaches. The corresponding evaluation is conducted in section 13.

[241] Note that all these aspects concerning the decoding overhead also apply during the constructions of the respective R-trees.

11. QUERY ALGORITHMS

In this section, the conceptual algorithms for the evaluated query types of the centralized application scenario are outlined. We apply both range queries (section 11.1) and kNN queries (section 11.2). The query algorithms are not restricted to any summarization approach and can be applied for R-trees with arbitrary node summaries. From a conceptual point of view, the query algorithms solve the resource selection and also the trivial result merging problem with regard to the corresponding query type.

11.1. Range Query Algorithm

In the following, a conceptual range query algorithm for R-trees is outlined. As described in section 2.5, a range query retrieves all data points which are no further from the query point q than a distance q_{rad}. Consequently, the query algorithm needs to visit all nodes in the R-tree whose MINDIST between their spatial footprint and q is smaller or equal to q_{rad}. The query result can be of arbitrary size—it might be empty, contain all the data points administered in the R-tree, or anything in between.
Below, we verbally describe a conceptual range query algorithm for the R-tree. It is depicted more formally in algorithm 5.

At first, the *root* node of the R-tree is inserted into the queue Q_{nodes} which maintains the nodes that still need to be considered (line 2). As long as Q_{nodes} is not empty, the elements of Q_{nodes} are successively processed (lines 3-19). For this, the first element *node* is removed from Q_{nodes} and considered afterwards (line 4).
If *node* is an internal node, its child nodes are successively considered if they need to be inserted into Q_{nodes} (lines 5-11). If the MINDIST between the current child node $node_{child}$ and q is smaller or equal to q_{rad}, $node_{child}$ is added to Q_{nodes} (lines 7-9).
If *node* is a leaf node, its data points are successively assessed if they are part of the query result (lines 12-18). In case the distance between the currently considered data point dp and q is smaller or equal to q_{rad} (line 14), dp is added to the query result list L_{result} (line 15).
When Q_{nodes} is empty, L_{result} is returned as the range query's result and the algorithm terminates (line 20). L_{result} contains all data points that are located within the query ball of the range query.

11.2. kNN Query Algorithm

In the following, a conceptual algorithm for precise kNN queries in R-trees is outlined. As described in section 2.5, an kNN algorithm retrieves a natural number k of closest data points to a query point q. In general, the kNN algorithm for R-trees is very similar to the kNN algorithm for the distributed application scenario but also exhibits a few differences (see section 5.2):

ALGORITHM 5: A conceptual range query algorithm for R-trees.

Data: L_{result} the query result list of data points, initially empty
q the query point
q_{rad} the query radius
root the root node of the R-tree
Q_{nodes} queue of nodes to consider by the algorithm
node currently considered R-tree node
$node_{child}$ currently considered child node of *node*
dp data point stored in the currently considered leaf node *node*
Output: L_{result} as the list of data points which are located within the query ball

```
1  Algorithm rangeQuery(L_result, q, q_rad, root)
2      Q_nodes.add(root)
3      while Q_nodes ≠ ∅ do
4          node = Q_nodes.pop()
5          if node.isInternalNode() then
6              for node_child in node.getChildren() do
7                  if MINDIST(node_child, q) ≤ q_rad then
8                      Q_nodes.push(node_child)
9                  end
10             end
11         end
12         if node.isLeafNode() then
13             for dp in node.getDataPoints() do
14                 if distance(dp, q) ≤ q_rad then
15                     L_result.add(dp)
16                 end
17             end
18         end
19     end
20     return L_result
```

— a priority queue of nodes (PQ_{nodes}) replaces the ranked resources,
— it has to be differentiated between internal nodes and leaf nodes, and
— the pruning of internal nodes can lead to the exclusion of subtrees from further consideration (i.e. in contrast to the distributed application scenario, possibly not all spatial footprints of all the resources have to be considered).

Our conceptual kNN algorithm is an adapted version of the kNN algorithm of Roussopoulos et al. ([Roussopoulos et al. 1995]) and works iteratively instead of being based on recursion. It is described verbally in the following and depicted more formally in algorithm 6:

At first, *root* is inserted into PQ_{nodes}. PQ_{nodes} maintains the nodes which are (possibly) to be considered (line 2). Generally, the nodes in PQ_{nodes}

are sorted in ascending order by their MINDIST to q. The pruning distance $dist_{prun}$ represents the distance to the current k-th nearest neighbor. Initially, it is set to ∞ (line 3). Then, the query result list L_{result} is filled with k dummy data points whose distances to q are also ∞ (line 4).[242]

As long as PQ_{nodes} is not empty, the elements of PQ_{nodes} are successively processed (lines 5-27). For this, the first element *node* is removed from PQ_{nodes} and considered afterwards (line 6). In case the MINDIST between *node* and q is not smaller than $dist_{prun}$, the consideration of PQ_{nodes} is aborted (lines 7-9) and L_{result} is returned as query result (line 28).

Otherwise, if *node* is an internal node, its child nodes are successively considered (lines 10-17). For the currently considered $node_{child}$, at first, the distance *dist* is calculated which is the MINDIST between $node_{child}$ and q (line 12). If *dist* is smaller than $dist_{prun}$, $node_{child}$ is inserted into PQ_{nodes} with *dist* being its priority (lines 13-15). The smaller *dist*, the higher the priority of $node_{child}$. In case of equal priorities of nodes, the newly added $node_{child}$ is placed at the backmost position of its pack.

If *node* is a leaf node, its data points are successively assessed if they are part of the current k nearest neighbors (lines 18-26). In case the distance between the currently considered data point dp and q is smaller than $dist_{prun}$ (line 20), dp replaces the current k-th nearest neighbor which is at the backmost position of L_{result} (line 21). Afterwards, L_{result} is sorted in ascending order by distance to q (line 22). Finally, $dist_{prun}$ is set to the distance of the last element of L_{result} which is the current k-th nearest neighbor (line 23).

When PQ_{nodes} is empty or the `while`-loop has been aborted prematurely, L_{result} is returned as query result and the algorithm terminates (line 28). L_{result} contains the k closest data points to q. Hereby, the k closest data points are sorted in ascending order by distance to q.

[242] Note that the dummy data points are only included for descriptive convenience and are not present in the real implementation of the kNN algorithm.

ALGORITHM 6: A conceptual kNN algorithm for R-trees (adapted from [Roussopoulos et al. 1995]).

Data:

L_{result}	list of the current k closest data points with their distances to q
q	the query point
k	number of nearest neighbors to retrieve
root	the root node of the R-tree
PQ_{nodes}	priority queue of nodes to consider by the algorithm
$dist_{prun}$	current pruning distance
node	currently considered R-tree node
$node_{child}$	currently considered child node of *node*
dist	distance (MINDIST) between the current $node_{child}$ and q
dp	data point stored in the currently considered leaf node *node*

Output: L_{result} as the list of the k closest data points to q

```
1  Algorithm kNNQuery(L_result, q, k, root)
2      PQ_nodes.insertWithPriority(root,∞)
3      dist_prun = ∞
4      fillWithDummyDataPoints(L_result)
5      while PQ_nodes ≠ ∅ do
6          node = PQ_nodes.pop()
7          if dist_prun ≤ MINDIST(node, q) then
8              break
9          end
10         if node.isInternalNode() then
11             for node_child in node.getChildren() do
12                 dist = MINDIST(node_child, q)
13                 if dist < dist_prun then
14                     PQ_nodes.insertWithPriority(node_child, dist)
15                 end
16             end
17         end
18         if node.isLeafNode() then
19             for dp in node.getDataPoints() do
20                 if distance(dp, q) < dist_prun then
21                     L_result.replace(k − 1, dp)
22                     sort(L_result)
23                     dist_prun = L_result.get(k − 1).getDistanceToQueryPoint()
24                 end
25             end
26         end
27     end
28     return L_result
```

12. EVALUATION ENVIRONMENT

In this section, we briefly outline the evaluation environment. First, we explain how we assess the results (section 12.1). Then, the evaluated data collections are described (section 12.2). Afterwards, the diverse settings (parameterizations of the summarization approaches, R-tree parameters, query parameters, etc.) are presented and justified (section 12.3).

12.1. Assessment of the Results

As already mentioned in section 10.5, disk-resident multidimensional data structures like the R-tree are usually evaluated based on the number of disk accesses since the page access time on hard drive disks (including the mechanical placement of the read/write head at the appropriate section on the disk) is regarded to be the most time-consuming operation.[243] Consequently, our evaluation of the queries is focused on the number of required disc accesses.

Nevertheless, since our summaries presumably introduce substantial overhead with regard to CPU time (see section 10.5), we also evaluate runtimes in milliseconds (ms). It is planned that this includes the construction times of the R-trees using the respective summarization approaches as well as the execution times of the queries.

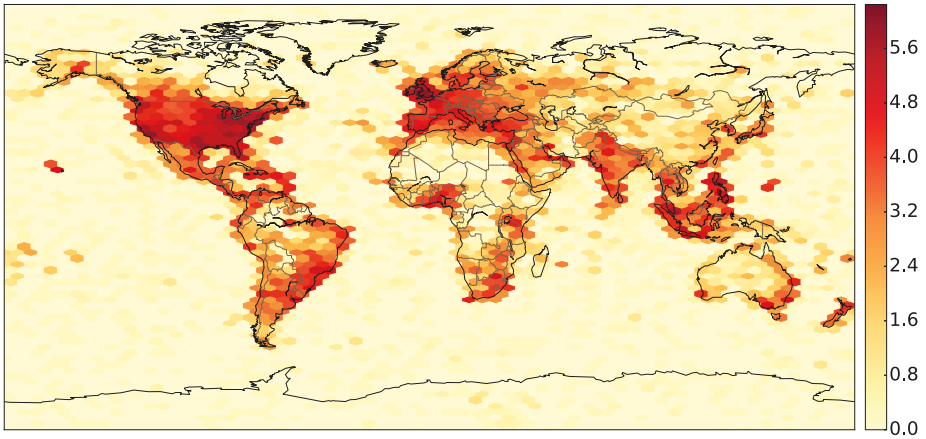
The query experiments are computed on machines with an Intel i7-7700 @3.6 GHz and 64 GB of DDR4 memory (@2,400MHz) under a Linux 64-bit 4.12.12 kernel. For the construction times of the R-tree, we utilize equally equipped hardware but running under a 64-bit Windows 10 Enterprise LTSB operating system.

12.2. Data Collections

We evaluate two data collections to guarantee the robustness of the results. The `T` collection is created from the set of geotagged Twitter tweets which has already been utilized for the `T1` and `T2` collections of the distributed application scenario. In contrast to the `T1` and `T2` collections, no division into a training and a test data set is conducted but the data points featuring the coordinates ($x = 0; y = 0$) are removed. It is obvious that they are no valid spatial positions for tweets but simultaneously occur quite frequently (28,029 occurrences). Consequently, 26,739,754 data points remain for the `T` collection. For these, 21,367,988 unique locations exist, i.e. about 20% of the data points are duplicates. The `R` collection is a set of 25 million randomly generated data points. The data points are evenly distributed in the data space. For their creation, we initialized a random number generator and alternatively created float values between -180 and $+180$ (the currently created data point's x value), and between -90 and $+90$ (the

[243] Nowadays, fast hard drive disks have average page access times of about 9 to 15 ms, see [Guri et al. 2017, p.10] and https://www.webopedia.com/TERM/A/access_time.html (last visit: 16.08.2018).

currently created data point's y value). No duplicate data points exist for the R collection. The spatial distributions of the collections' data points are depicted in Figure 92a (T collection) respectively Figure 92b (R collection).



(a) Spatial distribution of the data points of the T collection. The coloring is log-scaled.

(b) Spatial distribution of the data points of the R collection. The bins at the data space's boundaries have a different color as they are smaller and therefore contain less data points. The coloring is log-scaled.

Fig. 92: Spatial distribution of the data points of the collections that are used in the evaluation of the centralized application scenario.

12.3. Experimental Setup

In the following, we outline the setup for the evaluation.
The parameterizations tested for MBRQT^{c,a} respectively QTMBR$^{b}_{c,a}$ are listed in Figure 93. In total, six different parameterizations are tested for MBRQT^{c,a}. For QTMBR$^{b}_{c,a}$, four different parameterizations are tested.
The idea behind the parameterizations of MBRQT^{c,a} is to keep the amount of utilized storage space close to that of the MBR. As the evaluation of the distributed application scenario showed, already the use of very small internal quadtrees can lead to a significant reduction of the indexed surface areas in comparison to a plain MBR. Across all parameterizations,

the smallest threshold surface area value is $a = 1.0E{-}7\ dsu^2$. For smaller MBRs, we do not conduct a refinement as we assume that for data point sets which are spatially that narrow, the MBR is a sufficiently accurate summary. Generally, the variation of the parameters is set in such a way that if possible, it is revealed whether rather coarse or rather accurate quadtree refinements are more appropriate for MBRQT^{c,a}. Note that even with a maximum amount of only 8 respectively 16 quadtree cells, substantial improvements can be achieved with regard to a node's indexed surface area in case the data points to describe are located favorably.

The smallest threshold surface area for QTMBR$^{b}_{c,a}$ is the same as for MBRQT^{c,a}, i.e. $a = 1.0E{-}7\ dsu^2$. Nonetheless, the occupied cells of the basic quadtree are additionally refined with quantized MBRS of either $b = 8$ or $b = 12$ bits per bound (i.e. 256 or 4,096 positions can be distinguished per dimension). Thus, at first thought about the respective parameterizations, it might seem that the maximum spatial accuracy of the QTMBR$^{b}_{c,a}$ summaries is much greater. Nevertheless, it has to be considered that the basic MBRs of MBRQT^{c,a} can be 'infinitely' small (i.e. a point in the extreme case) as they are full-precision rectangles. For QTMBR$^{b}_{c,a}$, the smallest indexable spatial unit is basically a cell of the quantization raster of an occupied quadtree cell.[244] Hence, we have to parameterize QTMBR$^{b}_{c,a}$ in such a way if we want the QTMBR$^{b}_{c,a}$ summaries to keep up with the MBR respectively MBRQT^{c,a} summaries to some extent with regard to the smallest indexable spatial units.[245] The parameterizations of QTMBR$^{b}_{c,a}$ are varied to reveal whether it is more beneficial to increase the quantization accuracy of the refining MBRs or to spend the storage space for additional cells of the basic quadtree. Due to the small amount of MBRQT^{c,a} parameterizations respectively QTMBR$^{b}_{c,a}$ parameterizations, for the remainder of this thesis, we address the parameterizations by their *names* specified in Figure 93 because this increases the readability of text. For example, MBRQT$^{8,0.001}$ is referred to as MBRQT_1.

In this context, note that the concrete setting of the values of parameter c for the MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ parameterizations in combination with the specification of the respective summary-from-summaries calculation processes (introducing target depth td which is calculated as $\lfloor \log_4\ c \rfloor$) has the following consequence: Presumably, the internal nodes' MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summaries will not be as detailed as it was the case for e.g. spatially spread resources in the distributed application scenario. This

[244] Note that in pathological cases, the smallest indexable spatial unit of QTMBR$^{b}_{c,a}$ summaries might also correspond to a point. It is the case when all data points of the summarized node are located at exactly the same position which is simultaneously at an intersection of the occupied cell's quantization raster's hyperplanes.

[245] In view of the small data point capacity of the leaf nodes (5 respectively 25 data points, see setting of the R-tree parameters below) and the spatial distribution of especially the T collection's data points, the size of the smallest indexable spatial unit is of utmost importance.

MBRQT_	1	$\text{MBRQT}^{8,0.001}$
	2	$\text{MBRQT}^{8,1.0E\text{-}5}$
	3	$\text{MBRQT}^{8,1.0E\text{-}7}$
	4	$\text{MBRQT}^{16,0.001}$
	5	$\text{MBRQT}^{16,1.0E\text{-}5}$
	6	$\text{MBRQT}^{16,1.0E\text{-}7}$

QTMBR_	1	$\text{QTMBR}^{8}_{16,1.0E\text{-}5}$
	2	$\text{QTMBR}^{12}_{16,1.0E\text{-}5}$
	3	$\text{QTMBR}^{8}_{32,1.0E\text{-}7}$
	4	$\text{QTMBR}^{8}_{64,1.0E\text{-}7}$

Fig. 93: Listing of the $\text{MBRQT}^{c,a}$ respectively $\text{QTMBR}^{b}_{c,a}$ parameterizations tested in the evaluations of the T and R collections.

is because the maximum depth of their quadtree structures is restricted to td which is e.g. 3 for $c = 64$. On the other hand, it has to be noticed that in particular for the internal nodes directly above leaf node level, the amount of input rectangles ($\rightarrow$ indexed areas of *all* child node summaries) for their summaries' calculation can be quite large. Hence, very detailed internal node summaries could have massive impacts on the corresponding R-trees' construction times—especially since in addition, the existence of the UPDATE-ascension ($\rightarrow$ recalculation of all internal node summaries on the path towards the root) has to be taken into account. In this context, it is important to consider that usually, more than 95% of an R-tree's nodes are leaf nodes. Therefore, it is reasonable to focus on the leaf node access performance as eventual deficiencies there can never be compensated by the internal node access performance. Consequently, with regard to the setup of our evaluation, we think that the given setting of the c parameter values is a fair compromise between a solid spatial accuracy of the internal nodes' $\text{MBRQT}^{c,a}$ and $\text{QTMBR}^{b}_{c,a}$ summaries and reasonable construction times for the corresponding R-trees.
For the R-tree itself, the page size is set to 4,096 B. As we assume a primary index R-tree (i.e. the media objects associated with the spatial data points are administered directly in the leaf nodes), we have to define a payload size for the media objects. For this, we evaluate two different sizes and assess their effects on the results. For convenience, we assume that in the first setting, up to five media items can be administered per leaf node whereas in the second setting, up to 25 media items can be stored per leaf node. Consequently, there are four different basic scenarios for the evaluation:

— **T_5**: T collection and leaf node capacity 5.
— **T_25**: T collection and leaf node capacity 25.
— **R_5** : R collection and leaf node capacity 5.
— **R_25**: R collection and leaf node capacity 25.

For the SELECTBEST(.)-method of INSERT(.), we apply the modification described in section 10.2.1 for all summarization types (i.e. also for the MBR). This is made to ensure the comparability of the results. In addition, the original SELECTBEST(.)-method of Guttman's classical R-tree only consid-

ers the surface area enlargement which might promote the creation of degenerated MBRs (since degenerated MBRs have only very small surface areas). Furthermore, for the R*-tree, Beckmann et al. evaluated adapted SELECTBEST(.)-methods which also take the overlap and the margin into account (in different combinations). The best combination applied Guttman's original, surface-area-enlargement-based strategy for the internal nodes while at leaf node level, it performed an (expensive) overlap minimization strategy. In total, *slight* improvements over Guttman's original strategy resulted [Beckmann et al. 1990, p. 325]. Nevertheless, in their paper on the Revised R*-tree, Beckmann and Seeger describe the R*-tree's SELECTBEST(.)-method (among other points) as problematic with regard to zero volume data [Beckmann and Seeger 2009, p. 802f]. Since we utilize point data and the leaf node capacity of our R-trees is rather small (5 or 25), this is relevant at least with regard to the T collection because there, duplicate data points exist (i.e several data points featuring the exact same coordinates). For the R collection, no duplicates exist but the data points may be arranged in such a way that the data points of the leaf nodes are aligned on a horizontal or vertical line ($\rightarrow$ zero volume MBR). Consequently, for all these reasons, we apply our modified, MBR-center-driven SELECTBEST(.)-method for all summarization approaches and in all our experiments.

As another aspect, it is reasonable to utilize caching mechanisms to increase the general efficiency of the R-tree. Common strategies are to cache the first l levels of the R-tree (i.e. all nodes of these levels including the deserialized summary instances of their child node entries), a fixed amount of nodes (including the deserialized summary instances of their child node entries), or a fixed amount of deserialized summary instances. In our *query experiments*, we apply the latter strategy, i.e. we cache the deserialized summary instances of the 1,000 most recently considered nodes. For keeping track of which summary instance needs to be removed from the cache once it is full, we utilize an FIFO queue of integer values. These integer values are the IDs of the nodes which are described by the cached summary instances. The summary instances themselves are administered as *values* in a hash table using the IDs of the nodes they describe as *keys*. This hash table enables us to efficiently guarantee that no ID is contained more than once in the FIFO queue as well as an efficient access to the deserialized summary instances. For the construction of the R-trees, we do not apply caching since we want to estimate the *worst case* construction time differences for our summarization approaches in comparison to the MBR approach.

In each assessment, we conduct 200 queries. For the query points of the T collection, the first 200 data points of the collection's data point set are taken as query points. The data points which are selected as query points are *not* inserted into the R-tree. Hence, the amount of data points administered in the R-tree is 26,739,554 data points for the T collection. For the query points of the R collection, we simply create 200 additional random

data points. For each collection, the query points are used for both the kNN queries as well as the range queries. See Figure 94 for a visualization of the locations of the `T` collection's (top) and the `R` collection's (bottom) query points.

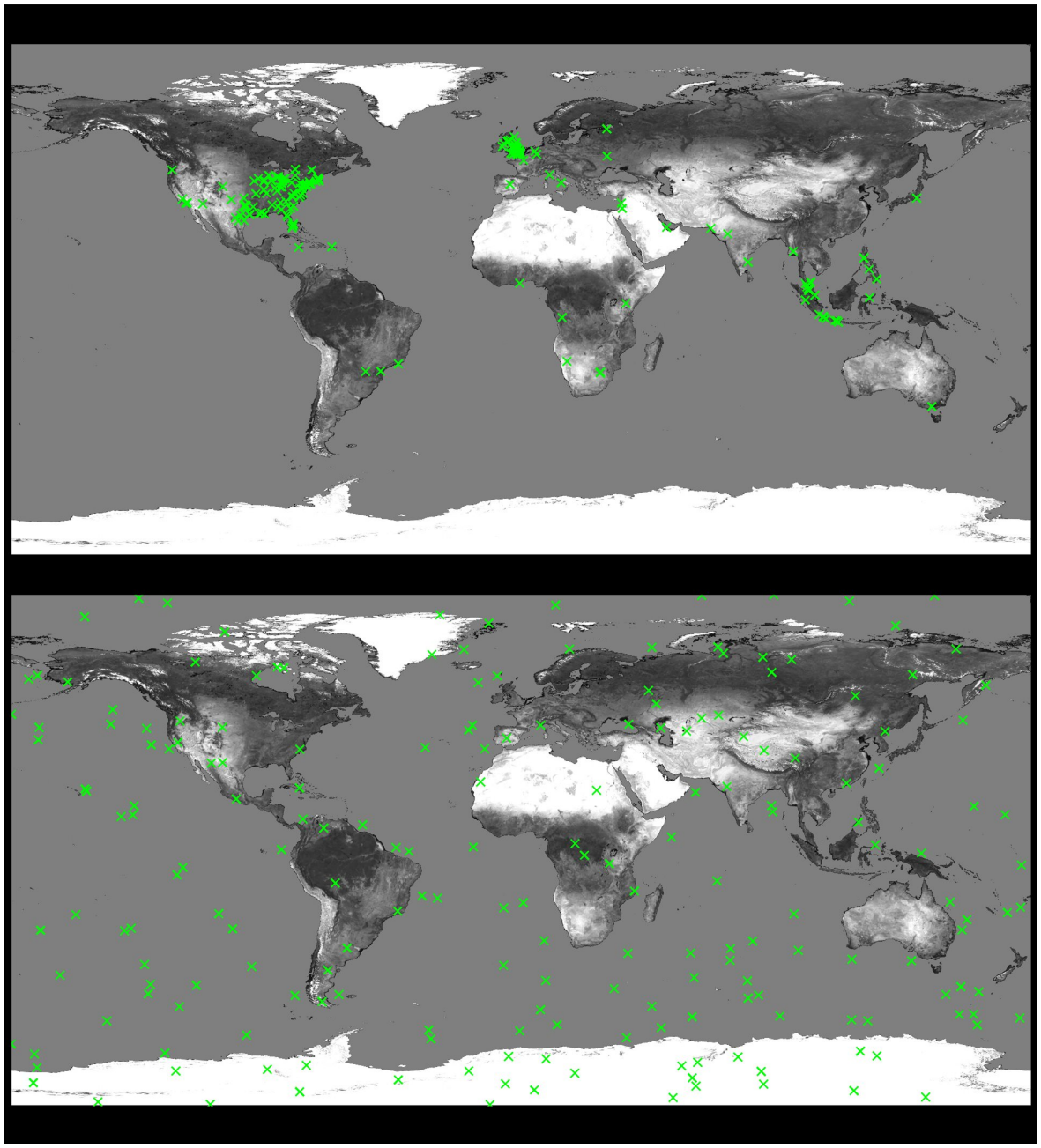


Fig. 94: Locations of the query points for the `T` collection (top) and the `R` collection (bottom). The green crosses depict the locations.

For the kNN queries, we set the values of k to 10 respectively 1,000. For the range queries, we assess two different query radii. They query radii are set that by assuming a regular spatial distribution of the data points, result set sizes of 10 respectively 1,000 data points can be expected. Assume q_{rad} to be the query radius, sa_{ds} to be the surface area of the data space, n_{dc} to be the number of data points in the data collection, and en_{dp} to be the expected

number of data points in the query result. Then, the corresponding query radii can be calculated by the following equation:

$$q_{rad} = \sqrt{\frac{\frac{sa_{ds}}{n_{dc}} \cdot en_{dp}}{\pi}}$$

For example, for the R collection, the query radius for which $en_{dp} = 10$ data points can be expected in the query result is calculated as follows:

$$q_{rad} = \sqrt{\frac{\frac{360\,dsu \cdot 180\,dsu}{25{,}000{,}000} \cdot 10}{\pi}} = \sim 0.0908\,dsu$$

The (rounded) q_{rad} values resulting for the T respectively the R collection and the different values for en_{dp} are depicted in Table 40. For the queries, the *exact* q_{rad} values are used.

Table 40: Query radii (q_{rad}) for the range queries conducted for the T and the R collections. The listed q_{rad} values are rounded to four decimal places.

	T collection	R collection
$en_{dp} = 10$	0.0878 *dsu*	0.0908 *dsu*
$en_{dp} = 1,000$	0.8783 *dsu*	0.9083 *dsu*

13. EVALUATION

In this section, we conduct the evaluation of the R-trees into which the diverse techniques' summaries have been integrated. Specifically, we briefly assess the differences in the respective R-trees' construction times (section 13.1). In section 13.2, the structural properties of the various R-trees are analyzed. Afterwards, the R-trees are evaluated with regard to their ultimate purpose: their query performances (section 13.3). Finally, we once again summarize the results of the entire evaluation, discuss the degree of achievement of thesis objective ❷, and outline starting points for future work (all in section 13.4).

13.1. Assessment of the R-tree Construction Times

In the following, the R-tree construction times are assessed. In each of the four scenarios (→ T_5, T_25, R_5, and R_25), each of the eleven techniques has an R-tree of its own. For each R-tree, we employ the respective techniques' summaries during the entire construction phase. As mentioned in section 12.3, we measure the construction times without applying any caching to *conservatively* assess the temporal overhead which is introduced with the integration of our summarization approaches. This means the basic conditions are set such that the results correspond to a *worst case* estimation of this overhead. In Figure 95, the different techniques' overall construction times are depicted for the four different scenarios.

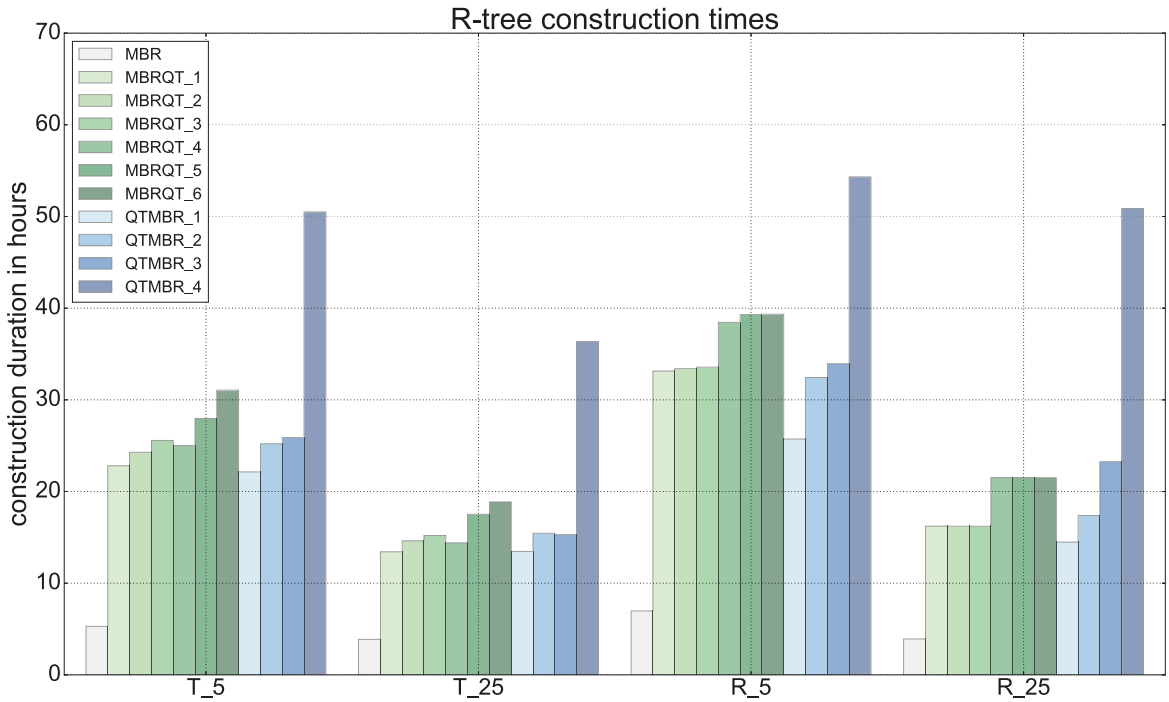


Fig. 95: R-tree construction times for the eleven different techniques in the four different scenarios.

In general, it can be seen that for each technique, their R-tree construction durations are longer in the scenarios with a leaf node capacity of 5

($\rightarrow$ T_5 respectively R_5) than in the respective counterpart scenarios with leaf node capacity 25 ($\rightarrow$ T_25 respectively R_25). For example, the MBR approach's R-tree requires 5.30 hours for its construction in the T_5 scenario but only 3.88 hours in the T_25 scenario. This was to be expected: In the respective R-trees of the T_5 respectively R_5 scenario, due to the lower leaf node capacity of 5, there are more leaf nodes than in the corresponding R-trees of the T_25 respectively R_25 scenario. The greater amount of leaf nodes also leads to a greater amount of internal nodes in the R-trees. Consequently, the overall amount of nodes is significantly greater. Nevertheless, it is not necessarily the case that a greater amount of nodes inevitably has to lead to longer construction times. This is because the insertion of a data point is restricted to a single path in the R-tree. However, in the long run, a greater number of nodes results in R-trees of a greater height (i.e. R-trees with a greater amount of levels) which causes more node accesses on the way to the leaf nodes when inserting a new data point. For example, the MBR approach's R-tree has about 7.72 million nodes and a height of 5 in the T_5 scenario whereas in the T_25 scenario, it is 1.58 million nodes and a height of 4.[246] Moreover, also during the eventually occuring subsequent UPDATE-ascension (in case the inserted data point is not contained in the indexed area(s) of its leaf node's summary), the path to the root is longer. This leads to a) more node accesses and b) more summary-from-summaries calculations. It is further important to consider that an R-tree construction is a continuous process in which all R-trees are initially empty. Hence, the more nodes an R-tree has at the end of the construction phase, also the sooner (i.e. after less inserted data points) its height increases. This means that during the entire construction phase, there are repeated periods of time during which the R-trees of the T_5 respectively the R_5 scenario have greater heights than those of the T_25 respectively R_25 scenario.

Additionally, for R-trees with a greater leaf node capacity, the amount of summary (re)calculations should be significantly lower. It is because the more data points are stored in a leaf node, the greater the spatial spread of these data points is in case similar distributional properties are given. Therefore, the leaf node's summary covers a larger surface area given greater leaf node capacities. This intuition is also confirmed by the numeric results: A look into the results for the cumulated indexed surface areas of the leaf nodes shows that e.g. for the MBR approach's R-tree in the T_5 scenario, this cumulated indexed surface area accounts for 17,569.9 dsu^2 (already see Table 59 on page 352) while for the corresponding R-tree in the T_25 scenario, it sums up to 49,697.9 dsu^2 (see Table 63 on page 359). The greater the covered surface areas, the more likely it is that a newly inserted data point is already contained in one of the indexed area(s) of the leaf node it is assigned to. Hence, the amount of summary recalculations

[246] The structural differences between the respective R-trees are assessed in more detail in section 13.2.

for both leaf nodes as well as the internal nodes on the path towards the root during the subsequent UPDATE-ascension should definitely be lower for the R-trees with a leaf node capacity of 25. Overall, it is easily comprehensible why the construction times of the R-trees with a leaf node capacity of 5 are longer than those of the corresponding R-trees with a leaf node capacity of 25.
For each technique, in the scenarios involving the R collection (R_5 respectively R_25), the R-tree construction durations are longer than for their T collection's counterparts (T_5 respectively T_25) although the T collection features about 1.74 million data points more. For example, the MBR approach's R-tree requires 6.97 hours for its construction in the R_5 scenario (while storing 25 million data points) but only 5.30 hours in the T_5 scenario (while storing 26.7 million data points). The reason for this is that for the T collection, duplicate data points exist and also the spatial distribution of the data points is more erratic. Thus, more newly inserted data points are already contained within the indexed areas of their leaf nodes' summaries. As a consequence, a smaller amount of leaf node summaries has to be recalculated in the T_5 respectively the T_25 scenario as opposed to the R_5 respectively the R_25 scenario which reduces the respective CPU times. These reductions are all the more important as each leaf node summary's recalculation triggers the recalculation of the internal nodes' summaries on the path towards the root. Hence, it is understandable that with regard to the *single techniques*, the R-trees in the R collection's scenarios generally require more time than the corresponding R-trees in the T collection's scenarios.
The analysis so far exclusively included *technique-specific* comparisons, i.e. the statements are valid when comparing e.g. the MBR approach's four R-trees (one for each scenario $\rightarrow$ T_5, T_25, R_5, R_25) to each other. Now, we want to conduct *scenario-specific* comparisons, i.e. e.g. compare the runtimes of the eleven techniques in the T_5 scenario to each other. In general, this assessment is extremely complex because there is a multitude of factors influencing the construction times of an R-tree. Based on the sequence of the insertion process for a new data point, a list of influencing factors has to contain the following aspects:[247]

— **a)** The costs of deserializing the summaries which are considered during an insertion. A deserialization encompasses the reconstruction of the indexed areas from the summaries' respective bit vectors.
— **b)** The costs of the MINDIST calculations during the insertion of a data point dp. In general, these costs are dependent on the amount of indexed areas for which the distance to dp is calculated.
— **c)** The costs of updating a leaf node's summary after inserting dp into the very leaf node. For the MBR approach, the MBR is simply resized to

[247] Note though that this list is not necessarily complete and that there might be additional factors impacting the construction times.

also contain dp. For MBRQT^{c,a} and QTMBR$^{b}_{c,a}$, the summary has to be completely recalculated, considering all data points which are stored in the leaf node.

— **d)** The costs of updating the summaries of the internal nodes which are on the path from the leaf node ln storing dp to the root node in case ln's summary has been updated. For all summarization approaches, the summary-from-summaries calculation processes have to be performed for all the nodes on this path in case ln's summary has been updated.

Obivously, the costs of the deserialization, the summary-from-data-points calculation, and the summary-from-summaries calculation (i.e. the 'elementary operations' of **a)**, **c)**, and **d)**) differ from each other for the various techniques. Furthermore, the aspects **a)** and **b)** are also strongly dependent on the amount of summaries which are considered during the insertion process—which for their part depend on the fanout and the height of the corresponding R-trees. This is due to workflow of the insertion process: At each level of the R-tree, a *selectedNode* is chosen for the insertion of dp (see section 10.1 and algorithm 4). The sequence of *selectedNodes* defines the insertion path. To proceed to the next level, all child nodes of the current *selectedNode* are considered as candidates for the insertion of dp, i.e. to be the *selectedNode* of the next level. Hence, the **a)** number of summaries to deserialize is dependent on the amount of child nodes of the *selectedNodes* (i.e. the *selectedNodes*' fanouts) and the height of the respective R-tree. Furthermore, the **b)** total amount of indexed areas to which the distance to dp is calculated is dependent on the amount of indexed areas per summary, the amount of child nodes of the *selectedNodes*, and the height of the respective R-tree. Moreover, also factor **d)** depends on the height of the respective R-tree. As our corresponding structural analysis will show (see section 13.2.2), the structural properties (the height, the number of nodes, and the fanout) of the respective R-trees differ very strongly in some cases. In general, these structural properties depend to a large extent on the memory utilization rates of the leaf nodes and the internal nodes. For the memory utilization rates of the leaf nodes, the suitable assignment of the data points to the leaf nodes is of relevance. The assignment of data points to nodes is handled by the SELECTBEST(.)-method. The SELECTBEST(.)-method utilizes the nodes' summaries to determine the 'most appropriate' child node for the newly inserted data point dp. Since the properties of the MBR, the MBRQT^{c,a}, and the QTMBR$^{b}_{c,a}$ summaries differ from each other but simultaneously, the same SELECTBEST(.)-method is utilized for all approaches, the suitability of the SELECTBEST(.)-method might differ for the

[248] In principle, also the split algorithm is certainly a factor with some influence on the memory utilization rates of internal and leaf nodes. However, for all R-trees, the R*-split is utilized. The R*-split does not work directly on the summaries but utilizes the *MBRs of the respective indexed areas*. Therefore, its impact on the results should be fairly similar for the techniques' respective R-trees.

various approaches. For the memory utilization rates of the internal nodes, the sizes of the summaries are relevant in addition (although more as an indirect factor).[248] As a related aspect, also the number of node splits has an impact on the construction times. Obviously, the greater the number of nodes in an R-tree, the greater the number of node splits that have been conducted. A split introduces overhead due to the application of the split algorithm and the subsequent (re-)calculation of summaries. Furthermore, during the growth periods of the R-trees, their structural properties may change and vary. Moreover, it is conceivable that our implementation still might require some optimizations: For example, specific methods might be implemented suboptimally, or inappropriate data structures might have been utilized. And so on.

In a nutshell, there are too many influencing factors for an *analytical* assessment and comparison of the construction times of the respective techniques' R-trees. Therefore, in the following, we confine to a brief discussion of the apparent differences between the techniques' results but do not attempt to analyze the reasons behind the observations.

In a comparison of the different techniques, it is evident that in all scenarios, the MBR approach requires the shortest construction times by far. This is within the expectations since the MBR approach is the simplest summarization approach in every respect (summary calculation itself, deserialization, etc.). Overall, the R-tree construction with the MBR approach is at least 3.5 times faster than with any other technique—in any scenario. Also see Table 41 where the relative overall construction times are showcased as percentage values (with the times of the MBR approach corresponding to 100% in each row). Specifically, Table 41 not only depicts a) the relative overall construction times (top of each cell) but also b) the relative average insertion times for the last 100 data points (bottom of each cell). When comparing the relative values of the overall times with those of the last 100 insertions, in most cases, the relative values of the latter are greater. For example, in the T_5 scenario, the R-tree construction with MBRQT_4 requires 472% of the overall construction time with the MBR approach but for the last 100 data point insertions, MBRQT_4 requires 543% of the MBR approach's time. Consequently, it is evident that the relative advantage of the MBR approach increases the greater the amount of data points is which have already been inserted into the R-trees.

In a benchmarking between the overall construction times of the $\text{MBRQT}^{c,a}$ approach's techniques and the $\text{QTMBR}^{b}_{c,a}$ approach's techniques, it shows that much depends on the respective parameterizations. The highest $\text{QTMBR}^{b}_{c,a}$ parameterization (QTMBR_4) is steadily the most time-consuming technique of all. Nevertheless, the lower parameterizations of $\text{QTMBR}^{b}_{c,a}$ are equally fast as or partially even faster than the lower $\text{MBRQT}^{c,a}$ parameterizations. The results show that by utilizing appropriate parameterizations, the R-tree construction with $\text{QTMBR}^{b}_{c,a}$ summaries can be similarly fast as the construction with $\text{MBRQT}^{c,a}$ summaries. We do

Table 41: Comparison of the relative R-tree construction times, percentage specification. The construction times of the MBR approach are the benchmark and correspond to 100% in each row. At the top of each cell, the relative *overall construction times* are listed. At the bottom of each cell, the relative *average insertion times* for the last 100 data points are listed.

	MBR	MBRQT_1	MBRQT_2	MBRQT_3	MBRQT_4	MBRQT_5	MBRQT_6	QTMBR_1	QTMBR_2	QTMBR_3	QTMBR_4
R_5	100	476	479	482	552	564	565	369	466	487	779
[in %]	100	484	491	479	530	578	559	403	525	541	811
R_25	100	414	414	414	549	549	549	370	444	593	1,298
[in %]	100	439	481	462	636	616	581	361	422	550	1,397
T_5	100	430	458	483	472	528	586	418	476	488	953
[in %]	100	459	498	530	543	522	635	473	577	586	1,053
T_25	100	346	377	392	371	451	486	347	398	394	937
[in %]	100	385	421	418	395	508	538	378	444	413	1,055

not go further into detail at this point for the *inter*-approach comparison of the results but conduct *intra*-approach comparisons for MBRQT^{c,a} and QTMBR$^{b}_{c,a}$.

Regarding the different MBRQT^{c,a} parameterizations, it is evident that the parameterizations MBRQT_1 to MBRQT_3 require less time than the parameterizations MBRQT_4 to MBRQT_6. For the scenarios involving the R collection, the differences between those two groups are very evident. *Within* in each of the groups, the construction times are almost identical in the R_5 scenario as well as in the R_25 scenario. For MBRQT_1 to MBRQT_3, it applies that $c = 8$ while for MBRQT_4 to MBRQT_6, it applies that $c = 16$. Hence, parameter c is obviously the predominant influencing factor for the construction times while parameter a is only of secondary importance. For the T_5 scenario as well as the T_25 scenario, the differences between both groups are not as stepped and clear anymore. This means that besides parameter c, also parameter a now obviously has a significant impact on the resulting construction times. Nevertheless, the general tendency (greater impact of parameter c) is confirmed. In real-world data sets (like the T collection), the consequences of the inherent differences between the parameterizations are generally more diffuse due to the non-regularity of the data point distribution.

For the QTMBR$^{b}_{c,a}$ parameterizations, the results show that the construction times are by far the largest for QTMBR_4. The other three parameterizations are always relatively close together. Hereby, QTMBR_1 generally and distinctively requires the shortest construction times while QTMBR_2

and QTMBR_3 are the closest matches. This is surprising as the only difference in the parameterizations of QTMBR_1 ($b = 8$) and QTMBR_2 ($b = 12$) is the quantization accuracy. Nevertheless, already this change can have significant impacts on the respective R-trees' structures. Overall, aside from the mentioned observations, no clear patterns are apparent from the intra-approach comparison of the $\text{QTMBR}^{b}_{c,a}$ parameterizations' results.
As outlined before, we refrain from a theoretical analysis of the reasons behind these results due to the unmanagable complexity of influencing factors. The only reasonable assessment which can be conducted under the given circumstances is an exemplary profiling[249] to identify eventual bottlenecks in the respective techniques' R-tree construction processes via empirical evidence. We perform our exemplary profiling for a set of selected R-trees in the T_5 scenario. The corresponding techniques are the MBR approach, MBRQT_1, and three different $\text{QTMBR}^{b}_{c,a}$ parameterizations (due to the greater intra-approach disparities of their results): QTMBR_1, QTMBR_2, and QTMBR_4. Within the scope of this profiling, the `T` collection's first 100 *query points* are inserted into the respective techniques' R-trees as additional data points.
For these insertions, in Table 42, the time-dominant methods alongside their most time-consuming subroutines are listed for each of these five techniques. In general, the three time-dominant methods for the insertion of the 100 additional data points are:

— `contains`: Checks whether a newly inserted data point dp is contained within the indexed area(s) of a summary. For all techniques, the subroutine of `contains` which consumes the majority of the elapsed CPU time is the deserialization process.
— `getMidPoint`: Determines the center point of a summary. This method is required to resolve *ambiguous* or *inconclusive* scans in the SELECTBEST(.)-method (see section 10.2.1). For all techniques, the subroutine of `getMidPoint` which consumes the majority of its CPU time is the deserialization process.
— `updateSummaryAfterSplit` (`updateSumASpl`): After a node has been split, the summaries of the two resulting new nodes *and* of all the parent nodes on the path towards the root have to be recalculated. For all techniques, almost the entire CPU time of `updateSumASpl` is taken up by the subroutine `createSummaryBySummaries` (`createSumBySums`) which performs the summary-from-summaries calculation. For `createSumBySums`, the two most time-consuming subroutines are `getBoundingBoxfromSummaries` (`getBBfSums`) and `colorLeafNodes`

[249] The profiling was conducted with JProfiler version 10.1.2 of ej-technologies GmbH (https://www.ej-technologies.com/products/jprofiler/overview.html, last visit: 26.10.2018).

(colorLN).[250] For the former, again, it is the deserialization process which takes up almost the entire CPU time for all techniques.[251]

Hence, it is evident that one specific property is very stable for all selected example techniques: Obviously, the deserialization procedure is a general bottleneck in our implementation. Within a deserialization process, the indexed areas are reconstructed from the summary's corresponding bit vector. As can be seen from Table 42, the amounts of method invocations of the deserialization subroutine are very large, in general. As outlined at the beginning of section 13.1, no caching is applied respectively the cache size is set to 1 (i.e. only the reconstructed indexed areas of *one* node's summary are held in cache). Hence, almost each method invocation which requires the spatial information of another node than that for which the summary is currently cached triggers a deserialization of this other node's associated summary (which is then put into the cache). This happens fairly often during the R-tree construction process, e.g. when traversing the paths to the leaf node level during the INSERT(.)-method (where at each level, all child nodes of *selectedNode* are assessed as candidates to store the newly inserted data point on the basis of their summaries). For the contains- and the getMidPoint-methods, the need for the respective nodes' spatial information is self-evident. For the summary-from-summaries calculation of a node N, the indexed areas of N's child nodes are collected and used as input for the calculation. As a consequence, also the createSumBySums-method triggers a great amount of deserializations.
More specifically, the bottleneck of our current deserialization implementation is the Java class BitArray which is used to represent the summaries' bit vectors. The BitArray class does not offer a method for returning a subset of a given BitArray instance—which is an operation we require fairly often during the deserialization process. Therefore, we had to implement a custom method for getting a subset (specified by a start and an end index) of a BitArray instance. In this implementation, we iterate *bit by bit* over the specified range of the BitArray instance to acquire the required subset. Obviously, this is suboptimal with regard to runtime. Due to the great amount of invocations of the deserialization-method, the total CPU time consumed by the deserialization process accumulates heavily.
In general, the total construction times displayed in Figure 95 seem to correlate *roughly* with the number of internal nodes in the R-trees, i.e. the greater the number of internal nodes, the greater the total construction

[250] Naturally, the MBR approach does not have a colorLN-method as its summary does not make use of quadtree structures.

[251] Note that as single exception from this, for MBRQT_1, about half of the CPU time spent for createSumBySums is consumed by the getBoundingBox-method which reconstructs the basic MBR from a *single* MBRQT^{c,a} summary instance (in contrast, the input parameter of the getBBfSums-method is a *set* of summaries). Nevertheless, also for getBoundingBox, the subroutine consuming almost the entire CPU time is the deserialization process.

Table 42: Result table of the examplarily conducted profiling for the insertion of the 100 additional data points into different R-trees of the T_5 scenario. Each time-critical method has its own row. In each cell except those of the 'total time'-row, the total CPU time of the row's method (big number at the top of the cell), its share of the *total time* (small, bracketed number in the middle), and the total number of method invocations (small number at the bottom) are displayed.

	QTMBR_1	QTMBR_2	QTMBR_4	MBR	MBRQT_1
total time	**30,781** *ms*	**28,739** *ms*	**68,755** *ms*	**8,030** *ms*	**27,879** *ms*
contains	17,260 *ms* (56.0%) 51,311	17,321 *ms* (59.9%) 45,783	20,623 *ms* (30.0%) 40,121	4,763 *ms* (59.1%) 37,290	13,855 *ms* (49.6%) 28,619
↪ deserialize	16,760 *ms* (54.4%) 51,311	16,880 *ms* (58.3%) 45,783	20,174 *ms* (29.3%) 40,121	4,374 *ms* (54.3%) 37,290	15,581 *ms* (48.7%) 28,619
getMidPoint	2,848 *ms* (9.2%) 16,248	1,319 *ms* (4.6%) 5,534	8,032 *ms* (11.7%) 19,824	610 *ms* (7.6%) 8,882	1,287 *ms* (4.6%) 5,716
↪ deserialize	2,075 *ms* (8,7%) 8,124	1,255 *ms* (4.3%) 2,767	7,785 *ms* (11.3%) 9,912	428 *ms* (5.3%) 4,441	1,282 *ms* (4.6%) 2,858
updateSumASpl	10,057 *ms* (32.6%) 180	9,448 *ms* (32.7%) 177	38,926 *ms* (56.6%) 408	2,087 *ms* (25.9%) 180	12,266 *ms* (44.0%) 220
↪ createSumBySums	10,033 *ms* (32.6%) 180	9,424 *ms* (32.6%) 133	38,894 *ms* (56.5%) 255	2,030 *ms* (25.2%) 135	5,667 *ms* (20.3%) 165
↪ getBBfSums	7,023 *ms* (22.8%) 135	6,831 *ms* (23.6%) 133	15,869 *ms* (23.1%) 255	1,969 *ms* (24.4%) 135	5,383 *ms* (19.3%) 138
↪ deserialize	6,943 *ms* (22.5%) 23,039	6,764 *ms* (23.4%) 19,939	15,748 *ms* (22.9%) 33,634	1,887 *ms* (23.4%) 16,922	5,342 *ms* (19.1%) 12,773
↪ colorLN	2,634 *ms* (8.5%) 135	2,108 *ms* (7.3%) 133	20,324 *ms* (29.5%) 255	-	62 *ms* (0.2%) 138
↪ getBoundingBox	-	-	-	-	6,548 *ms* (23.5%) 15,004
↪ deserialize	-	-	-	-	6,495 *ms* (23.3%) 15,004

[252] Already see Table 56 on page 346 (T_5 scenario), Table 61 on page 357 (T_25 scenario), and Table 66 (R_5 scenario) on page 362—all in section 13.2.2—for the tables depicting the amounts of internal nodes for the respective R-trees in the diverse scenarios (for the R_25 scenario, there is no corresponding table in this work).

times.[252] This is reasonable: the insertions are restricted to a single path in the R-tree and therefore, the height of the R-tree is of great impact. R-trees with a greater number of internal nodes grow faster, i.e. there are periods during the insertion process in which they have an extra level compared to R-trees with a lower final number of internal nodes (which in the very end triggers more deserialization processes). These periods might also explain why the overall construction times of the R-trees of QTMBR_1 are shorter than those of the R-trees of QTMBR_2—despite the fact that for our exemplary 100 additional insertions into the constructed R-trees, the duration is *shorter* for QTMBR_2 than for QTMBR_1 (28,739 ms as opposed to 30,781 ms). This might be because the height of the respective R-trees is the same during these 100 insertions but due to the lower fanout of the QTMBR_2 R-tree[253], the number of summaries to deserialize per level is lower for QTMBR_2. In any case, the numbers listed in Table 42 show that the amount of invocations of the deserialization-method is clearly lower for QTMBR_2 than for QTMBR_1. Overall, the divergence between the *total construction times* ($\rightarrow$ QTMBR_1 $<$ QTMBR_2) and our *100 exemplary insertions* ($\rightarrow$ QTMBR_1 $>$ QTMBR_2) is therefore explainable, especially since for the 100 insertions, the differences between QTMBR_1 and QTMBR_2 are so small that it might also be noise.
We do not want to overanalyze the results displayed in Table 42 at this point since it is clear that an optimization of the deserialization process (which includes a replacement of the `BitArray` class) is the first starting point for future work. Nevertheless, we would like to point out some additional aspects which are apparent from Table 42:

— The coloring of the initial quadtree structure's leaf nodes for the summary-from-summaries calculation of $\text{QTMBR}^{b}_{c,a}$ is significantly more expensive than for $\text{MBRQT}^{c,a}$. This is plausible as the consideration of a cell of the initial quadtree can be aborted for $\text{MBRQT}^{c,a}$ after it is intersected by *any* of the input areas (and therefore is colored black), see section 10.4.1. For $\text{QTMBR}^{b}_{c,a}$, the overlap area between each occupied quadtree cell and each input area must be determined explicitly as from these overlap areas, the quantized MBR of this quadtree cell is calculated (see section 10.4.2). Our corresponding implementation does not use any kind of pruning mechanism, i.e. for each quadtree cell, it is iterated over the entire set of input rectangles to determine the overlap areas. Hence, there is no possibility of a premature abortion at any stage. The runtime measurements indicate that this is an implementation which might require further optimizations.
— With a greater spatial accuracy of the $\text{QTMBR}^{b}_{c,a}$ summaries, the costs of the summary-from-summaries calculation grow rapidly for $\text{QTMBR}^{b}_{c,a}$— both absolutely as well as in relation to the other time-dominant oper-

[253] In the T_5 scenario, the fanout of the R-tree of QTMBR_2 is 115.4 whereas for the R-tree of QTMBR_1, it is 151.1.

ations such as `contains`. As can be seen from Table 42, for example, the CPU time spent for `colorLN` takes up 8.5% of the total time for QTMBR_1 but 29.5% of the total time for QTMBR_4. This is easily explainable: The number of cells of the initial quadtree for the summary-from-summaries calculation is dependent on target depth td which is defined as $\lfloor \log_4 c \rfloor$. As it applies that $td = 2$ for QTMBR_1 (with $c = 16$) but $td = 3$ for QTMBR_4 (with $c = 64$), the number of cells in the initial quadtree is $4^2 = 16$ for QTMBR_1 but $4^3 = 64$ for QTMBR_4. Due to the implementation of the `colorLN`-method (which does not use any pruning), it is easily understandable why the relative significance of this method is far more pronounced for the total runtime of QTMBR_4.

— A greater spatial accuracy ($\rightarrow$ greater summary sizes) leads to a lower fanout and therefore ultimately results in more node splits. Consequently, a greater amount of internal node summaries must be (re-)calculated due to the splits. As an example, compare the amount of invocations of the `updateSumASplit`-method between QTMBR_1 (180, fanout 151.1) and QTMBR_4 (408, fanout 61.3).

— The summary-from-data-points calculation seems to be sufficiently fast. Hence, it never emerges as one the procedures which take up significant amounts of CPU time. For example, the 51 method invocations for QTMBR_4 take up only 340 ms or 0.6% of its total CPU time for the 100 additional insertions.

At this point, we conclude the evaluation of the R-trees' construction times. It has been shown that when comparing the single techniques in a specific scenario, such an evaluation is extremely complex due to the plethora of influencing factors and their complex interplay. This is especially the case when the R-trees to compare differ greatly in their structures. However, with the cache size set to 1 and the identified bottleneck in form of the deserialization routine, the measured construction times should be a fair reproduction of the *worst case* construction time overhead for our summarization approaches in comparison to the MBR approach. The suboptimal deserialization implementation strongly favors the MBR approach: Due to its very simple bit vector structure, less subsets need to be extracted from the overall bit vector during the deserialization process of an MBR summary in comparison to MBRQT^{c,a} or QTMBR$^{b}_{c,a}$ summaries. By utilizing appropriate caching strategies and an improved implementation of the bit vectors, it should be possible to reduce the temporal overhead of our summarization approaches. This is future work, though. The empirically determined worst case construction times for R-trees using our tested MBRQT^{c,a} or QTMBR$^{b}_{c,a}$ parameterizations are between ~350% and ~1,300% of the R-trees utilizing the MBR approach. Furthermore, by means of the exemplary insertion of the 100 additional data points, it has been shown that the differences between the MBR approach and our summarization approaches are getting larger the more data points are already stored in the R-trees.

13.2. Assessment of the Structural Differences

In the following, we evaluate the structural differences between the R-trees which result for the different techniques. Hereby, we assess both 'MBR-like' and 'summary-like' R-trees. For the former, we take the R-tree which has been constructed by using MBR summaries. Then, the MBR summaries are replaced with the summaries of the respective other techniques. For the latter, the respective R-trees have been entirely constructed by utilizing the respective techniques' summaries, i.e. they are the R-trees for which the construction times have been assessed in section 13.1.

13.2.1. MBR-LIKE R-TREES. We start with the evaluation of the MBR-like R-trees. For this, in each scenario, we build an R-tree by utilizing MBR summaries. After the construction, the MBR summaries are replaced with the summaries of the other techniques without considering any storage space limitations. Hence, the basic R-tree structure (with its assignment of child nodes to parent nodes and data points to leaf nodes) is exactly the same, and only the summaries differ between the different MBR-like R-trees. We examine the MBR-like R-trees to assess the extent to which the utilization of our summarization approaches can be advantageous *at all*.

Structural Analysis for the MBR-like R-trees in the T_5 scenario

For the MBR-like R-trees, we first evaluate the T_5 scenario as this is the scenario fitting a 'real-world application' of a primary index R-tree the most. Table 43 shows the structural information on the basic R-tree which has been built by using MBR summaries. The basic R-tree has a height of 5 and 7.72 million nodes from which 66,169 are internal nodes, i.e. about 99.14% of the R-tree's nodes are leaf nodes. Hence, the R-tree is very flat and wide which is its typical phenotype. The memory utilization of both internal (68.4%) and leaf nodes (69.9%) is close to the usual 69% memory utilization of bucket-based access structures.[254] The fanout of the R-tree is 116.7, i.e. an internal node has 116.7 child nodes on average.

Table 43: Structural information on the basic R-tree in the T_5 scenario.

	height	# nodes [in M]	# int. nodes	avg mem. utilization int. nodes	# leaf nodes [in M]	avg mem. utilization leaf nodes	avg fanout
basic R-tree	5	7.72	66,169	68.4%	7.65	69.9%	116.7

[254] For 'well-functioning' bucket-based data structures which split an overflowing bucket or node into two, the resulting average memory utilization is always about $ln\,2$ or 69%. This applies to e.g. multidimensional data structures such as the R-tree as well as to classical one-dimensional access structures such as the B-tree. With complementary strategies as for example the forced reinsertion of the R*-tree, the memory utilization rates can oftentimes be further improved.

In Table 44, the MBR-summary-replacement's impacts on the internal nodes' memory consumption are shown. For *all* MBRQT^{c,a} parameterizations, the average memory consumption of the internal nodes is increased by 25%. Due to the byte- and word-alignment, for *each* MBRQT^{c,a} summary (regardless of the parameterization), 24 B are allocated on disk. Hence, a node entry in an MBRQT^{c,a} R-tree[255] requires a constant $24\,B$ (for the MBRQT^{c,a} summary) $+\,8\,B$ (for the pointer) $=\ 32\,B$ instead of $16\,B + 8\,B = 24\,B$ as in the MBR R-tree. As a consequence, the 'true' storage space capacity of 4,096 B per internal node is fairly often surpassed with the MBRQT^{c,a} summaries: The 75%-quantile of the node sizes is already significantly greater (4,384 B). In contrast, the lowest QTMBR$^{b}_{c,a}$ parameterization (QTMBR_1) remains well below the storage space requirements of the MBR approach as solely 1,867.7 B are consumed on average. Also, the maximum memory consumption is only 2,888 B. For QTMBR_2 and QTMBR_3, the average memory consumption is very similar to the MBR approach. In rare cases, however, the maximum capacity of 4,096 B is moderately exceeded (see the 'max'-column of Table 44). QTMBR_4 has the greatest storage space requirements and consumes more than 4,096 B *on average*. Nevertheless, the median memory consumption is slightly below the available 4,096 B, i.e. more than half of the nodes would still fit into a 'real' R-tree. In total, the memory consumption analysis for the internal nodes shows that a lot of the nodes could be kept without further operations when the MBR summaries are replaced by MBRQT^{c,a} respectively QTMBR$^{b}_{c,a}$ summaries. Consequently, in case the construction times are too long (or if other problems arise) when utilizing our summarization approaches, R-trees could also be built with use of the MBR approach, first. Then, after the R-tree has been built, the MBR summaries are replaced with the corresponding summaries. Finally, eventually occuring overfull nodes have to be split. This is a comparatively inexpensive operation.
In the rightmost column of Table 44, the respective amounts of indexed areas are listed. While the 'lower' parameterizations of MBRQT^{c,a} and especially QTMBR$^{b}_{c,a}$ index only rather few more areas than the MBR approach, the 'higher' parameterizations of both MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ describe the nodes with significantly greater amounts of areas. In particular, this applies to MBRQT_3 (+56.9% indexed areas in comparison to the MBR approach), MBRQT_6 (+71.9%), and QTMBR_4 (+101.7%).
Table 45 outlines the summary-related results. The second column shows the average memory consumption of the byte- and word-aligned summaries, i.e. how much storage space the specific summaries take up on disk on average. For the MBR approach, it is a constant 16 B. As mentioned before, all MBRQT^{c,a} summaries (regardless of the parameterization) consume exactly 24 B. The QTMBR$^{b}_{c,a}$ parameterizations exhibit

[255] With MBRQT^{c,a} R-tree, we mean an R-tree in which the nodes are described by MBRQT^{c,a} summaries. For the other approaches and also for the *techniques*, the denomination is analogous (e.g. MBRQT_4 R-tree).

Table 44: Memory consumption of the internal nodes in the MBR-like R-trees in the T_5 scenario. Additionally, the respective amounts of indexed areas are listed.

memory consumption internal nodes → technique ↓	avg [in B]	median [in B]	75%-quant. [in B]	max [in B]	number of indexed areas
MBR	2,800.3	2,736	3,288	4,080	7,720,601
MBRQT_1	3,733.8	3,648	4,384	5,440	7,833,075
MBRQT_2	3,733.8	3,648	4,384	5,440	8,918,120
MBRQT_3	3,733.8	3,648	4,384	5,440	12,115,991
MBRQT_4	3,733.8	3,648	4,384	5,440	8,016,731
MBRQT_5	3,733.8	3,648	4,384	5,440	9,382,939
MBRQT_6	3,733.8	3,648	4,384	5,440	13,269,197
QTMBR_1	1,867.7	1,824	2,192	2,888	7,727,228
QTMBR_2	2,793.0	2,736	3,288	4,112	7,726,814
QTMBR_3	2,806.3	2,744	3,296	4,304	7,849,230
QTMBR_4	4,151.3	4,072	4,904	7,168	15,573,029

more variance. Whereas for QTMBR_1, exactly 8 B are constantly consumed for all summaries, variable values emerge for the other three parameterizations—as can be deduced from the average values. The average memory consumption of both QTMBR_2 (15.9 B) and QTMBR_3 (16.1 B) is very similar to that of the MBR approach. QTMBR_4, on the other hand, requires considerably more (27.6 B). Naturally, due to the identical structure of the MBR-like R-trees, the average memory consumption of the summaries goes hand in hand with the average memory consumption of the nodes.[256]

The third column of Table 45 ('average memory utilization') shows to what extent the available memory is utilized by the summaries, i.e. to what percentage the byte- and word-aligned storage space footprints on disk are actual summary data. The remainder is filling data added by the bit and byte stuffing. As an example, assume that a summary requires 303 bits to encode its information. 303 bits correspond to 37.875 B. Via bit stuffing, the summary is then aligned to 38 B and via byte stuffing, it is aligned to 40 B—which is the overall storage space footprint on disk for this summary. Therefore, the memory utilization rate of this summary would be $(303/8)\,B/40\,B = 94.7\%$. In general, the results show that the MBRQT^{c,a} summaries make only poor use of the additional storage space available. As $66.\overline{6}\%$ of their 24 B storage space footprints are already occupied by the basic full-precision MBRs (16 B), only the remainder is utilized for the re-

[256]Nevertheless, the relative results between the techniques are not identical for both aspects. This is because for the memory consumption of the nodes, also the pointers of the child node entries have to be considered. For example, on summary-level, QTMBR_4 consumes 72.5% more memory than the MBR approach while on node-level, it is only 48.2%.

Table 45: Summary-related results for the MBR-like R-trees in the T_5 scenario.

technique	avg memory consumption [in B]	avg memory utilization	cblq-nz	cblq-z	sum-nz/ lq-nz	sum-z/ lq-z
MBR	16.0	100%	-	-	100%	-
MBRQT_1	24.0	67.8%	99.5%	-	0.49%	-
MBRQT_2	24.0	68.6%	99.2%	-	0.80%	-
MBRQT_3	24.0	71.1%	99.2%	-	0.80%	-
MBRQT_4	24.0	67.9%	99.3%	-	0.74%	-
MBRQT_5	24.0	69.5%	92.4%	-	7.6%	-
MBRQT_6	24.0	74.0%	71.5%	-	28.5%	-
QTMBR_1	8.0	88.9%	0.01%	-	100.0%	-
QTMBR_2	15.9	57.2%	0.01%	-	100.0%	-
QTMBR_3	16.1	58.2%	0.05%	-	100.0%	-
QTMBR_4	27.6	86.0%	20.8%	$\ll 0.01\%$	79.2%	-

fining quadtree data, i.e. the metadata component and the linearly encoded quadtree structure (see Figure 89 on page 291). The best memory utilization rate is achieved by MBRQT_6 with a value of 74.0%, i.e. 1.76 B of the additional 8 B are used.[257] Thus, the strategy to specify the MBRQT^{c,a} parameterizations in such a way that the MBRQT^{c,a} summary sizes remain close to those of the MBR approach is not very successful with regard to the summaries' memory utilization rates. For QTMBR_1 and QTMBR_4, satisfactory utilization rates of more than 85% are achieved. In contrast, both QTMBR_2 and QTMBR_3 use only little more than half of the allocated storage space. For further studies, it would certainly make sense to make sure that better memory utilization rates are achieved for our summarization approaches. This can be done by either setting the parameters accordingly or by adjusting the summary calculation processes.
With regard to the summary description types (depicted in columns 4 to 7 in Table 45), it shows that for all techniques, zipping is basically never applied. This is no surprise as it was already evident in the evaluation of the distributed application scenario that both MBR data and quadtree data are of high entropy. Consequently and regardless of the respective parameterization, for future work, the zipping options can easily be omitted in the MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summary calculation processes, saving CPU time.[258]

[257] The 1.76 B result as $(74.0\% \cdot 24\,B) - 16\,B = 1.76\,B$.

[258] Therefore, whenever we refer to the CBLQ or the LQ encoding option in the following, we mean their non-zipped variants, i.e. `cblq-nz` respectively `lq-nz`.

In view of the linear quadtree encoding options (LQ and CBLQ), it shows that for most of the MBRQT^{c,a} respectively QTMBR$^{b}_{c,a}$ parameterizations, one of the options is dominant. For MBRQT^{c,a}, it is mostly the CBLQ code whereas for QTMBR$^{b}_{c,a}$, it is mostly the LQ code. As more than 99% of the nodes in the MBR-like R-trees are leaf nodes, the summary-related results are heavily dominated by the leaf node summaries.[259] In general, the results confirm the claim that the LQ code is beneficial for quadtrees with few black cells ($\rightarrow$ QTMBR$^{b}_{c,a}$ R-trees) whereas the CBLQ code is beneficial for quadtrees with many black cells ($\rightarrow$ MBRQT^{c,a} R-trees). For the QTMBR$^{b}_{c,a}$ summaries, it is as follows: The leaf nodes have a capacity of only 5, i.e. a leaf node's data points are mostly located in spatially very narrow regions of the data space (because the SELECTBEST$(.)$-method and the R*-split assign 'neighboring' data points to the same leaf nodes). Since QTMBR$^{b}_{c,a}$ needs to partition the entire data space, mostly only a single black quadtree cell results for the summary of a leaf node as either the maximum number of quadtree cells c is depleted or the threshold surface area a is undercut before the basic quadtree can build more black cells. This is confirmed by the number of indexed areas for QTMBR_1 to QTMBR_3 in Table 44: they only feature marginally greater numbers in comparison to the MBR approach. The only exception is QTMBR_4 which is parameterized in such a way that fairly often, a sufficient amount of black quadtree cells is built so that the CBLQ code becomes favorable (see the 20.8%-share of CBLQ-encoded summaries for QTMBR_4). Consequently, QTMBR_4 also has more than twice as many indexed areas as the MBR approach (see Table 44).

For MBRQT^{c,a} summaries, in general, one can expect rather flat quadtree structures which tend to have more black cells because the refining quadtrees only have to partition the space within the basic MBRs. Therefore, the number of quadtree cells does not deplete prematurely like for QTMBR$^{b}_{c,a}$. Furthermore, if an MBR is refined, at least two black cells are guaranteed for the refining quadtree as at least two opposing quadrants of the MBR must contain data points.[260] Hence, a greater average amount of black cells per quadtree can be expected compared to the QTMBR$^{b}_{c,a}$ summaries. The CBLQ shares of the results seem to confirm these theoretical reflections. Nevertheless, it *only seems like* the CBLQ option is favorable for MBRQT^{c,a}: It has to be considered that in our data, the CBLQ encoding is captured as the standard encoding. This means that in case *no internal quadtree is built* because the basic MBR's surface area is already below the

[259] Note that this is the case for all subsequent analyses of the summary-related results.

[260] The exception is of course when a refining quadree is condensed to a single black cell because all initially built quadtree cells are black. However, given the erratic data distribution of the T collection, it stands to reason that this is not the standard case for the *leaf nodes*, especially when considering that only $69.9\% \cdot 5 = 3.5$ data points are stored in a leaf node on average and that parameter c is set 8 or 16 for the MBRQT^{c,a} parameterizations.

threshold surface area a, the metadata component is encoded exactly as for the CBLQ code (with '00', see footnote[239] on page 290). As just mentioned, in general, if a basic MBR is refined by a quadtree, at least two indexed areas result. However, the numbers of indexed areas in Table 44 show that for some MBRQT^{c,a} parameterizations, there are hardly more indexed areas than nodes in the basic R-tree, i.e. almost no nodes are described by more than one area. For example, for MBRQT_1, the number of indexed areas is only 1.46% greater than the number of nodes in the basic R-tree. Thus, the consideration of CBLQ as standard encoding must have a significant impact on the results. In general, the number of non-refined MBRs is dependent on parameter a. A look into our data shows that for the respective MBRQT^{c,a} parameterizations, the following amounts of nodes are described by an MBR whose surface area is smaller than a (i.e. the nodes are described by a non-refined MBR):

— MBRQT_1 and MBRQT_4 (for which $a = 0.001\ dsu^2$): 7,610,710 nodes (98.6% of the nodes in the basic R-tree)
— MBRQT_2 and MBRQT_5 (for which $a = 1.0E{-}5\ dsu^2$): 6,921,083 nodes (89.6% of the nodes in the basic R-tree)
— MBRQT_3 and MBRQT_6 (for which $a = 1.0E{-}7\ dsu^2$): 4,877,928 nodes (63.2% of the nodes in the basic R-tree)

By putting these numbers into relation with the values listed in Table 45, it shows that the seemingly strong preference for the CBLQ code is mostly because it is the standard encoding[261] and only rather rarely because it is more favorable[262]. The observation of the rising LQ shares for MBRQT_5 respectively MBRQT_6 in comparison to MBRQT_2 respectively MBRQT_3 is counter-intuitive, though, as in theory, larger quadtree structures with a greater amount of black cells should result for the former (due to the respective parameterizations). Nevertheless, the erratic spatial distribution of the T collection's data points and also the quadtree condensation have to be taken into account as additional influencing factors. We do not further investigate this issue at this point as the implications of the results for the summary description types are clear:
For many of the MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ parameterizations, the option between LQ and CBLQ encoding could also be entirely removed from the summary calculation processes because one of the encodings is the preferred option in almost 100% of the cases. For them, the appropriate encoding scheme could be preselected. This would save CPU time in the MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summary calculation processes. Additionally, the metadata component could be removed from the respective summaries'

[261] See e.g. MBRQT_1 with 1.46% more indexed areas in comparison to the MBR approach, 98.6% nodes with non-refined MBRs, and a 99.5% share of CBLQ-encoded summaries.
[262] See e.g. MBRQT_3 with 56.9% more indexed areas compared to the MBR approach, 63.2% nodes with non-refined MBRs, and a 99.2% share of CBLQ-encoded summaries.

bit vectors, reducing the storage space requirements. For techniques such as MBRQT_6 and QTMBR_4, the preferences are relatively mixed. Hence, in such cases, the option to select between LQ and CBLQ encoding can be retained. Otherwise, it is advisable to preselect the appropriate linear quadtree encoding scheme to save CPU time and storage space. In any case, the zipping options should be omitted.
Table 46 depicts the impacts of the MBR-summary-replacement on the indexed surface areas. In the table, the indexed surface areas are cumulated and listed level-wise. This means that e.g. the nodes at level 1 in the MBR R-tree index an overall surface area of 85,168.2 dsu^2. For the first three levels of the MBR R-tree, the cumulated indexed surface areas are significantly greater than the data space size of 64,800 dsu^2. Hence, the split algorithm obviously does not achieve a 'perfect' (i.e. overlap-free) division of the data points into groups and therefore, the *MBRs* of the nodes overlap each other to substantial extents. The visualization in Figure 96 shows that indeed, the MBR summaries of the five nodes at level 1 of the basic R-tree exhibit significant overlap areas. This implies that there is possibly also overlap between the *data point clouds* of the nodes. Such a situation is favorable for our summarization approaches as they are very effective for the differentiated description of mutually overlapping data point clouds—as has been demonstrated in the evaluation of the distributed application scenario.

Table 46: Cumulated indexed surface areas of the MBR-like R-trees in the T_5 scenario (all values in dsu^2).

technique	level 1	level 2	level 3	level 4 (leaf nodes)
MBR	85,168.2	104,823.8	89,183.2	18,751.4
MBRQT_1	85,168.2	104,778.9	87,060.8	9,913.0
MBRQT_2	85,168.2	104,778.9	87,067.9	9,869.4
MBRQT_3	85,168.2	104,778.9	87,065.5	9,889.1
MBRQT_4	84,967.1	101,683.9	72,905.2	3,003.9
MBRQT_5	84,967.1	101,683.9	72,795.0	2,942.7
MBRQT_6	84,967.1	101,683.9	72,759.3	2,941.5
QTMBR_1	81,625.3	99,917.6	94,816.3	15,355.7
QTMBR_2	79,871.0	95,237.5	82,103.0	6,671.5
QTMBR_3	81,625.3	99,848.9	93,728.0	496.7
QTMBR_4	73,547.9	82,306.0	73,039.5	0.9

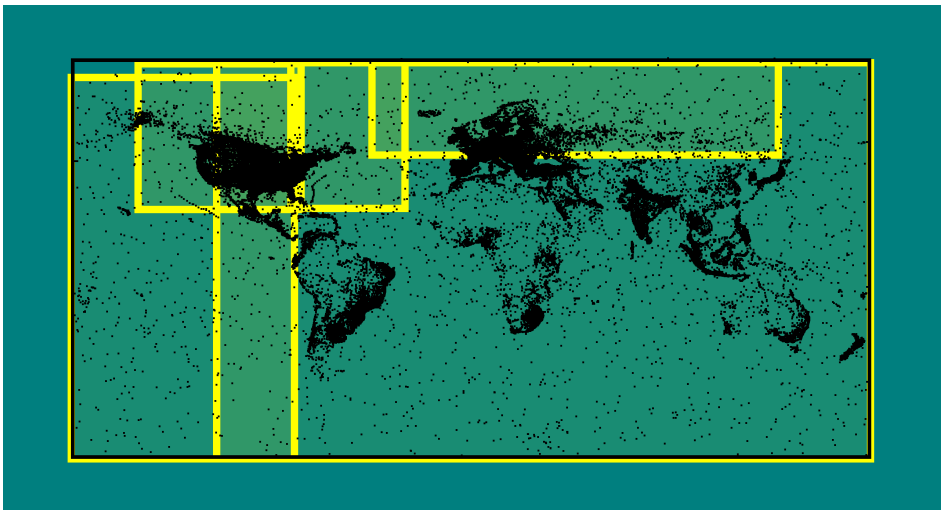


Fig. 96: Visualization of the MBR summaries of the five internal nodes at level 1 of the basic, MBR-like R-tree (T_5 scenario). The black dots denote the data points stored in the R-tree.

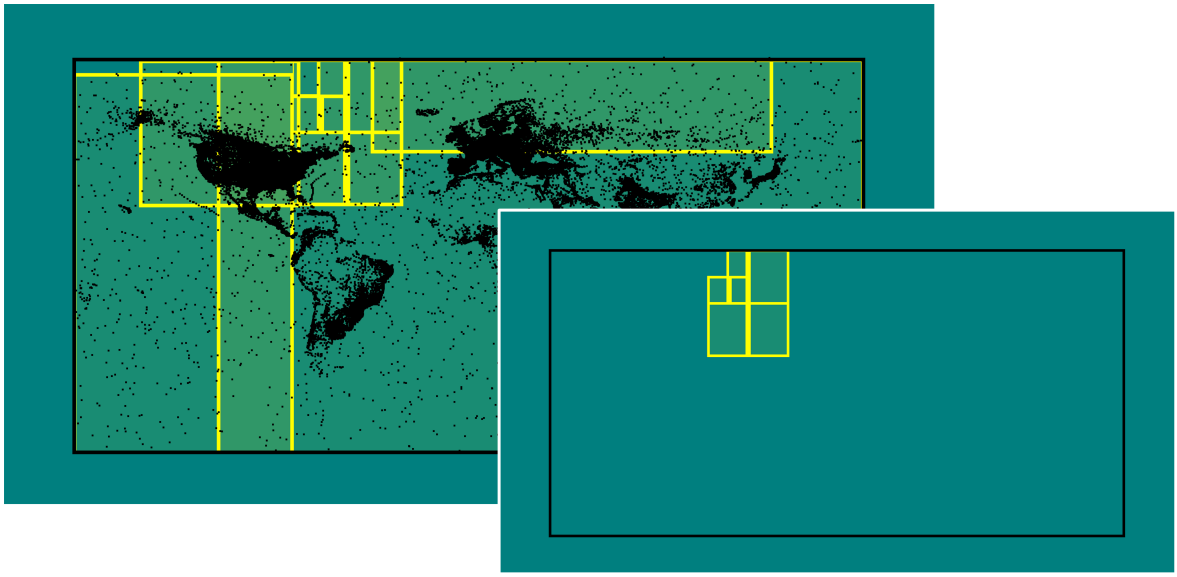


Fig. 97: Visualization of the MBRQT_6 summaries of the five internal nodes at level 1 of the basic, MBR-like R-tree (T_5 scenario). It can be seen that for four of the five nodes, their MBRQT_6 summary corresponds to a simple MBR (upper left image of the figure). Only for one node, a quadtree space partition is visible, i.e a reduction of the indexed surface area can be achieved. See lower right image of the figure (not displaying the collection's data points anymore): there are seven indexed areas but at the upper left corner, a region of dead space is excluded from being indexed.

Nevertheless, Table 46 shows that for the first three levels of the R-tree (which are made up of internal nodes), almost no improvements over

[263] Of course, this is an indirect effect as the summary of an internal node is calculated from its direct child nodes' summaries and not directly from the data points themselves. Nevertheless, since a summary covers all the data points stored in its node's subtree, the outcome is the same.

the MBR approach can be achieved by utilizing our approaches. For the MBRQT^{c,a} parameterizations, the reason is that the data space within the basic MBRs is so densely populated with data points that the refinement quadtrees are oftentimes condensed to a single black cell.[263] Hence, in such cases, their indexed surface areas correspond to their basic MBRs. See Figure 97 for an example visualization. The closer to the leaf node level, the greater the improvements of the MBRQT^{c,a} parameterizations (especially for those with $c = 16$) over the MBR approach. However, the overall improvements are only very modest for the internal nodes. The QTMBR$^{b}_{c,a}$ parameterizations are slightly better than all MBRQT^{c,a} parameterizations at the first two levels but are worse than the MBRQT^{c,a} parameterizations with $c = 16$ (i.e. MBRQT_4 to MBRQT_6) at level 3. Considering the massive superiority of the QTMBR$^{b}_{c,a}$ approach with regard to indexed surface areas in the distributed application scenario, it is surprising that the QTMBR$^{b}_{c,a}$ parameterizations do not offer significantly better results. Nevertheless, there are several reasons for these outcomes. For once, especially at level 1 and level 2, the utilization of (possibly) several indexed areas does not really pay off for QTMBR$^{b}_{c,a}$ in comparison to the MBR approach as—due to the large amount of data points in the level-1- and level-2-nodes' subtrees—the union of these areas still more or less corresponds to the full-precision MBR. See Figure 98 for an example visualization.

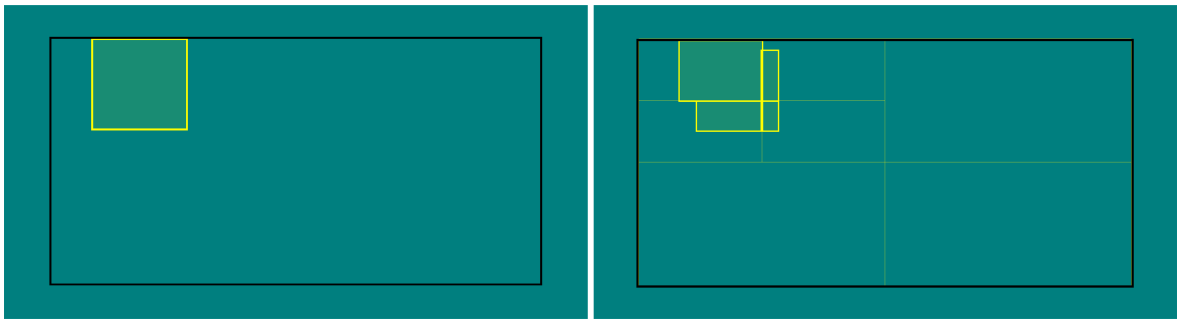


Fig. 98: Example visualization of the basic, MBR-like R-tree's node with $ID = 6,097,670$ (node 6,097,670, which is at level 1 of the R-tree) with its MBR summary (left) and its QTMBR_3 summary (right). It can be seen that the overall correspondance between both summaries is very large despite the fact that the QTMBR_3 summary indexes four separate areas.

Furthermore, for the *internal nodes*, the QTMBR$^{b}_{c,a}$ summaries are in general seriously restricted in their maximum spatial accuracy: In section 10.4, we outlined that in order to limit the worst case storage space requirements for internal node summaries, the target depth td of a summary's *initial* quadtree is set in a such way that even if each cell of the initial quadtree is occupied with data points, no more than c cells can oc-

cur in the *final* quadtree.[264] Therefore, we specified that td is calculated as $\lfloor \log_4\ c \rfloor$. The specification's consequences for the different QTMBR$^b_{c,a}$ parameterizations are displayed in Table 47.

Table 47: Overview of the theoretical maximum amounts of quadtree cells in the initial quadtree of an internal node's summary, listed for the various QTMBR$^b_{c,a}$ parameterizations. These theoretical maxima result from target depth td (which is used in the summary-from-summaries calculation processes for QTMBR$^b_{c,a}$ and also MBRQT^{c,a} summaries).

	c	$\log_4\ c$	$td = \lfloor \log_4\ c \rfloor$	theoretical maximum amount of quadtree cells
QTMBR_1	16	2	2	16
QTMBR_2	16	2	2	16
QTMBR_3	32	2.5	2	16
QTMBR_4	64	3	3	64

As can be seen in Figure 98, at the upper levels of the R-tree (i.e. level 1 and level 2), it is well possible that *several* basic quadtree cells of an internal node's summary are occupied with data points. However, at the latest at level 3, the data points administered in an internal node's subtree are located in rather narrow regions of the data space. Since for the internal nodes' summaries, the basic quadtrees are restricted in their maximum depth ($td = 2$ for QTMBR_1 to QTMBR_3, $td = 3$ for QTMBR_4), they remain very coarse. See Figure 99 for example visualizations. Both images show summaries of the level-3-node 5,866,624. All the data points in the subtree of node 5,866,624 are located in a spatially very narrow region.[265] In the top image of Figure 99, the QTMBR_2 summary of node 5,866,624 is displayed. In the bottom image, it is the node's QTMBR_4 summary. It can be seen that due to the limited depth of the respective basic quadtree structures, only seven (QTMBR_2) respectively ten (QTMBR_4) quadtree cells are built in the respective quadtree space partitions—from which in both cases only one cell is occupied, i.e. solely one quantized MBR describes the spatial footprint of node 5,866,624. Consequently, for the spatially narrow level-3-nodes, the QTMBR$^b_{c,a}$ summaries (regardless of the parameterization) mostly correspond to coarser approximations of the full-precision MBRs. Hence, it is easily understandable why the QTMBR$^b_{c,a}$ parameterizations only marginally improve respectively are partly even worse than

[264] Note that the same applies to the MBRQT^{c,a} summaries. However, due to the rather low values of the MBRQT^{c,a} parameterizations for c, the effects introduced by target depth td are not as pronounced as for the QTMBR$^b_{c,a}$ parameterizations.

[265] Actually, all data points in its subtree are duplicates, i.e. node 5,866,624 is an internal node for which all of its leaf nodes store duplicates. Consequently, the MBR summary of node 5,866,624 has a surface area of 0 dsu^2.

the MBR approach and the MBRQT^{c,a} parameterizations with regard to the indexed surface areas at level 3.

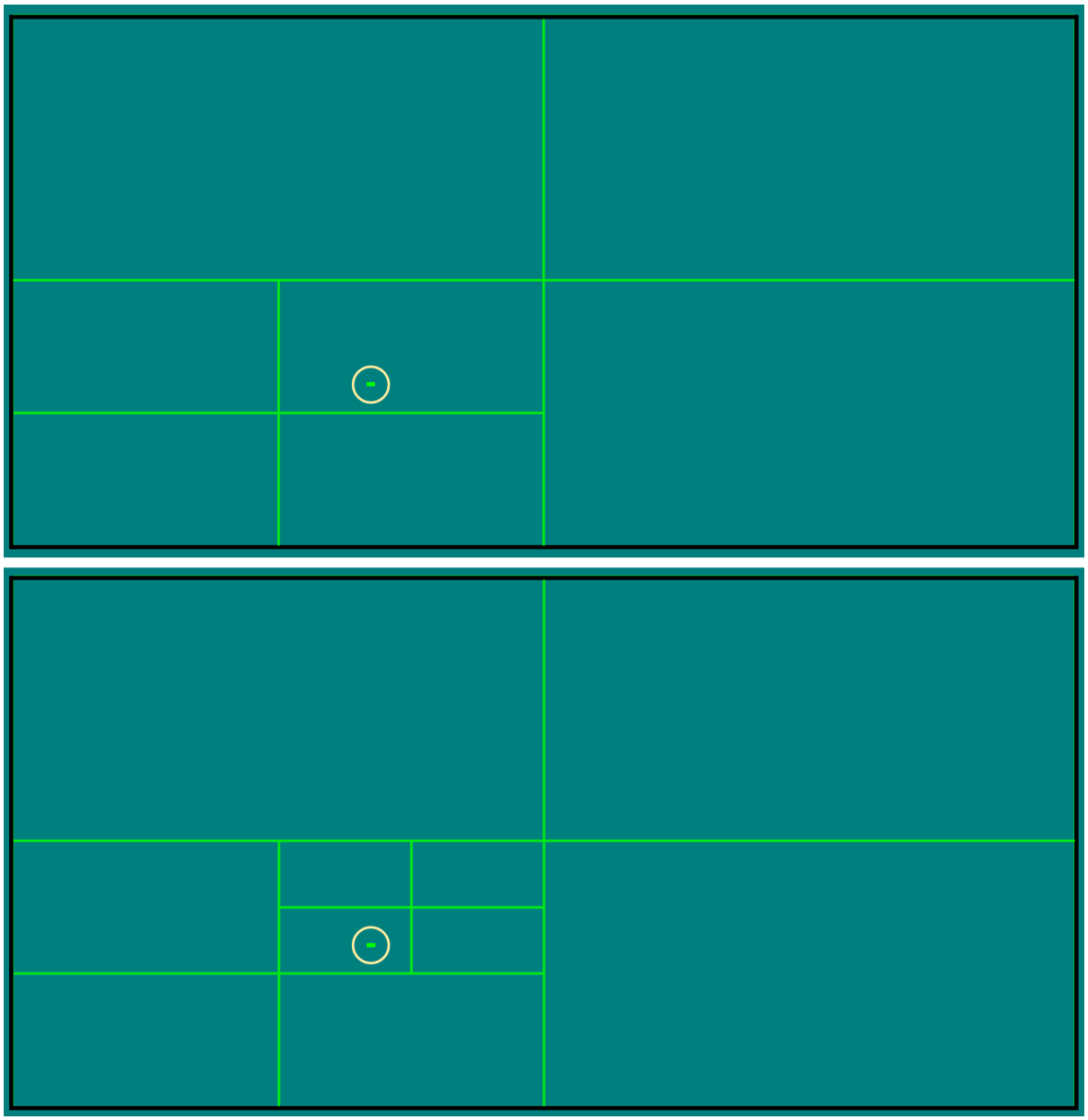


Fig. 99: Visualization of the QTMBR_2 summary (top) and the QTMBR_4 summary (bottom) of the level-3-node 5,866,624. Both summaries consist of a single quantized MBR. For the quantized MBRs to be visible at all, a great weight was given to their boundary lines. Additionally, the regions in which the quantized MBRs are located are highlighted with yellow circles.

Notably, the QTMBR_2 summaries actually have the most accurate smallest indexable spatial unit of the QTMBR$^{b}_{c,a}$ parameterizations: They only build the quadtree to a depth of $td = 2$ but the refining MBR is quantized

[266] 12 bits mean that 4,096 positions can be distinguished per dimension. As there are a minimum and a maximum boundary in each dimension, these 4,096 positions define 4,095 intervals in the very dimension. Hence, a quantization grid of 4,095 $\times$ 4,095 cells results.

with $b = 12$ bits, i.e. a quantization grid of 4,095 $\times$ 4,095 cells is used.[266] Therefore, a cell of a QTMBR_2 summary's quantization grid is actually smaller than the cell of a QTMBR_4 summary's quantization grid which builds the basic quadtree to a depth of $td = 3$ but only quantizes the refining MBRs with $b = 8$ bits (i.e. its quantization grid has 255 $\times$ 255 cells). QTMBR_1 and QTMBR_3 (both with $td = 2$ and $b = 8$) are the techniques for which their smallest indexable spatial units are the coarsest.

Overall, by use of our tested techniques, the MBR approach's cumulated indexed surface areas per R-tree level can be reduced by up to 21.5% for internal nodes (see QTMBR_4 on level 2). Considering the results for the distributed application scenario, this is rather disappointing. Nevertheless, the general main problem is easily comprehensible: For an internal node, the amount of data points administered in its subtree is very high. Consequently, the associated data space region is very densely populated with data points. This is the reason why similar problems occur as for the T2 collection's resources in the distributed application scenario: It would require a substantially higher parameterization of the MBRQT^{c,a} respectively the QTMBR$^{b}_{c,a}$ approach to achieve significant improvements with regard to the spatial accuracy—with the corresponding consequences on the storage space requirements (and subsequently for the number of distance calculations as well as, more importantly, the fanouts of the *summary-like* R-trees). As the cumulated indexed surface areas of the single levels of the R-trees show, the overlap between the internal nodes at the same level is considerable. For example, at level 2, the overlap between the MBR summaries is at least $(104,823.8\,dsu^2/64.800\,dsu^2) - 100\% = 61.8\%$ of the data space. Hence, it has to be possible to further reduce the indexed surface areas in substantial quantities—but not with the parameterizations we use for MBRQT^{c,a} respectively QTMBR$^{b}_{c,a}$. As the indexed surface area of a node corresponds to the probability of it being accessed while querying, it cannot be expected that in the T_5 scenario, the amount of disk accesses for internal nodes will be significantly reduced with our summarization approaches in comparison to the MBR approach—even though we test for MBR-like R-trees.

For the leaf nodes, it might be different as the cumulated indexed surface areas at leaf node level are greatly reduced. For the three MBRQT^{c,a} parameterizations with $c = 8$ (entitled as MBRQT^{8,a} in the following), the indexed surface areas can be reduced by roughly 50% whereas for the three parameterizations with $c = 16$ (MBRQT^{16,a}), up to 84% can be saved. For QTMBR_1, the low number of quadtree cells ($c = 16$) in combination with the quantization with $b = 8$ is obviously not enough to achieve substantial improvements over the MBR approach.[267] In contrast, for QTMBR_4, tremendous improvements result as the cumulated indexed surface area

[267] Note that at leaf node level, the basic quadtrees of the QTMBR$^{b}_{c,a}$ summaries are not artificially restricted in their depths since no target depth td is applied when calculating summaries from data points.

at leaf node level is reduced by more than 99.99%. Hence, as already evident in the distributed application scenario, the potential of $\text{QTMBR}^{b}_{c,a}$ to reduce indexed surface areas is enormous (even when taking the additional storage space expenses into account). In an intra-approach comparison of QTMBR_2 and QTMBR_3, both consume about the same amounts of storage space, see 'avg memory consumption'-column of Table 45. Hence, when assessing the indexed surface areas at leaf node level for QTMBR_3 (497 dsu^2) and QTMBR_2 (6,672 dsu^2), it is evident that for leaf node summaries, more detailed quadtrees (as for QTMBR_3 with $c = 32$ and $a = 1.0E-7$ dsu^2 as opposed to QTMBR_2 with $c = 16$ and $a = 1.0E-5$ dsu^2) are much more beneficial for reducing indexed surface areas than an accurate quantization. This is reasonable: In more detailed quadtrees, more black cells are built. As each black cell contains a quantized MBR *of its own*, the achieved surface area reductions are usually fairly remarkable in comparison to less detailed basic quadtrees.

Overall, the numbers indicate that it is more likely that disk accesses can be saved at leaf node level. Nevertheless, it has to be kept in mind that each value listed in Table 46 is cumulated over its entire level of the respective R-tree. This is especially important for the leaf node level with its 7.65 million nodes. In the distributed application scenario, it has been shown that the indexed surface areas were Zipf distributed, i.e. few resources were 'responsible' for the majority of the cumulated indexed surface areas. Due to the erratic spatial distribution of the T collection's data points and the low leaf node capacity, it is conceivable that this might also be the case here, especially since leaf nodes can also be entirely occupied with duplicates (leading to zero volume MBRs for the MBR approach and the $\text{MBRQT}^{c,a}$ parameterizations). Hence, despite overall smaller indexed surface areas, the $\text{QTMBR}^{b}_{c,a}$ parameterizations might be at a disadvantage in regions which are densely populated with data points because their smallest indexable spatial units are not 'infinitely' small—in contrast to a full-precision MBR.

In Figure 100, the distributions of the leaf nodes' indexed surface areas[268] are exemplarily depicted for the MBR approach as well as for the lowest and highest parameterizations of $\text{MBRQT}^{c,a}$ respectively $\text{QTMBR}^{b}_{c,a}$. On the x-axis, the nodes are sorted in descending order by their indexed surface areas.[269] It shows that indeed, the indexed surface areas of the nodes exhibit a typical long tail distribution, i.e. there are few nodes with large surface areas (making up the majority of the cumulated values) and many nodes with very small surface areas.

As expected, the curves of the MBR approach and MBRQT_1 respectively MBRQT_6 coincide after some points because parameter a prevents the re-

[268] Note that since Figure 100 displays leaf node results, the following explanations refer to the leaf node level.

[269] This is similar to Figure 56 on page 212 which displayed the indexed surface areas for the resources of the T1 collection in the distributed application scenario.

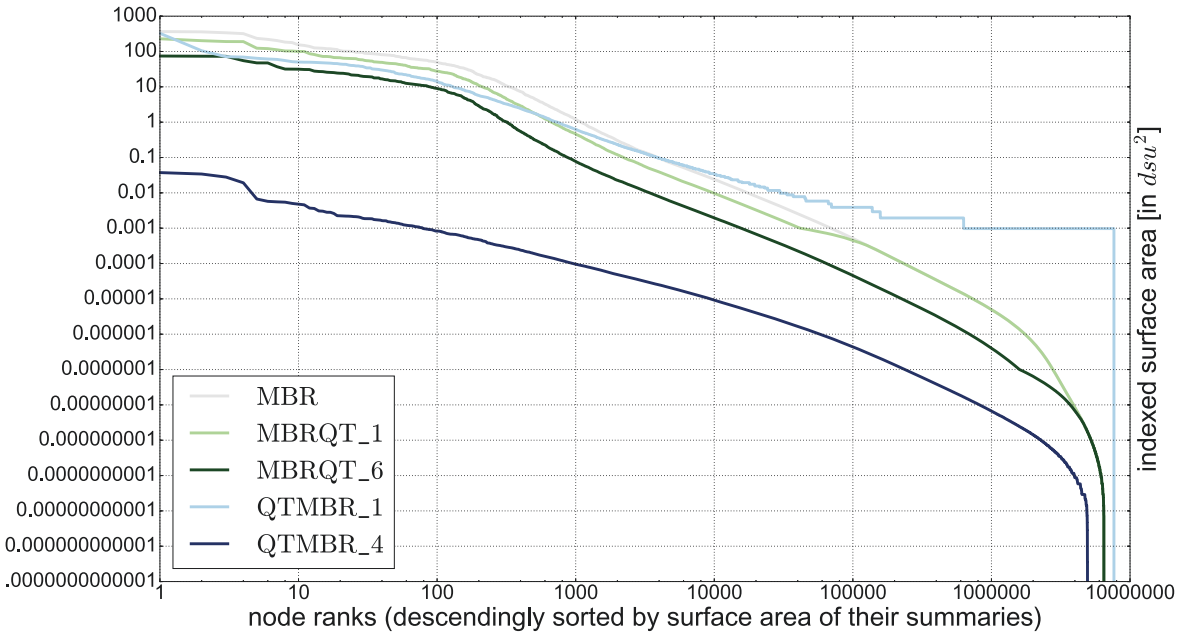


Fig. 100: Leaf nodes sorted in descending order by their indexed surface areas (MBR-like R-trees in the T_5 scenario). Both axes are log-scaled.

finement of the MBRQT^{c,a} summaries' basic MBRs. For MBRQT_1, the surface areas of the nodes correspond to those of the MBR approach from rank 130,183 on while for MBRQT_6, they correspond from rank 4,556,762 on. Prior to these points, the respective MBRQT^{c,a} parameterizations succeed in reducing the indexed surface areas. Hence, improvements with regard to the leaf node accesses can be expected for all MBRQT^{c,a} parameterizations as the indexed surface areas of all nodes are smaller than or at least equal to those of the MBR approach. Since for MBRQT_1, only 130,182 leaf nodes (of 7.65 million in total) feature a smaller indexed surface area in comparison to the MBR approach, the expectations should not be too high. For MBRQT_6, things look clearly better. Nevertheless, only the evaluation of the queries can show how big the improvements achievable on basis of the surface area reductions are.

For QTMBR_1, the situation is different. QTMBR_1 features smaller indexed surface areas than the MBR approach for the highest-ranked nodes. However, already from rank 4,018 on, it offers *larger* surface areas. Hence, since for QTMBR_1, the overwhelming majority of leaf nodes has a coarser summary in comparison to the MBR approach, it can be expected that the usage of QTMBR_1 leads to a worse leaf node access performance although the cumulated indexed surface area at leaf node level is smaller. Especially in data space regions with a high global point density, the suitability of QTMBR_1 might be severely limited.

For the highest-ranked nodes, QTMBR_4 reduces the indexed surface areas tremendously. For example, the highest-ranked node of the MBR approach has an indexed surface area of 366.4 dsu^2 whereas it is only 0.037 dsu^2 for QTMBR_4. But also the remainder of the nodes is described with significantly greater accuracy—until finally, the nodes' surface areas all

reach zero volume for both. Hereby, QTMBR_4 has more zero volume nodes (2,719,567) than the MBR approach (1,175,223). This is unexpected as for $\text{QTMBR}^{b}_{c,a}$, in general, zero volume summaries should only arise in very special cases: When the data points to describe are all located at exactly the same position (i.e. they are all on a point) or on an iso-oriented line, and *simultaneously*, this point or line is aligned with the hyperplanes of the occupied quadtree cell's quantization grid. See Figure 101 for an example visualization. Due to these surprising results, we exemplarily examined a node whose QTMBR_4 summary features a surface area of 0 dsu^2. This node stores four data points. Three of these data points feature the coordinates ($x = -159.3656$; $y = 21.98574$), one features the coordinates ($x = -159.36938$; $y = 21.988964$). The node's data point set is described by two rectangles. Their extents are the following:

— *rectangle 1 (containing the first three data points)*:
  — lower left: ($x = -159.3656$; $y = 21.985740$)
  — upper right: ($x = -159.3656$; $y = 21.985743$)
— *rectangle 2 (containing the fourth data point)*:
  — lower left: ($x = -159.36938$; $y = 21.988964$)
  — upper right: ($x = -159.36938$; $y = 21.988968$)

It can be seen that in both cases, the x values of the lower left and upper right corners are identical while for the y values, there are minimal differences. Consequently, both rectangles are actually (very short) vertical lines which have a surface area of 0 dsu^2. Obviously, for such spatially narrow data point sets, the basic quadtree of QTMBR_4 takes full advantage of its adaptiveness and the high amount of quadtree cells ($c = 64$) to describe very small black quadtree cells in which the data points are contained. Then, onto each black cell, a quantization grid with 256 positions per dimension[270] is superimposed. In the end, the quantization grid is so fine-grained that the 32-bit single precision floating-point number format is not sufficiently accurate anymore to be able to make distinctions and consequently, in at least one dimension, the lower and upper bound of the quantized MBR's interval fall together (as can be seen for the x values of the two rectangles in this example).[271]
Of course, it is also possible that the indexed areas of QTMBR_4 summaries are points instead of lines, i.e. the intervals' bounds fall together in both dimensions. An investigation shows that out of the 15,573,029 indexed areas or rectangles of QTMBR_4, 3,240,777 rectangles are actually lines

[270] This is because $b = 8$ for QTMBR_4. The resulting grid has 255×255 cells.

[271] The problem is the more pronounced, the greater the absolute numeric value in front of the decimal point is because then, the single precision floating-point number format depicts fewer decimal places. Hence, in the 'peripheral' regions of the data space, the indexation of lines (or points) should happen more frequently.

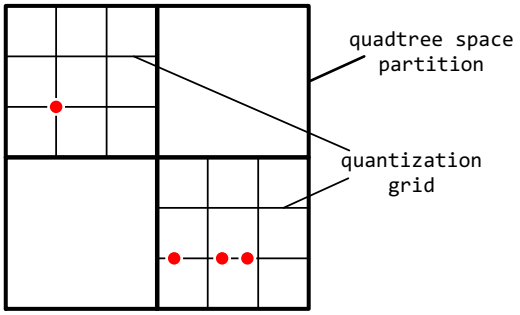


Fig. 101: Example visualization illustrating the cases in which QTMBR$^{b}_{c,a}$ summaries actually index points (upper left quadrant of the quadtree space partition) respectively lines (lower right quadrant).

and 4,965,344 rectangles are actually points.[272] Obviously, in comparison to point-like rectangles, the line-like rectangles are unfavorable for both range and kNN queries despite also featuring a zero volume surface area. The evaluation has to show the extent to which this has an adverse effect for QTMBR_4 in comparison to the MBR approach and the MBRQT^{c,a} parameterizations (for all of which point-like rectangles should usually result for zero volume nodes). Nevertheless, in any case, the reason behind the surprisingly high share of zero volume nodes in the QTMBR_4 R-tree has been revealed. For the other QTMBR$^{b}_{c,a}$ parameterizations, the amounts of point-like and line-like rectangles are very low as they do not run into the accuracy limitations of the number format.[273] At this point, we conclude the structural analysis of the MBR-like R-trees in the T_5 scenario and continue with the analysis of the MBR-like R-trees in the T_25 scenario.

Structural Analysis for the MBR-like R-trees in the T_25 scenario

The basic R-tree in the T_25 scenario has significantly less leaf nodes. This is because the leaf nodes' data point capacity is now 25 instead of 5. As a consequence, the R-tree also contains substantially less internal nodes, and its height is only 4. Still, 99.14% of the R-tree's nodes are leaf nodes, i.e. the proportions between the amounts of internal and leaf nodes do not change. Also, the memory utilization rates of both internal and leaf nodes as well as the fanout remain fairly the same (see Table 48).
With regard to the memory consumption of the internal nodes, Table 49 shows that in comparison to the T_5 scenario, QTMBR_3 consumes only slightly more memory on average but its *maximum* consumption is signif-

[272] Note that we did not examine how these lines and rectangles split among the 2,719,567 zero volume and 4,934,866 non-zero volume nodes of QTMBR_4.
[273] The concrete numbers are as follows:

— QTMBR_1: 25 points, 293 lines,
— QTMBR_2: 349 points, 851 lines, and
— QTMBR_3: 47 points, 1,105 lines.

Table 48: Structural information on the basic R-tree in the T_25 scenario.

technique	height	# nodes [in M]	# internal nodes	avg mem. utilization int. nodes	# leaf nodes [in M]	avg mem. utilization leaf nodes	avg fanout
basic R-tree	4	1.57	13,444	68.6%	1.56	68.5%	117.1

Table 49: Memory consumption of the internal nodes in the MBR-like R-trees in the T_25 scenario. Additionally, the respective amounts of indexed areas are listed.

memory consumption internal nodes → technique ↓	avg [in B]	median [in B]	75%-quant. [in B]	max [in B]	number of indexed areas
MBR	2,811.4	2,760	3,312	4,080	1,574,842
MBRQT_1	3,748.5	3,680	4,416	5,440	1,871,902
MBRQT_2	3,748.5	3,680	4,416	5,440	3,760,124
MBRQT_3	3,748.5	3,680	4,416	5,440	5,157,349
MBRQT_4	3,748.5	3,680	4,416	5,440	2,083,967
MBRQT_5	3,748.5	3,680	4,416	5,440	5,208,217
MBRQT_6	3,748.5	3,680	4,416	5,440	7,584,945
QTMBR_1	1,878.2	1,840	2,208	2,976	1,581,572
QTMBR_2	2,807.0	2,752	3,312	4,232	1,581,479
QTMBR_3	2,872.7	2,808	3,384	5,072	1,715,654
QTMBR_4	5,722.6	5,624	6,776	10,784	6,652,163

icantly greater (5,072 B instead of 4,304 B). QTMBR_4 consumes significantly more memory on average (5,722.6 B instead of 4,151.3 B) as well as in all listed descriptive statistic values. Both effects are caused by that the leaf nodes now contain up to 25 data points, making their summaries considerably more complex. This affects the memory consumption of the level-2-nodes which store the leaf nodes' summaries and make up 99.15% of the internal nodes (level 2 has 13,330 nodes). For the MBRQT^{c,a} parameterizations as well as QTMBR_1 and QTMBR_2, the rather low values for parameter c as well as the byte- and word-alignment prevent a notable increase of the level-2-nodes' memory consumption, though. The total amount of indexed areas is smaller than in the T_5 scenario due to the lower amount of nodes in the R-tree. Nevertheless, in relation to the MBR approach, our summarization approaches now index significantly more areas. For example, QTMBR_4 indexes 102% *more* areas than the MBR approach in the T_5 scenario but 322% more areas in the T_25 scenario. This also indicates an increased complexity of the non-MBR summaries.

In Table 50, the summary-related results are depicted. With regard to the *memory consumption* of the summaries, only the average QTMBR_4 summary sizes increase significantly (again, in accordance with the average memory consumption of the internal nodes). However, in general,

the average *memory utilization rate* of the summaries increases for all MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ parameterizations. The effect is greatest for QTMBR_4 (+4.3% in the *absolute* memory utilization), MBRQT_5 (+8.1%), and MBRQT_6 (+10.1%). With respect to the summary encoding types, the LQ share decreases significantly for MBRQT_6 (from 28.5% to 13.5%). This is although now, only 340,934 or 21.6% of the nodes in the R-tree are described by non-refined MBRs (as opposed to 63.2% in the T_5 scenario), i.e. the impact of considering the CBLQ encoding as standard is not as large anymore. The greater capacity of the leaf nodes leads to a greater spatial spread of the data points stored in the leaf nodes such that now, the basic MBRs are refined more often. Moreover, the refinement quadtrees also tend to have more black cells and thus are preferably CBLQ-encoded. For QTMBR_4, the CBLQ share increases tremendously (from 20.8% to 72.0%). The other results are fairly stable. In general, the results for the summary encoding types are as one would have expected them when considering the increased leaf node capacity of 25 and with the explanations concerning the summary encoding type results from the analysis of the T_5 scenario in mind (more black quadtree cells $\rightarrow$ greater CBLQ share).

Table 50: Summary-related results for the MBR-like R-trees in the T_25 scenario.

technique	avg memory consumption [in B]	avg memory utilization	cblq-nz	cblq-z	sum-nz/ lq-nz	sum-z/ lq-z
MBR	16.0	100%	-	-	100%	-
MBRQT_1	24.0	68.4%	99.2%	-	0.8%	-
MBRQT_2	24.0	72.0%	98.3%	-	1.7%	-
MBRQT_3	24.0	74.9%	97.6%	-	2.4%	-
MBRQT_4	24.0	69.2%	98.7%	-	1.3%	-
MBRQT_5	24.0	77.6%	92.2%	-	7.8%	-
MBRQT_6	24.0	84.2%	86.5%	-	13.5%	-
QTMBR_1	8.0	88.9%	0.03%	-	100.0%	-
QTMBR_2	16.0	57.3%	0.03%	-	100.0%	-
QTMBR_3	16.5	59.7%	0.80%	-	99.2%	-
QTMBR_4	40.9	90.3%	72.0%	-	28.0%	-

[274]For example, for MBRQT_3 and QTMBR_2, the reductions of the leaf nodes' cumulated indexed surface areas in comparison to the MBR approach are as follows:

— T_5 scenario, MBR-like R-trees:
  — MBRQT_3: -47.3%
  — QTMBR_2: -64.4%
— T_25 scenario, MBR-like R-trees:
  — MBRQT_3: -21.6%
  — QTMBR_2: -49.5%

For the cumulated indexed surface areas (see Table 51), there are no relevant changes in the results: Still, only marginal improvements can be achieved for the internal nodes' levels. At leaf node level, the *relative* improvements over the MBR approach are now generally smaller, although QTMBR_3 and QTMBR_4 still tremendously reduce the indexed surface areas.[274] This is even though in comparison to the T_5 scenario, our summarization approaches now index much more areas in relation to the MBR approach (as just outlined in conjunction with Table 49).[275] Hence, a naive deduction is that for the MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summaries, it now happens more often that the union of a node's indexed areas is not much of a surface area reduction versus the corresponding MBR summary. This means the situations are more frequently similar to the circumstances depicted in Figure 98 on page 326. However, this deduction does not take the long tail distributions of the leaf nodes' indexed surface areas into account. Those are depicted in Figure 102 (for the same exemplarily selected techniques as before).

Table 51: Cumulated indexed surface areas of the MBR-like R-trees in the T_25 scenario (all values in dsu^2).

technique	level 1	level 2	level 3 (leaf nodes)
MBR	83,653.7	83,898.2	49,657.6
MBRQT_1	83,653.7	83,094.1	38,971.1
MBRQT_2	83,653.7	83,118.4	38,721.7
MBRQT_3	83,653.7	83,084.4	38,932.0
MBRQT_4	82,970.2	77,226.2	23,628.0
MBRQT_5	82,970.2	77,143.0	23,584.5
MBRQT_6	82,970.2	77,134.8	23,573.4
QTMBR_1	81,200.1	84,113.7	28,405.7
QTMBR_2	79,176.9	78,436.5	25,082.7
QTMBR_3	80,992.5	83,592.5	5,932.4
QTMBR_4	73,141.6	70,758.2	487.0

There, it shows that the superiority of MBRQT_1, MBRQT_6, and QTMBR_4 at the *single ranks* is less pronounced than in the T_5 scenario (see Figure 100). Nevertheless, the *share of leaf nodes* whose indexed surface areas are reduced is actually greater now. For example, for MBRQT_1, the indexed surface areas correspond to those of the MBR approach from

[275] As discussed in the evaluation of the distributed application scenario, a greater number of indexed areas might imply a more accurate spatial summarization as the data point set to describe is divided into a greater amount of groups which can then be tightly delineated by a geometric description, each.

rank 176,775 on, i.e. 176,774 or 11.2% of the leaf nodes are described with greater spatial accuracy. In the T_5 scenario, it was only 130,182 nodes which made up 1.7% of the leaf nodes in the corresponding basic R-tree. With regard to MBRQT_6, the indexed surface areas are now advantageous for 89.0% of the leaf nodes in the T_25 scenario as opposed to only 59.5% in the T_5 scenario. For the $\text{QTMBR}^{b}_{c,a}$ parameterizations, the results derivable from Figure 102 are a little more diversified but it basically also applies for them that the surface area reduction is not as large anymore for the highest-ranked leaf nodes but more favorable with regard to the share of leaf nodes for which improvements or deteriorations (see QTMBR_1) arise. Considering these findings, it is more likely to improve the leaf node access performance in the T_25 scenario than in the T_5 scenario. Again, it is evident that the cumulated indexed surface areas do not show the 'whole story'. Therefore, predictions taking solely this key figure into account are not very worthwhile.
With regard to the anticipated developments of the MBR-like R-trees' leaf node access performances from the T_5 scenario to the T_25 scenario, additionally to be considered is the following: A greater average amount of data points stored in the leaf nodes generally leads to less leaf node accesses when querying for the same result set—which means that from this point of view, it will be more difficult to achieve improvements in the T_25 scenario. Nevertheless, only the evaluation can show if there are actually differences in the improvements over the MBR approach between the T_5 scenario and the T_25 scenario and, in case there are differences, how they look like.

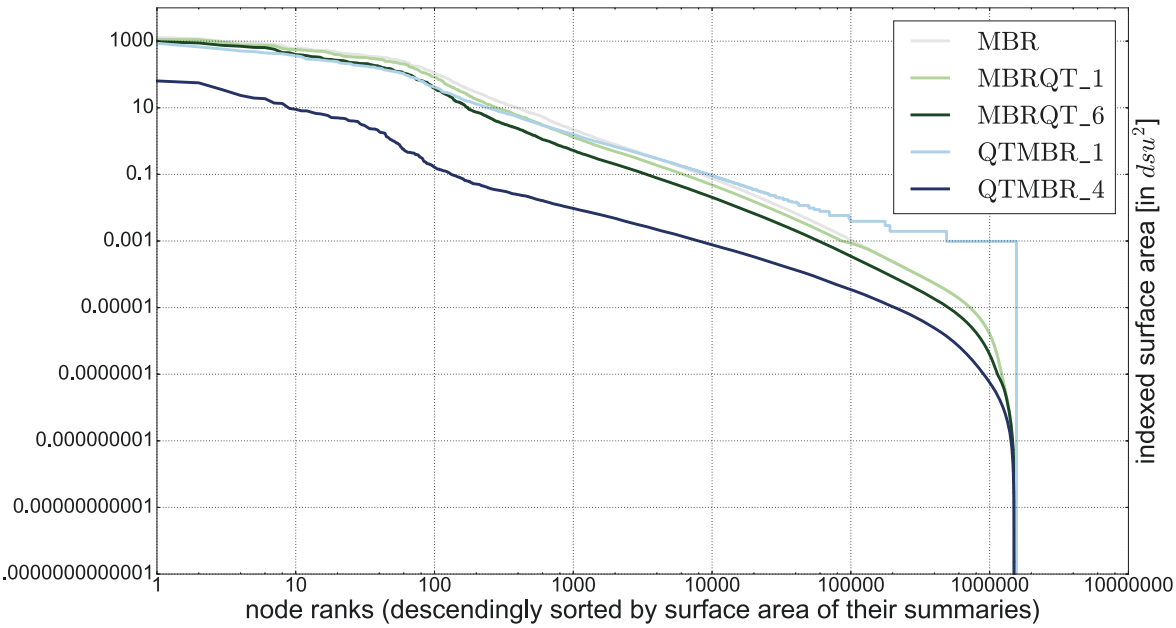


Fig. 102: Leaf nodes sorted in descending order by their indexed surface areas (MBR-like R-trees in the T_25 scenario). Both axes are log-scaled.

Structural Analysis for the MBR-like R-trees in the R_5 scenario

The R collection has only 25 million data points as opposed to the 26.7 million data points of the T collection. As a result, the basic R-tree in the R_5 scenario has fewer leaf nodes (6.86 million instead of 7.65 million) and consequently also fewer internal nodes (58,961 instead of 66,169) than the basic R-tree in the T_5 scenario (see Table 52). However, this means that still, 99.15% of the R-tree's nodes are leaf nodes.[276] Furthermore, the memory utilization of the leaf nodes is slightly greater in the R_5 scenario (72.9% instead of 69.9%) as there are less outlier data points which typically lead to the creation of sparsely filled leaf nodes.

Table 52: Structural information on the basic R-tree in the R_5 scenario.

technique	height	# nodes [in M]	# internal nodes	avg mem. utilization int. nodes	# leaf nodes [in M]	avg mem. utilization leaf nodes	avg fanout
basic R-tree	5	6.91	58,961	68.7%	6.86	72.9%	117.3

Table 53: Memory consumption of the internal nodes in the MBR-like R-trees in the R_5 scenario. Additionally, the respective amounts of indexed areas are listed.

memory consumption internal nodes → technique ↓	avg [in B]	median [in B]	75%-quant. [in B]	max [in B]	number of indexed areas
MBR	2,814.7	2,736	3,288	4,080	6,914,823
MBRQT_1	3,752.9	3,648	4,384	5,440	18,048,629
MBRQT_2	3,752.9	3,648	4,384	5,440	20,957,973
MBRQT_3	3,752.9	3,648	4,384	5,440	20,991,061
MBRQT_4	3,752.9	3,648	4,384	5,440	20,823,987
MBRQT_5	3,752.9	3,648	4,384	5,440	24,080,782
MBRQT_6	3,752.9	3,648	4,384	5,440	24,112,948
QTMBR_1	1,889.8	1,840	2,208	2,896	7,013,105
QTMBR_2	2,819.6	2,760	3,296	4,248	7,013,014
QTMBR_3	3,160.9	3,088	3,696	5,160	9,878,049
QTMBR_4	5,191.4	5,080	6,088	7,888	24,413,684

Compared to the T_5 scenario, the memory consumption of the internal nodes is fairly stable (slightly increased) for all techniques except QTMBR_3 and QTMBR_4 which exhibit significantly increased storage

[276] For the R-trees built with MBR summaries, our data shows that generally, out of the totality of an R-tree's nodes from level 1 to level i ($1 < i < \textit{R-tree.height}$), about 99.15% of the nodes are at level i. Obviously, this is related to the average fanout of the nodes as e.g. $1 - (1/117.3) = 99.15\%$ (117.3 is the fanout of the basic MBR-like R-tree in the R_5 scenario).

space requirements (see Table 53). Again, the reason is the greater spatial spread of the data points stored in a leaf node[277], leading to a greater amount of black cells in the basic quadtrees of QTMBR_3 and QTMBR_4. Due to the subsequent refinement of each black cell with a quantized MBR, this surplus on black cells increases itself for the summaries' storage space footprints which in turn increase the internal nodes' average memory consumption. Furthermore, the amounts of indexed areas for the MBRQT^{c,a} parameterizations are now clearly greater compared to the MBR approach. The surplus ranges from 161% (MBRQT_1) to 249% (MBRQT_6) *more* indexed areas. Anew, this is because of the greater spatial spread of the data points which leads to more basic MBRs being refined by a quadtree, and more black cells in these quadtrees. In contrast, for QTMBR_1 and QTMBR_2, still only few (less than 1.5%) more areas are indexed. For QTMBR_3, it is 42.9% more areas whereas for QTMBR_4, it is 24,413,684 indexed areas in total or 253% more areas compared to the MBR approach. Hence, for QTMBR_1 and QTMBR_2 summaries, the basic quadtrees are still oftentimes so coarse that all the data points to summarize are located in a single quadtree cell.[278]

Table 54: Summary-related results for the MBR-like R-trees in the R_5 scenario.

technique	avg memory consumption [in B]	avg memory utilization	cblq-nz	cblq-z	sum-nz/ lq-nz	sum-z/ lq-z
MBR	16.0	100%	-	-	100%	-
MBRQT_1	24.0	74.5%	99.1%	-	0.9%	-
MBRQT_2	24.0	77.0%	99.1%	-	0.9%	-
MBRQT_3	24.0	77.0%	99.1%	-	0.9%	-
MBRQT_4	24.0	82.1%	62.7%	-	37.3%	-
MBRQT_5	24.0	86.6%	39.5%	-	60.5%	-
MBRQT_6	24.0	86.6%	39.0%	-	61.0%	-
QTMBR_1	8.1	88.7%	0.02%	-	100.0%	-
QTMBR_2	16.0	57.5%	0.02%	-	100.0%	-
QTMBR_3	19.0	65.3%	2.3%	-	97.7%	-
QTMBR_4	36.3	92.7%	50.3%	$\ll$ 0.01%	49.7%	-

[277]Note that now, the greater spatial spread is caused by the regular spatial distribution of the data points rather than by the increased number of data points in the leaf nodes (as in the previously discussed T_25 scenario).

[278]This applies to both internal nodes at level 3 (due to the utilization of target depth td in the summary-from-summaries calculation, as has been shown in the analysis of the T_5 scenario) and leaf nodes. The aggregated results are dominated by the leaf node summaries, though, as 99.15% of the nodes are leaf nodes.

The summary-related results in Table 54 show that due to the comparatively larger amount of indexed areas and the consequent increases in the sizes of the summaries' bit vectors, the memory utilization rates of the summaries generally increase for all MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ parameterizations. This means the byte- and the word-alignment do not introduce as much filling data as in the T_5 scenario. With regard to the summary encoding types, QTMBR_4 now has a share of 50.3% CBLQ-encoded summaries (instead of 20.8%). The results of the other QTMBR$^{b}_{c,a}$ parameterizations remain fairly stable. These observations are within the expectations. For the three MBRQT^{16,a} parameterizations, the share of LQ-encoded summaries is now between 37.3% (MBRQT_4) and ~61% (for both MBRQT_5 and MBRQT_6).[279] Considering the increased amount of indexed areas and the consequential greater amount of black cells in the quadtrees in the R_5 scenario, it is an unexpected change. This is particularly true in view of the disparity to the results for the three MBRQT^{8,a} parameterizations. For example, MBRQT_3 has 20,991,061 indexed areas and a CBLQ share of 99.1% while for MBRQT_4, the numbers are 20,823,987 and 62.7%. Hence, despite a very similar amount of indexed areas, the encoding preferences are fairly different. The amounts of nodes described by non-refined basic MBRs are as follows:

— MBRQT_1 and MBRQT_4: 1,889,229 nodes (27.3% of the nodes in the R-tree)
— MBRQT_2 and MBRQT_5: 33,453 nodes (0.48% of the nodes in the R-tree)
— MBRQT_3 and MBRQT_6: 906 nodes (0.013% of the nodes in the R-tree)

Hence, the impact of counting the CBLQ encoding as standard encoding option is much less pronounced now. Nevertheless, due to the regular spatial distribution of the data points, the condensation of quadtrees becomes significantly more important here. Furthermore, the relation between the surface areas of the basic MBRs of the leaf nodes' summaries and parameter a certainly has to be investigated. We leave a more detailed clarification of these encoding-type-related phenomena up to future work as in this thesis, the summarization approaches are evaluated as they are. Thus, apart from identifying starting points for improvements, concrete suggestions (such as in which situations the linear quadtree encoding should be preset to which scheme) as well as their subsequent implementation and evaluation are not in the scope of this work.

Table 55 shows that for the cumulated indexed surface areas, on *all* levels with internal nodes, the MBRQT^{c,a} parameterizations only achieve *minimal* improvements compared to the MBR approach—if they do not even index exactly the same surface areas. This is because of the regular spatial distribution of the data points, for the internal nodes, only *very rarely*

[279] In the T_5 scenario, the corresponding LQ shares are MBRQT_4: 0.74%, MBRQT_5: 7.6%, MBRQT_6: 28.5%.

Table 55: Cumulated indexed surface areas of the MBR-like R-trees in the R_5 scenario (all values in dsu^2).

technique	level 1	level 2	level 3	level 4 (leaf nodes)
MBR	68,069.0	79,190.3	85,091.3	21,706.2
MBRQT_1	68,069.0	79,190.3	85,085.5	13,159.5
MBRQT_2	68,069.0	79,190.3	85,085.5	12,680.8
MBRQT_3	68,069.0	79,190.3	85,085.5	12,680.4
MBRQT_4	68,069.0	79,173.2	84,762.4	5,063.6
MBRQT_5	68,069.0	79,173.2	84,761.9	4,271.2
MBRQT_6	68,069.0	79,173.2	84,762.1	4,271.0
QTMBR_1	67,869.2	83,966.8	129,011.2	52,729.1
QTMBR_2	67,675.8	79,394.8	87,528.1	22,904.5
QTMBR_3	67,869.2	83,838.3	126,064.7	14,303.8
QTMBR_4	67,368.5	81,572.2	104,412.2	68.8

regions of dead space can be excluded by the MBRQT^{c,a} summaries. The same applies to level 1 and the QTMBR$^{b}_{c,a}$ parameterizations. At level 2 and level 3, the QTMBR$^{b}_{c,a}$ parameterizations are even worse than the MBR approach. The reasons behind this have already been discussed in the analysis of the T_5 scenario: At level 1 and level 2, the *union* of the indexed areas of a node's QTMBR$^{b}_{c,a}$ summary oftentimes corresponds to an approximation of the node's full-precision MBR. See Figure 103 for example visualizations. Of course, the restriction of the summaries' basic quadtrees to target depth td plays a role for this correspondance, too. Furthermore, the restriction is important at level 3 (which is the level above leaf node level): Since also in the R_5 scenario, the data points of a level-3-node's subtree are all located in a spatially narrow region, most often, only a single basic quadtree cell is black for a level-3-node's QTMBR$^{b}_{c,a}$ summary. This means that only one quantized MBR results for the node's spatial footprint (similar to the depiction in Figure 99 on page 328). Hence, in the very end, a level-3-node's QTMBR$^{b}_{c,a}$ summary only approximates the corresponding full-precision MBR. Due to the quantization, this approximation is less accurate than the full-precision MBR, leading to a greater indexed surface area. In fact, QTMBR_2 is best in its peer group for the internal nodes levels. This is reasonable as it has the most accurate quantization with $b = 12$, i.e. its indexed areas are more accurate approximations of the full-precision MBRs than those of the other QTMBR$^{b}_{c,a}$ parameterizations.

At leaf node level, the MBRQT^{8,a} parameterizations reduce the cumulated indexed surface area of the MBR approach by rougly 40% while the

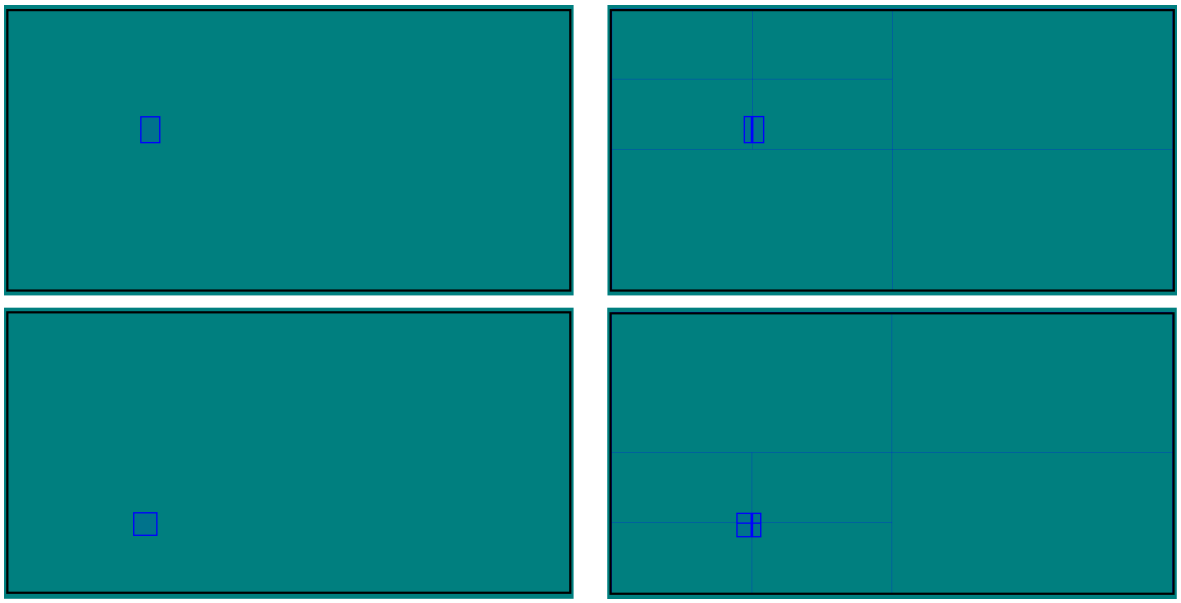


Fig. 103: Two exemplarily selected nodes (the node IDs are 775,999 and 853,853) at level 2 of the R_5 scenario's MBR-like R-tree. At the left, the nodes' areas which are indexed with the MBR approach are depicted. At the right, the corresponding areas when using QTMBR_3 are shown. It can be seen that the unions of the indexed areas of the QTMBR_3 summaries as a whole resemble the corresponding MBR summaries. However, due to the quantization, the areas indexed by the QTMBR_3 summaries only correspond to a *coarser approximation* of the respective full-precision MBRs.

MBRQT^{16,a} parameterizations achieve reductions of about 80%. QTMBR_1 (+143% indexed surface area compared to the MBR approach) and QTMBR_2 (+5.5%) are worse than the MBR approach. QTMBR_3 (-34.1%) is slightly worse than the three MBRQT^{8,a} parameterizations. As usual, QTMBR_4 (68.8 dsu^2 indexed surface area) achieves tremendous improvements over the MBR approach (21,706.2 dsu^2) and is the most successful of all techniques in this regard, by far.
In total, the pure cumulated indexed surface areas imply that it is less likely to improve the MBR approach in the R_5 scenario than in the T_5 scenario. This applies in particular to the QTMBR$^b_{c,a}$ parameterizations except QTMBR_4. However, a look at the distributions of the leaf nodes' indexed surface areas (see Figure 104) shows that in the R_5 scenario, no classical long tail distributions can be identified. Furthermore, almost no leaf nodes with a zero volume surface area exist anymore (for QTMBR_1, there is not even a single zero volume node). Thus, the cumulated indexed surface areas are now a much better indicator of the diverse techniques' overall indexing performance. In total, the superiority of our summarization approaches might even be substantially greater compared to the T_5 scenario. We return to this in the evaluation of the queries.

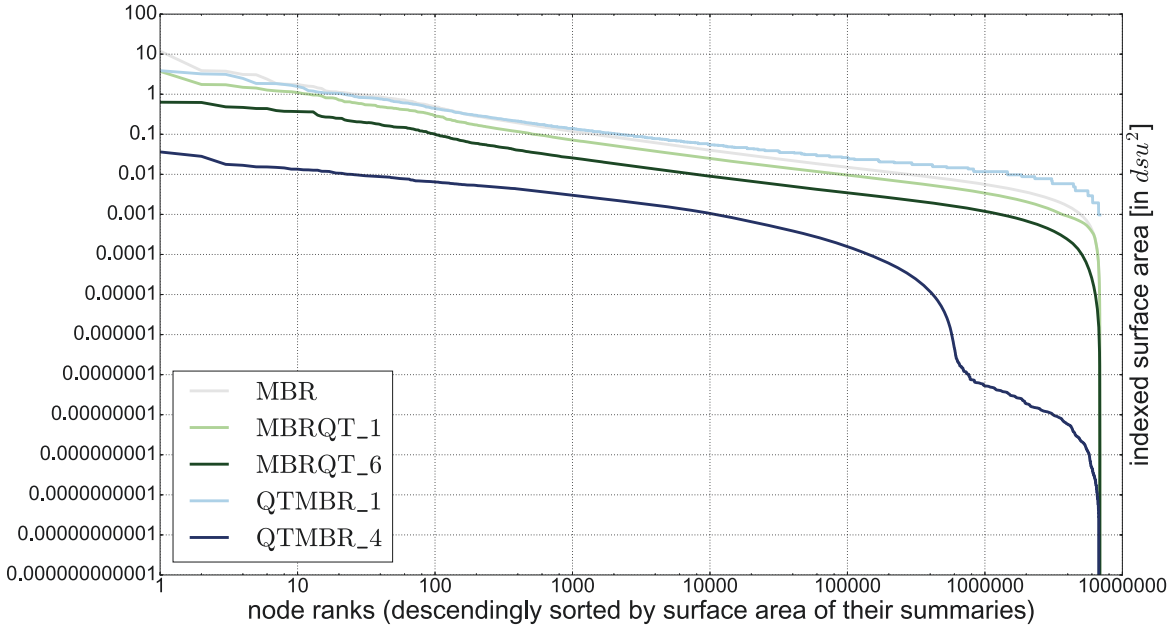


Fig. 104: Leaf nodes sorted in descending order by their indexed surface areas (MBR-like R-trees in the R_5 scenario). Both axes are log-scaled.

Structural Analysis for the MBR-like R-trees in the R_25 scenario

The basic MBR-like R-tree in the R_25 scenario has 1,448,237 nodes from which 12,230 are leaf nodes, i.e. 99.16% of its nodes are leaf nodes. Its height is 4 and the fanout is 118.4.
In the R_25 scenario, the results for the three MBRQT^{8,a} parameterizations are identical. The same goes for the three MBRQT^{16,a} parameterizations. Hence, parameter c is obviously the limiting factor here which is reasonable due to the regular spatial distribution of the data points *and* the leaf node capacity of 25 ($\rightarrow$ greatest spatial spread of all scenarios).
In comparison to the T_25 scenario, the internal nodes' memory consumption is stable for most techniques. Only QTMBR_3 (3,823.0 B on average instead of 2,872.7 B) and QTMBR_4 (8,990.4 B instead of 5,722.6 B) exhibit significantly increased storage space requirements. For both, the R_25 scenario generally leads to substantially larger storage space requirements than the other three scenarios. As a consequence, R_25 is the only scenario in which the QTMBR_3 R-tree's internal nodes allocate more storage space on average than those of the MBRQT^{c,a} parameterizations' R-trees. With regard to the summary-related results, the only notable changes are the following:

— The MBRQT^{8,a} parameterizations feature 81.2% CBLQ-encoded summaries whereas the MBRQT^{16,a} parameterizations feature 99.9% of these. Hence, for the first time, the proportions between both groups are as one would expect them to be without taking the existence of non-refined MBRs and the quadtree condensation into account.
— Furthermore, QTMBR_3 features 39.7% CBLQ-encoded summaries while for QTMBR_4, their share is 98.4%.

The cumulated indexed surface areas for the internal nodes show basically the same relative results between the techniques as in the R_5 scenario. For the leaf nodes, it is similar, only QTMBR_2 is now minimally better than the MBR approach (0.08% less cumulated indexed surface area) and naturally, the achieved reductions are generally smaller than in the other scenarios. The remaining relative results for the cumulated indexed surface areas are as follows (all in comparison to the MBR approach):

— MBRQT^{8,a}: -9.3%,
— MBRQT^{16,a}: -33.1%,
— QTMBR_1: +27.4%,
— QTMBR_3: -29.5%, and
— QTMBR_4: -90.1%.

At this point, we conclude the structural analysis of the MBR-like R-trees and continue with the structural analysis of the summary-like R-trees.

13.2.2. SUMMARY-LIKE R-TREES. In this section, the summary-like R-trees are evaluated, i.e. the R-trees which have been built by using the respective techniques' summaries in the construction phase. Hence, the structures of the resulting R-trees differ from each other which makes the comparison of the results more difficult. Nevertheless, they are realistic R-trees which consider the storage space limitations of the R-tree nodes—in contrast to the MBR-like R-trees. Again, we start with the assessment of the T_5 scenario.

Structural Analysis for the Summary-like R-trees in the T_5 scenario

Table 56 shows the results for the respective R-trees' structures. For the summary-like R-tree built with MBR summaries[280], similar structural results as for the basic, MBR-like R-tree in the T_5 scenario are obtained. In principle, both R-trees should be identical as the prerequisites (the data point set and its insertion order, and the applied algorithms SELECTBEST$(.)$ and R*-split) are the same. Nevertheless, slight differences arise as some randomness is involved in each R-tree construction (when ties are resolved in the SELECTBEST$(.)$-method and also in the R*-split).
Despite the randomness, when comparing the summary-like MBRQT^{c,a} R-trees with each other, all of them have almost identical numbers of internal and leaf nodes. Also, their memory utilization rates of both internal and leaf nodes are virtually identical to the summary-like MBR R-tree. Nevertheless, as 32 B are required for a child node entry with an MBRQT^{c,a} summary (instead of 24 B as with an MBR summary), the fanout of the MBRQT^{c,a} R-trees is 25% smaller in comparison to the MBR R-tree. Additionally, this leads to 33% more internal nodes for the MBRQT^{c,a} R-trees. In general, a greater fanout is advantageous with respect to the expectable amount of node accesses: The greater the fanout, the greater the average amount of child node entries in the internal node which is currently accessed. Hence, more child nodes can be assessed for their potential relevance with each internal node access that is conducted.[281] For the number of internal nodes, it applies that the greater their number, the greater the probability of conducting unnecessary accesses as well as that the query result is dispersed over several subtrees.
All the QTMBR$^{b}_{c,a}$ parameterizations have a greater amount of internal nodes in their summary-like R-trees than the MBR approach. With regard

[280]Note that we call this R-tree the *summary-like MBR R-tree* in the following. We proceed analogously with the R-trees for the other techniques, for example the *summary-like MBRQT_1 R-tree* or the *summary-like QTMBR_1 R-tree*. The specific techniques' MBR-like R-trees are referred to analogously, e.g. the *MBR-like MBRQT_1 R-tree*.

[281]Of course, this is an isolated consideration of the fanout's value in an R-tree. In a non-isolated view, a greater fanout might be compensated by a greater spatial accuracy of the entries' summaries which are stored in the accessed 'low-fanout' internal node. In fact, this is the assumption which underlies the utilization of the MBRQT^{c,a} approach as the fanout of the summary-like MBRQT^{c,a} R-trees is *guaranteed* to be lower than that of the summary-like MBR R-tree.

Table 56: Structural information on the summary-like R-trees in the T_5 scenario.

technique	height	# nodes [in M]	# int. nodes	avg mem. utilization int. nodes	# leaf nodes [in M]	avg mem. utilization leaf nodes	avg fanout
MBR	5	7.72	66,118	68.4%	7.65	69.9%	116.8
MBRQT_1	5	7.74	88,242	68.5%	7.65	69.9%	87.7
MBRQT_2	5	7.74	88,341	68.5%	7.65	69.9%	87.6
MBRQT_3	5	7.74	88,258	68.5%	7.65	69.9%	87.7
MBRQT_4	5	7.74	88,244	68.5%	7.65	69.9%	87.7
MBRQT_5	5	7.74	88,233	68.5%	7.65	69.9%	87.7
MBRQT_6	5	7.74	88,331	68.4%	7.65	69.9%	87.6
QTMBR_1	5	10.97	72,618	59.0%	10.90	49.1%	151.1
QTMBR_2	5	9.13	79,083	67.4%	9.05	59.1%	115.4
QTMBR_3	5	8.14	85,687	56.0%	8.06	66.4%	95.0
QTMBR_4	5	7.75	126,361	57.0%	7.63	70.1%	61.3

to the fanout, the one of QTMBR_1 is significantly greater (151.1 as opposed to 116.8, corresponding to +29.6%) while for QTMBR_2, the fanout is roughly the same as for the MBR approach. The fanouts of QTMBR_3 (95.0, -18.7%) and QTMBR_4 (61.3, -47.5%) are substantially lower. For all $\text{QTMBR}^{b}_{c,a}$ parameterizations, it shows that problems with the memory utilization rates of both internal nodes (QTMBR_1, QTMBR_3, and QTMBR_4 all feature less than 60%) as well as leaf nodes (especially for QTMBR_1 and QTMBR_2, modestly for QTMBR_3) arise. For the *leaf nodes*, it is evident that the less spatially accurate the corresponding $\text{QTMBR}^{b}_{c,a}$ summaries are (already see the indexed areas per level in Table 59 on page 352), the lower the memory utilization rate is. For example, QTMBR_1 indexes a cu-

[282]Note that in the T_5 scenario's summary-like R-trees, the indexing of lines or points does only happen in very rare cases for QTMBR_1 to QTMBR_3 and not nearly as often as for QTMBR_4. The numbers are as follows:

— QTMBR_1: 26 points, 308 lines,
— QTMBR_2: 329 points, 817 lines,
— QTMBR_3: 44 points, 958 lines, and
— QTMBR_4: 4,278,237 points, 3,481,735 lines.

Thus, the results are very similar to those of the corresponding MBR-like R-trees in the T_5 scenario (see footnote[273] on page 333). Overall, for QTMBR_1 to QTMBR_3, it *actually applies* that the smallest indexable spatial unit is a cell of the refined quadtree cell's quantization grid, and that points or iso-oriented lines are only described in the very rare cases when these points or lines are aligned with the quantization grid's hyperplanes. It can also be seen that QTMBR_2 has a greater tendency of describing points than QTMBR_1 and QTMBR_3 because it quantizes with $b = 12$ instead of $b = 8$.

mulated surface area of 18,346.6 dsu^2 and has an average leaf node memory utilization of only 49.1% whereas the numbers for QTMBR_3 are 655.6 dsu^2 and 66.4%. The problems with the memory utilization rates at leaf node level arise because for QTMBR_1 to QTMBR_3, the smallest indexable spatial units are too large for data space regions which are very densely populated with data points.[282] For these regions, many leaf nodes exist which are then described by exactly the same single area in the corresponding $\text{QTMBR}^b_{c,a}$ summaries. See Figure 105 for an example visualization. Consequently, when a new data point dp is inserted, the SELECTBEST(.)-method has to choose between many equally described candidate leaf nodes to store dp. This results in a random assignment as denoted in the description of the SELECTBEST(.)-method in section 10.2.1. However, our implementation is not truly random: Generally, the leaf nodes are considered in the order in which their entries are stored in their parent node. This means that a list of the equally well-suited candidate leaf nodes is always sorted by how the leaf node entries are ordered in the R-tree. If there is a set of candidates (i.e. the list's size is greater than 1), our implementation always selects the first element of the candidate list to store dp. This works well for the MBR and $\text{MBRQT}^{c,a}$ approaches as there are usually only very few equally well-suited candidates due to the 'infinite' maximum spatial accuracy of the MBR—but not for QTMBR_1 to QTMBR_3 with their large candidate sets (which result because of these techniques' comparatively coarse smallest indexable spatial units). Therefore, new data points located in very densely populated regions are always inserted into the same candidate leaf node *cln* which is occassionally split. The poorly-filled other candidates (which include the earlier split node siblings of *cln*) remain untouched. As a result of this, QTMBR_1 to QTMBR_3 have much more leaf nodes which are poorly-filled. Hence, these techniques are already at an disadvantage in comparison to the other techniques before even considering spatial properties: The less data points are stored in a leaf node on average, across the more leaf nodes the relevant data points are usually dispersed. Due to its enormous maximum spatial accuracy, the leaf node memory utilization problems do not apply to QTMBR_4: Its *leaf node* memory utilization rate is even marginally greater than for the MBR approach. However, the *internal nodes*' memory utilization rate of QTMBR_4 as well as of QTMBR_1 and QTMBR_3 is similarly degenerated. This phenomenon has the same causes as for the leaf node level: Their smallest indexable spatial units (which are much larger for the internal nodes than for the leaf nodes due to the restrictions imposed by target depth td) are too coarse for level-3-nodes and regions with a very high point density as they quan-

tize with only $b = 8$ bits.[283] QTMBR_2 with its more accurate smallest indexable spatial unit ($td = 2, b = 12$) can mitigate these problems almost completely for the internal nodes (67.4% average memory utilization). Nevertheless, future work has to be aware of this implementation detail.

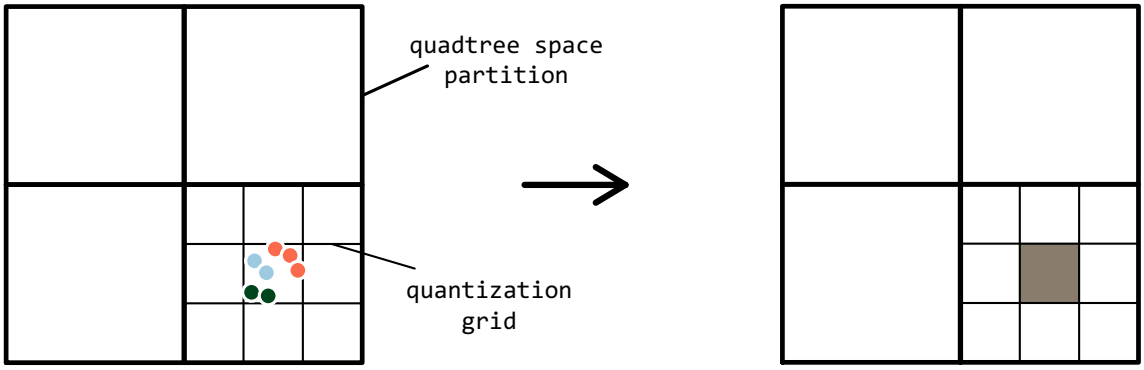


Fig. 105: Example visualization illustrating a situation in which for several nodes (let us assume red, green, and blue), exactly the same single area is indexed by their respective QTMBR$^b_{c,a}$ summaries. The data points of the red, the green, and the blue node are all located in the same cell of the quantization grid (left). This cell is the smallest indexable spatial unit of QTMBR$^b_{c,a}$ summaries in the given situation. Consequently, for each of the three nodes, the quantized MBR depicting its spatial footprint corresponds to this cell of the quantization grid (right).

In total, the QTMBR$^b_{c,a}$ parameterizations have (partially significantly) more nodes in their R-trees than the MBR approach and the MBRQT^{c,a} parameterizations. However, the height of all R-trees is still the same (5). In Table 57, the results for the internal nodes' memory consumption and the total number of indexed areas are outlined. The memory consumption values exhibit no surprises. Also, for all summary-like MBRQT^{c,a} R-trees,

[283] In [Henrich 1990, p. 163ff], Henrich showed that, depending on the general conditions, the memory utilization rate of bucket-based multidimensional data structures might cyclically rise up to considerably more than 69% and then drop significantly below again even if in the long-term, the memory utilization converges to 69%. Hence, the memory utilization rate may also vary simply depending on the current growth period of the structure. Thus, this could be an alternative explanation to the encountered anomalies. However, we think that these anomalies are rather caused by problems with the spatial accuracy and the non-random dissolution of ties because the anomalies only arise for summary-like QTMBR$^b_{c,a}$ R-trees in the T_5 and T_25 scenarios. There is no plausible reason why only these R-trees should be in an unfavorable growth period and not a single other R-tree in any other scenario (including the summary-like QTMBR$^b_{c,a}$ R-trees in the R_5 and R_25 scenarios). This is especially the case since overall, the R-trees are similarly 'grown-up' (despite the differences in their respective structures). For example, the summary-like QTMBR_3 R-tree has 98.94% leaf nodes in the T_5 scenario, 99.01% leaf nodes in the T_25 scenario, 99.05% leaf nodes in the R_5 scenario, and 98.87% leaf nodes in the R_25 scenario. Also, from the entirety of the internal nodes in the summary-like QTMBR_3 R-tree, 99.42% (T_5), 99.40% (T_25), 99.43% (R_5), and 99.42% (R_25) are at the level above leaf node level.

Table 57: Memory consumption of the internal nodes in the summary-like R-trees in the T_5 scenario. Additionally, the respective amounts of indexed areas are listed.

memory consumption internal nodes → technique ↓	avg [in B]	25%-quant. [in B]	median [in B]	75%-quant. [in B]	number of indexed areas
MBR	2,802.0	2,304	2,736	3,288	7,719,352
MBRQT_1	2,807.4	2,304	2,752	3,296	7,863,885
MBRQT_2	2,804.0	2,304	2,752	3,296	8,989,043
MBRQT_3	2,806.3	2,304	2,752	3,296	12,189,985
MBRQT_4	2,807.5	2,304	2,752	3,296	8,126,033
MBRQT_5	2,806.7	2,304	2,752	3,296	9,674,963
MBRQT_6	2,803.7	2,304	2,752	3,296	13,815,935
QTMBR_1	2,418.3	2,128	2,448	2,464	10,979,262
QTMBR_2	2,762.5	2,280	2,664	3,240	9,133,612
QTMBR_3	2,292.6	1,848	2,232	2,448	8,379,125
QTMBR_4	2,333.8	1,816	2,248	2,552	18,232,576

the numbers of indexed areas are very similar to their MBR-like counterparts. In contrast, the summary-like QTMBR$^b_{c,a}$ R-trees all exhibit significantly more indexed areas in comparison to their corresponding MBR-like R-trees. For QTMBR_1 to QTMBR_3, to large extents, this is caused by the greater amount of nodes in the respective summary-like R-trees. For example, the summary-like QTMBR_1 R-tree has 10,972,259 nodes and 10,979,262 indexed areas. Hence, like for the MBR-like QTMBR_1 R-tree in the T_5 scenario (7,720,602 nodes and 7,727,228 indexed areas), a node is only very rarely described by more than one indexed area. However, for QTMBR_4, it does *not* apply that the surplus on indexed areas is mainly caused by the surplus on nodes in the R-tree: Its summary-like R-tree now has about 30,000 (or 0.4%) more nodes[284] but simultaneously, there are also roughly 2.66 million (or 17%) more indexed areas. In general, the majority of indexed areas are from the leaf nodes' summaries: Due to the restrictions imposed by target depth td, the internal nodes' QTMBR_4 summaries index way less areas on average than the leaf nodes' QTMBR_4 summaries. Furthermore, 98.4% of the nodes in the QTMBR_4 R-tree are leaf nodes. Thus, since the overall amount of leaf nodes in the summary-like QTMBR_4 R-tree is even slightly lower compared to the corresponding MBR-like QTMBR_4 R-tree, this surplus of indexed areas implies a greater spatial spread of the data points in the leaf nodes (because usually, a greater spatial spread leads to more indexed areas). Hence, it is an indication that the assignment of newly inserted data points to leaf nodes by using QTMBR_4 summaries (or QTMBR$^b_{c,a}$ summaries in general) dur-

[284] This surplus is caused by the greater amount of internal nodes. The amount of leaf nodes is now even slightly lower due to their greater average memory utilization (70.1% as opposed to 69.9%).

ing the R-tree construction is not as selective as by using MBR summaries. This would be no surprise as the SELECTBEST(.)-method is not adapted to the special properties of the QTMBR_4 summaries and therefore, especially at upper levels of the R-tree, suboptimal subtrees might be chosen for the insertion of new data points.

Table 58: Summary-related results for the summary-like R-trees in the T_5 scenario.

technique	avg memory consumption [in B]	avg memory utilization	cblq-nz	cblq-z	sum-nz/ lq-nz	sum-z/ lq-z
MBR	16.0	100%	-	-	100%	-
MBRQT_1	24.0	67.8%	99.4%	-	0.6%	-
MBRQT_2	24.0	68.7%	99.0%	-	1.0%	-
MBRQT_3	24.0	71.2%	99.0%	-	1.0%	-
MBRQT_4	24.0	68.0%	99.1%	-	0.9%	-
MBRQT_5	24.0	69.7%	91.7%	-	8.3%	-
MBRQT_6	24.0	74.5%	68.9%	-	31.1%	-
QTMBR_1	8.0	89.0%	0.01%	-	100.0%	-
QTMBR_2	15.9	57.2%	$<$0.01%	-	100.0%	-
QTMBR_3	16.1	58.5%	0.06%	-	99.9%	-
QTMBR_4	30.0	87.8%	29.8%	$\ll$0.01%	70.2%	-

The summary-related results are depicted in Table 58. They are almost identical to the summary-related results of the *MBR-like* R-trees in the T_5 scenario. Obviously, the summaries themselves are hardly influenced by the constructional differences between the summary-like and the MBR-like R-trees. Thus, the observed congruence between the summary-related results of the summary-like and the MBR-like R-trees can also be expected for the other scenarios. As a consequence, we skip the assessment of the summary-related results for the subsequent scenarios.
Table 59 showcases the results for the cumulated indexed surface areas per level. In general, due to the different numbers of nodes per level in the respective R-trees, the indexed surface areas of the summary-like R-trees are not as easily comparable to each other as for the MBR-like R-trees. Therefore, in Table 60, the amounts of nodes at the different levels of the respective R-trees are depicted to support the assessment of the numbers listed in Table 59. Surprisingly, for example at level 1, the *summary-like* MBR R-tree now has 6,740.8 dsu^2 (or rougly 7.9%) less indexed surface

[285] Both R-trees have five nodes at level 1, i.e. the differences are not caused by a varying amount of nodes.

area in comparison to the *MBR-like* MBR R-tree.[285] At leaf node level, the summary-like R-tree indexes 6.3% less surface area. As discussed at the beginning of the current scenario's assessment, the same prerequisites apply to both of these specific R-trees, and differences are only caused by randomized components. Hence, this shows that randomness has non-negligible impacts on the results. In addition to the general difficulty of an analysis due to the varying R-tree structures, this observation is another reason we do not want to go too much into detail for the assessment of the cumulated indexed surface areas of the summary-like R-trees to avoid an overinterpretation of the results. However, the results reveal that still, substantial amounts of overlap exist between the internal nodes' summaries at all levels of the respective R-trees. Especially for the MBRQT^{c,a} parameterizations, the single levels now often feature greater amounts of indexed surface areas than the MBR approach, i.e. there is more overlap between the nodes in total.

In a comparison of the summary-like and the MBR-like R-trees, at leaf node level, the changes for the cumulated indexed surface areas of the single techniques are within the range of expectations—except for QTMBR_4. Also, no significant changes in the 'balance of power' *between* the single techniques are evident. Nevertheless, the overall results for the summary-like QTMBR_4 R-tree (almost 80% increase for the cumulated indexed surface areas at leaf node level and the already mentioned 17% increase for the number of indexed areas in comparison to the MBR-like QTMBR_4 R-tree) imply, again, that the assignment of data points to leaf nodes by using QTMBR$^{b}_{c,a}$ summaries in the R-tree construction phase might not be as targeted as by using the MBR approach.

In total, for the summary-like R-trees, the combination of a greater number of internal nodes, a greater amount of leaf nodes (in some cases), a lower fanout (in most cases), and less favorable results with regard to the cumulated indexed surface areas indicate that it will be more difficult for our summarization approaches to achieve improvements over the MBR approach than for the MBR-like R-trees. Of course, this is also the reasonable a priori assumption as the MBR-like R-trees do not consider any storage space limitations (which favors all MBRQT^{c,a} and most of the QTMBR$^{b}_{c,a}$ parameterizations as they consume more storage space than the MBR approach). Furthermore, the SELECTBEST(.)-method and the R*-split still fit best for MBR summaries, and no algorithms are applied which have been specifically optimized for MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summaries. Nevertheless, the cumulated indexed surface areas at leaf node level show that the identified weakness of the MBR—its lack of spatial accuracy—can be tackled successfully in the centralized application scenario, in general, and also by summary-like R-trees. The evaluation has to show whether these improvements also pay off in the query performances.

However, as discussed before, the cumulated indexed surface areas are not really a very suitable means for assessing the *average* indexing per-

Table 59: Cumulated indexed surface areas of the summary-like R-trees in the T_5 scenario (all values in dsu^2).

technique	level 1	level 2	level 3	level 4 (leaf nodes)
MBR	78,427.4	92,255.0	80,020.4	17,569.9
MBRQT_1	75,932.4	95,063.9	80,097.8	9,225.2
MBRQT_2	87,832.2	107,546.7	88,486.1	10,887.6
MBRQT_3	86,155.4	106,694.1	86,977.2	9,745.5
MBRQT_4	82,402.4	95,943.4	66,259.4	2,907.6
MBRQT_5	82,616.9	94,188.0	65,277.2	2,999.0
MBRQT_6	84,257.9	101,237.1	67,488.8	3,032.7
QTMBR_1	73,256.4	93,846.3	88,668.6	18,348.6
QTMBR_2	76,324.6	91,105.2	77,784.4	6,264.3
QTMBR_3	74,707.7	87,559.3	83,682.4	655.6
QTMBR_4	69,291.7	78,909.4	71,668.0	1.6

Table 60: Nodes per level of the summary-like R-trees in the T_5 scenario.

technique	level 1	level 2	level 3	level 4 (leaf nodes)
MBR	5	573	65,539	7,653,235
MBRQT_1	11	1,003	87,227	7,653,239
MBRQT_2	11	998	87,331	7,652,612
MBRQT_3	12	989	87,256	7,651,770
MBRQT_4	10	978	87,255	7,653,812
MBRQT_5	13	1,002	87,217	7,650,745
MBRQT_6	12	994	87,324	7,650,822
QTMBR_1	3	422	72,192	10,899,641
QTMBR_2	3	449	78,630	9,047,936
QTMBR_3	3	496	85,187	8,055,697
QTMBR_4	5	721	125,634	7,625,175

formance of the respective techniques. Therefore, Figure 106 depicts the distributions of the leaf nodes' indexed surface area per node for examplarily selected techniques and the currently assessed T_5 scenario's summary-like R-trees. In general, they are fairly similar to the corresponding distri-

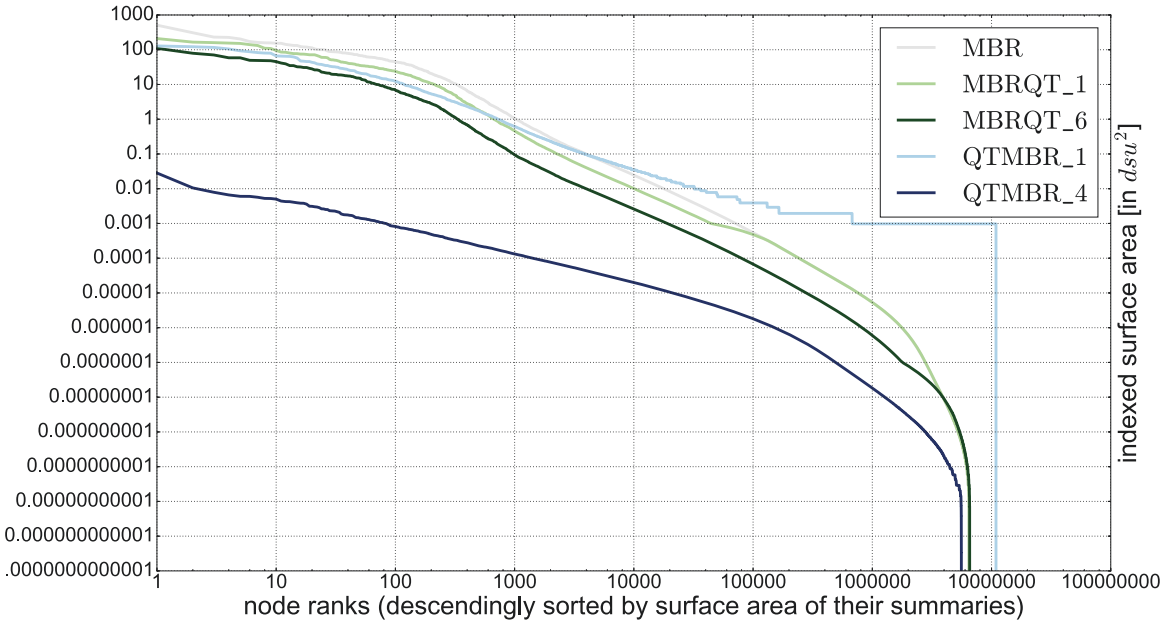


Fig. 106: Leaf nodes sorted in descending order by their indexed surface areas (summary-like R-trees in the T_5 scenario). Both axes are log-scaled.

butions for the MBR-like R-trees in the T_5 scenario (see Figure 100 on page 331). Aside from the significantly greater amount of leaf nodes for QTMBR_1 and the resulting consequences on its curve, the most noteworthy change concerns QTMBR_4: Its curve hits the x-axis clearly later in comparison (even though there are 29,258 less leaf nodes in the summary-like QTMBR_4 R-tree), i.e. there are less nodes with a zero volume surface area now. Also see Figure 107, where the curves of the MBR-like and the summary-like QTMBR_4 R-tree are juxtaposed. Aside from the first few nodes, the curve of the summary-like QTMBR_4 R-tree is always above the MBR-like QTMBR_4 R-tree's curve, i.e. the overwhelming majority of its leaf node summaries are coarser. Furthermore, its first zero volume leaf node is only at rank 5,584,259 (as opposed to rank 4,934,867). Once again, this indicates a degeneration of the summary-like QTMBR_4 R-tree which is caused by the unsuitable SELECTBEST(.)-method.
In this context, it has to be noted that the distribution of the indexed surface area per node is a very suitable means of comparison for *MBR-like* R-trees but might present a misleading picture of the relative indexing perfomances for *summary-like* R-trees. This is because the assignment of data points to nodes differs between the respective summary-like R-trees. As previously shown, especially QTMBR_4 is extremely successful with minimizing the indexed surface areas (also already see e.g. Figure 111 on page 364). However, even though QTMBR_4 offers smaller indexed surface areas than the MBR approach for all leaf node ranks as depicted in Figure 106, the indexation of the leaf nodes does not necessarily have to be more appropriate for QTMBR_4 than for the MBR approach. Consider Figure 108 as an example: On the left, there is an $\text{QTMBR}^{b}_{c,a}$ summary which overall indexes clearly less surface area than the MBR summary on the right.

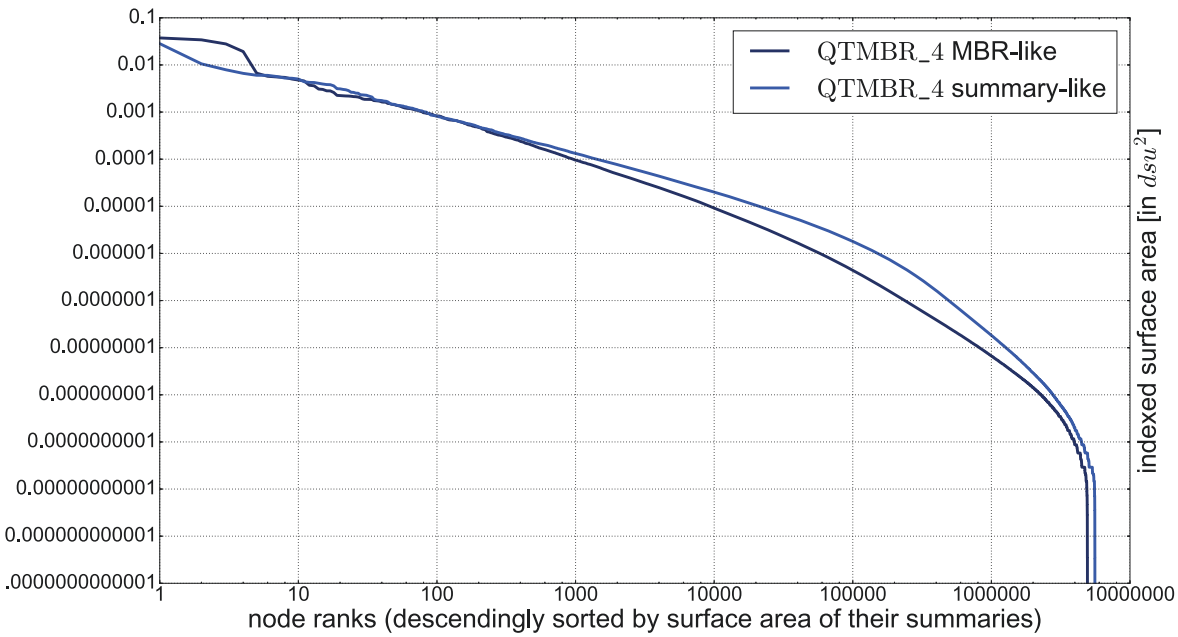


Fig. 107: Comparison of the sorted leaf nodes' indexed surface areas for the MBR-like and the summary-like QTMBR_4 R-tree, both in the T_5 scenario. Both axes are log-scaled.

However, on the left, the data points are widely scattered throughout the data space whereas on the right, they are much better clustered. We argue that even though the indexed surface area is greater, the illustrated MBR summary represents a more appropriate leaf node indexation as it has to be considered that there are also other leaf nodes whose data point clouds might be intermixed with the depicted data point clouds. Additionally, it is self-evident that 'clustered' data points should be kept in the same leaf nodes as this is beneficial with regard to the amount of leaf node accesses when querying. However, we have doubts that for the $\text{QTMBR}^{b}_{c,a}$ parameterizations and maybe also for the $\text{MBRQT}^{c,a}$ parameterizations, clustered data points are always kept close together in the summary-like R-trees.

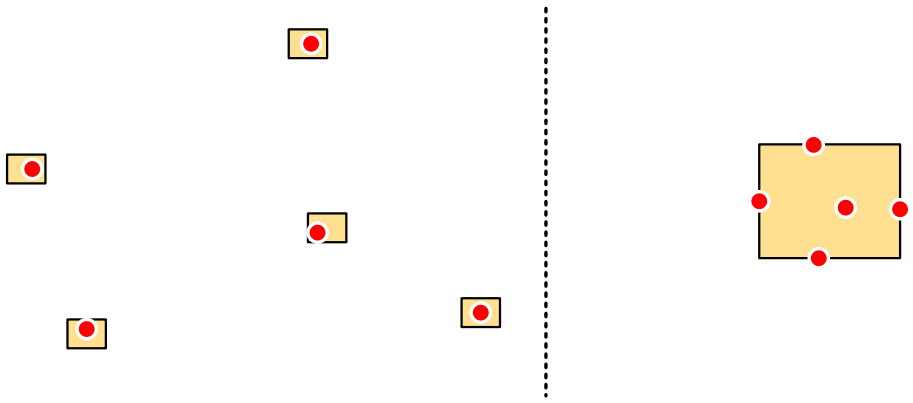


Fig. 108: Exemplary visualization illustrating the potential inappropriateness of the distribution of the indexed surface area per node as key figure for the summary-like R-trees. On the left, an $\text{QTMBR}^{b}_{c,a}$ summary is shown while on the right, an MBR summary is depicted.

This is also evident from Figure 109 which, in general, shall allow to compare the techniques for the spatial *scattering* of the data points stored in the leaf nodes, i.e. it shall display how well clustered data points are held together. Anew, the usual exemplarily selected techniques are displayed. To create the figure, we proceeded as follows for each technique: For each leaf node in the respective R-tree, the maximum distance md between any pair of the data points stored in this very leaf node is calculated. Hence, we have a maximum distance md for each leaf node. The set of leaf nodes is then sorted in descending order by md. Afterwards, the node ranks are normalized, i.e. mapped to the interval [0;1]. Finally, a curve is created for each technique: The y-axis shows the log-scaled maximum distances md. The x-axis displays the normalized node ranks on a linear scale. Consequently, the curves serve as a measure for how well the data points are clustered in the leaf nodes: The lower the curve, the better data point clusters are held together in the leaf nodes. A comparison between the techniques shows that evidently, the spatial scattering of the leaf nodes' data points is greatest in the QTMBR$^{b}_{c,a}$ R-trees.
Hereby, it is difficult to compare the QTMBR_1 R-tree to the other R-trees as its amount of leaf nodes is much greater than those of the other R-trees. However, its curve suggests that the spatial scattering is greatest in the QTMBR_1 R-tree's leaf nodes even though the average amount of data points stored in them is the lowest by far (only 49.1% leaf node memory utilization). For the remaining techniques, the amount of leaf nodes is nearly the same (about 7.63 million for QTMBR_4 and roughly 7.65 million for the remaining techniques, see Table 56 on page 346). It is clearly evident that for QTMBR_4, the spatial scattering is significantly greater than for the MBR approach and the MBRQT^{c,a} parameterizations. For the latter, it interestingly shows that MBRQT_6 exhibits a slightly greater spatial scattering than MBRQT_1. The MBR approach is minimally better than MBRQT_1 (which can only be seen when Figure 109 is viewed in a high zoom level). Hence, also the MBRQT^{c,a} parameterizations exhibit a greater spatial scattering than the MBR approach. In general, the assignment of data points to leaf nodes is the task of the SELECTBEST$(.)$-method, i.e. eventual misassignments of data points to leaf nodes are attributable to this method. Figure 109 indicates that for QTMBR$^{b}_{c,a}$, the leaf nodes' spatial scattering is higher than for the other approaches (at least in the T_5 scenario). This implies that the SELECTBEST$(.)$-method is unsuitable for QTMBR$^{b}_{c,a}$. Also the previously discussed total amount of indexed areas in the summary-like QTMBR_4 R-tree had similar implications. But also for MBRQT^{c,a}, the SELECTBEST$(.)$-method does not seem 100% appropriate as the MBRQT^{c,a} parameterizations' spatial scattering is also greater than that for the MBR approach. Hereby, it is interesting that the technique with the spatially more accurate summaries (MBRQT_6) exhibits a greater scattering than the one with the spatially less accurate summaries (MBRQT_1). We keep track of these interesting observations in the subse-

quent structural analyses. Usually, a greater spatial scattering of the leaf nodes' data points should be disadvantageous for the query performance. However, in the end, only the query results can show how large possible adverse effects turn out to be.

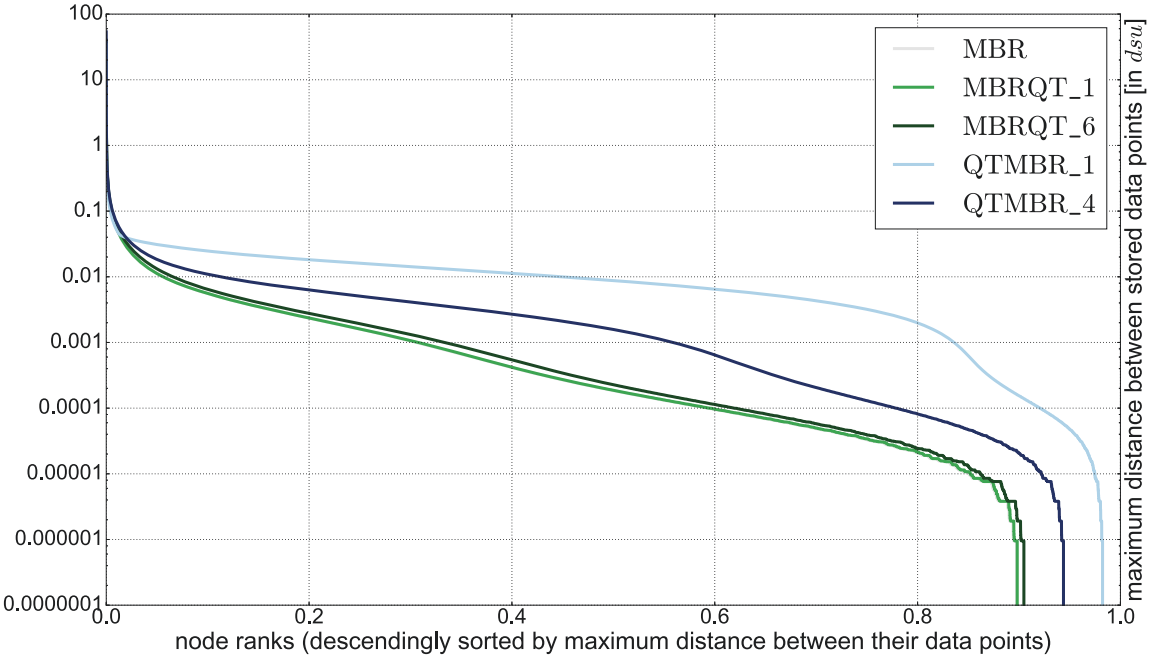


Fig. 109: Leaf nodes sorted in descending order by the maximum distance between the data points they store (for the T_5 scenario's summary-like R-trees). The node ranks are normalized. The x-axis is linear-scaled whereas the y-axis is log-scaled.

Structural Analysis for the Summary-like R-trees in the T_25 scenario

In the following, the T_25 scenario 's summary-like R-trees are assessed. Table 61 shows the structural information on the respective R-trees. For the first time, the heights of the R-trees are varying: For the MBR approach and QTMBR_1 to QTMBR_3, the height is 4 whereas for the remaining techniques, the height is 5. This is disadvantageous for the latter as the path to the leaf node level is longer.

For the $\text{QTMBR}^{b}_{c,a}$ parameterizations, the internal and leaf nodes' memory utilization rates are generally a little better now but still not as good as for the MBR approach and the $\text{MBRQT}^{c,a}$ parameterizations. The greater leaf node capacity leads to a greater spatial spread of the data points stored in a leaf node which obviously *generally* mitigates the problems leading to the low average memory utilization rates for the $\text{QTMBR}^{b}_{c,a}$ parameterizations. Nevertheless, as before, there are (partly substantially) more leaf nodes for QTMBR_1 to QTMBR_3 in comparison to the MBR approach. In general, also the amounts of internal nodes are greater for our summarization approaches than for the MBR approach. Only QTMBR_1 has less internal nodes—in spite of having the by far greatest number of leaf

Table 61: Structural information on the summary-like R-trees in the T_25 scenario.

technique	height	# nodes [in M]	# int. nodes	avg mem. utilization int. nodes	# leaf nodes [in M]	avg mem. utilization leaf nodes	avg fanout
MBR	4	1.58	13,518	68.3%	1.56	68.5%	116.5
MBRQT_1	5	1.58	17,938	68.8%	1.56	68.5%	88.1
MBRQT_2	5	1.58	17,966	68.7%	1.56	68.5%	87.9
MBRQT_3	5	1.58	18,064	68.3%	1.56	68.5%	87.5
MBRQT_4	5	1.58	17,946	68.8%	1.56	68.5%	88.0
MBRQT_5	5	1.58	17,965	68.7%	1.56	68.5%	87.9
MBRQT_6	5	1.58	18,012	68.5%	1.56	68.5%	87.7
QTMBR_1	4	1.88	11,758	62.6%	1.87	57.2%	160.1
QTMBR_2	4	1.66	14,134	68.5%	1.64	65.1%	117.2
QTMBR_3	4	1.61	15,960	60.8%	1.59	67.2%	100.7
QTMBR_4	5	1.60	33,500	60.6%	1.56	68.4%	47.7

nodes. This is due to its significantly greater fanout (160.1 on average as opposed to 116.5 for the MBR approach). In principle, the R-trees' fanouts are very stable in comparison to the T_5 scenario's summary-like R-trees. Solely for QTMBR_4 (-22.2% fanout), significant changes can be observed. As the leaf node capacity is now 25 instead of 5, the QTMBR_4 leaf node summaries become more complex and therefore require even more storage space, leading to the decreased fanout for QTMBR_4. For the other parameterizable techniques, there are only slight changes despite the greater leaf node capacity (either due to their limited spatial accuracy, or the byte- and word-alignment).

The results for the memory consumption of the internal nodes are depicted in Table 62. Obviously, the changes in the memory consumption go hand in hand with the nodes' memory utilization rates. In total, no unusual results are evident. For the number of indexed areas (rightmost column of Table 62), the only thing worth mentioning is that for the summary-like QTMBR_4 R-tree in the T_25 scenario, there are 1.2 million (or 18%) more indexed areas in comparison to the corresponding MBR-like QTMBR_4 R-tree—even though featuring merely 20,000 more internal nodes (the number of leaf nodes is roughly 1.56 million for both R-trees). Anew, this surplus of 1.2 million indexed areas cannot solely be explained by the additional number of internal nodes and thus has to be co-created by the leaf nodes for which now, obviously, more areas are indexed in comparison. Again, this implies a greater spatial spread of the data point sets in the leaf nodes. Hence, it is a further indication that the assignment of data points to nodes

Table 62: Memory consumption of the internal nodes in the summary-like R-trees in the T_25 scenario. Additionally, the respective amounts of indexed areas are listed.

memory consumption internal nodes → technique ↓	avg [in B]	25%-quant. [in B]	median [in B]	75%-quant. [in B]	number of indexed areas
MBR	2,796.6	2,304	2,736	3,288	1,575,182
MBRQT_1	2,817.8	2,304	2,752	3,328	1,908,645
MBRQT_2	2,812.8	2,304	2,752	3,328	3,916,860
MBRQT_3	2,799.1	2,304	2,752	3,296	5,209,801
MBRQT_4	2,816.6	2,336	2,752	3,328	2,137,405
MBRQT_5	2,813.5	2,304	2,752	3,328	5,266,430
MBRQT_6	2,806.6	2,304	2,752	3,296	7,771,532
QTMBR_1	2,565.9	2,120	2,464	2,944	1,888,995
QTMBR_2	2,807.6	2,328	2,736	3,312	1,663,064
QTMBR_3	2,492.1	1,936	2,376	2,936	1,806,343
QTMBR_4	2,480.7	1,840	2,368	2,992	7,856,048

in the R-tree construction phase is less targeted when using our summarization approaches.
In Table 63, the cumulated indexed surface areas per level are showcased. Table 64 depicts the amounts of nodes at the different levels of the respective summary-like R-trees. Due to the varying heights of the R-trees, for the internal nodes' levels, it makes only sense to directly compare techniques whose R-trees have the same height. As before, QTMBR_1 to QTMBR_3 struggle against the MBR approach on the internal nodes' levels. Also, QTMBR_4 is worse than the MBRQT^{16,a} parameterizations at level 3. Hence, the relative results show more of the already known patterns.
In general, in comparison to the T_25 scenario's *MBR-like* R-trees, the MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ parameterizations' relative improvements over the MBR approach with regard to the cumulated indexed surface areas at leaf node level are similar but slightly lower in tendency. See Table 65 for a juxtaposition of the concrete values. This could be a further *subtle* hint that the assignment of data points to leaf nodes is less targeted when constructing R-trees by use of our summarization approaches. Nevertheless, the differences are not as big that they could not also have been caused by randomness.
To evaluate this further, Figure 110 shows the distributions of the spatial scattering per leaf node for the exemplarily selected techniques. Since the leaf node memory utilization rate for QTMBR_1 increased in comparison to the T_5 scenario (from 49.1% to 57.2%), the amounts of leaf nodes are no longer as divergent as before. Consequently, the curves are better comparable now. Nonetheless, the curves still imply that QTMBR_1 features the greatest spatial scattering of all techniques. For about the

Table 63: Cumulated indexed surface areas of the summary-like R-trees in the T_25 scenario (all values in dsu^2). The numbers of the respective leaf node levels are in bold print.

technique	level 1	level 2	level 3	level 4
MBR	82,813.7	84,056.9	**49,697.9**	-
MBRQT_1	69,742.0	89,027.9	85,041.7	**40,816.9**
MBRQT_2	70,973.5	100,539.1	95,839.5	**43,111.3**
MBRQT_3	76,016.2	95,097.3	90,131.6	**41,959.9**
MBRQT_4	67,002.9	84,430.5	73,595.4	**22,321.3**
MBRQT_5	72,824.6	90,499.4	77,005.9	**23,693.4**
MBRQT_6	67,665.2	84,422.9	76,105.7	**22,355.8**
QTMBR_1	86,491.7	88,276.2	**29,261.4**	-
QTMBR_2	82,824.9	82,909.5	**25,338.6**	-
QTMBR_3	83,161.9	86,173.0	**6,755.6**	-
QTMBR_4	67,201.6	83,222.3	79,994.9	**479.9**

Table 64: Nodes per level of the summary-like R-trees in the T_25 scenario.

technique	level 1	level 2	level 3	level 4
MBR	120	13,397	**1,561,665**	-
MBRQT_1	2	201	17,734	**1,561,590**
MBRQT_2	2	203	17,760	**1,561,219**
MBRQT_3	3	207	17,853	**1,562,002**
MBRQT_4	2	200	17,743	**1,561,655**
MBRQT_5	2	203	17,759	**1,561,550**
MBRQT_6	2	203	17,806	**1,561,770**
QTMBR_1	73	11,684	**1,870,551**	-
QTMBR_2	84	14,049	**1,641,833**	-
QTMBR_3	95	15,864	**1,591,175**	-
QTMBR_4	2	190	33,307	**1,563,943**

first 10% of the node ranks, QTMBR_1 is actually fairly competitive and clearly better than QTMBR_4. It then falls behind all the other techniques, though. Overall, the results are quite stable: QTMBR_4 is still significantly worse than the MBR approach, MBRQT_1, and MBRQT_6. MBRQT_1 is yet slightly better than MBRQT_6. Hereby, the superiority of MBRQT_1 is apparently increased. The same goes for the MBR approach and MBRQT_1:

Table 65: Comparison of the tested techniques' percentage improvements over the MBR approach with regard to the cumulated indexed surface areas at leaf node level (for both the summary-like and the MBR-like R-trees in the T_25 scenario).

	MBR	MBRQT_1	MBRQT_2	MBRQT_3	MBRQT_4	MBRQT_5	MBRQT_6	QTMBR_1	QTMBR_2	QTMBR_3	QTMBR_4
summary-like, T_25	49,698 dsu^2	17.9%	13.3%	15.6%	55.1%	52.3%	55.0%	41.1%	49.0%	86.4%	99.0%
MBR-like, T_25	49,658 dsu^2	21.5%	22.0%	21.6%	52.4%	52.5%	52.5%	42.8%	49.5%	88.1%	99.0%

It is now clearly recognizable that the MBR approach's curve is below that of MBRQT_1. Obviously, the leaf nodes' (respectively their data points') greater spatial spread in the T_25 scenario amplifies the differences between the full-precision MBR-based techniques. Hereby, it applies anew that—given similar amounts of leaf nodes—the spatially more accurate the corresponding summaries should be in theory, the greater the spatial scattering in the leaf nodes. This is interesting in two respects: For once, it implies that the SELECTBEST(.)-method is the less appropriate the greater the benefit from more accurate summaries should be. Second, two conflicting aspects might impact the query results: On the one hand, we have a (theoretically) greater spatial accuracy of diverse techniques as opposed to others (such as from MBRQT_6 in comparison to the MBR approach), as indicated by the cumulated indexed surface areas and the distributions of the indexed surface area per node. On the other hand, the spatially more accurate techniques might suffer from a greater spatial scattering of the leaf nodes' data points. The query results have to show how the trade-off between both aspects turns out to be.

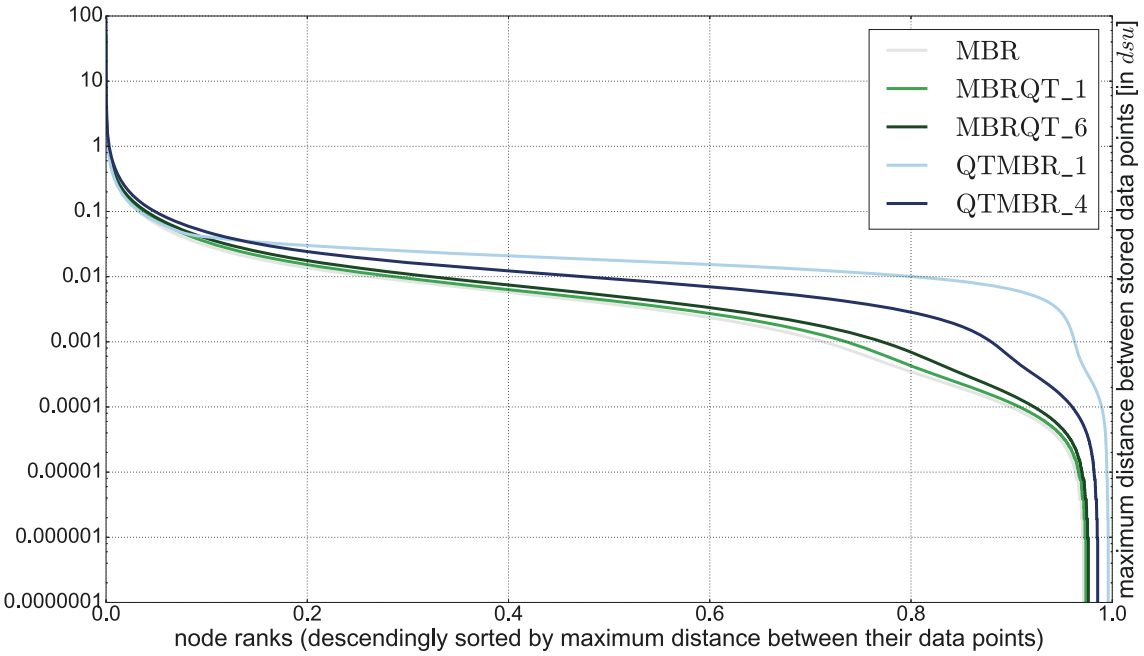


Fig. 110: Leaf nodes sorted in descending order by maximum distance between stored data points (for the T_25 scenario's summary-like R-trees). The node ranks are normalized. The x-axis is linear-scaled whereas the y-axis is log-scaled.

Structural Analysis for the Summary-like R-trees in the R_5 scenario

We now briefly analyze the results for the R_5 scenario's summary-like R-trees. All the R-trees have a height of 5. Table 66 shows that for all $\text{QTMBR}^{b}_{c,a}$ parameterizations, the memory utilization rates of the internal nodes are now on par with the other techniques. This also applies to the leaf node level where the memory utilization rates are nearly identical for all techniques. In contrast to the T collection, in the R collection, neither duplicate data points nor extremely densely populated data regions exist. Hence, the problems caused by the too coarse smallest indexable spatial units of the $\text{QTMBR}^{b}_{c,a}$ parameterizations do not occur here and therefore, the memory utilization rate problems are generally restricted to the T collection. As a consequence of the leaf nodes' nearly identical average memory utilization rates, their amount is almost the same in all R-trees. Since the number of internal nodes is negatively correlated with the fanout and in that regard, still a lot of disparity exists between the different techniques, there are between 39,287 (QTMBR_1, fanout 177.2) and 107,809 (QTMBR_4, fanout 64.5) internal nodes in the respective R-trees. For the summary-like R-trees, the fanouts are generally higher in the R_5 scenario compared to the T_5 scenario. The increase is greatest for QTMBR_1 (17.3%), QTMBR_3 (10.8%), and QTMBR_4 (5.2%). Obviously, this is because their internal nodes' memory utilization rates rise significantly from the T_5 scenario to the R_5 scenario (for example, the internal node memory utilization rate of QTMBR_1 improves from 59.0% to 69.7%). For the other techniques, rather marginal increases occur which match the

slight increases for the internal nodes' memory utilization rates in the R_5 scenario. We suppose that this is an effect of the *absence* of outlier data points or outlier data point clusters which can lead to slightly reduced fill rates for the internal nodes, too.

Table 66: Structural information on the summary-like R-trees in the R_5 scenario.

technique	height	# nodes [in M]	# int. nodes	avg mem. utilization int. nodes	# leaf nodes [in M]	avg mem. utilization leaf nodes	avg fanout
MBR	5	6.91	58,953	68.7%	6.86	72.9%	117.3
MBRQT_1	5	6.93	78,521	69.0%	6.85	73.0%	88.3
MBRQT_2	5	6.93	78,596	68.9%	6.85	73.0%	88.2
MBRQT_3	5	6.93	78,588	68.9%	6.85	73.0%	88.2
MBRQT_4	5	6.93	78,529	68.9%	6.85	73.0%	88.2
MBRQT_5	5	6.93	78,581	68.9%	6.85	73.0%	88.1
MBRQT_6	5	6.93	78,450	69.0%	6.85	73.0%	88.3
QTMBR_1	5	6.96	39,287	69.7%	6.92	72.2%	177.2
QTMBR_2	5	6.92	58,947	68.9%	6.86	72.9%	117.4
QTMBR_3	5	6.92	65,785	69.8%	6.86	72.9%	105.2
QTMBR_4	5	6.96	107,809	69.5%	6.85	73.0%	64.5

The results displayed in Table 67 reveal no apparent anomalies: The descriptive statistic values for the internal nodes' memory consumption are very similar for all techniques. Also, for the amounts of indexed areas, the relative results between the techniques are within the expectations. The $\text{QTMBR}^{b}_{c,a}$ parameterizations except QTMBR_4 index significantly less areas than the $\text{MBRQT}^{c,a}$ parameterizations which is obviously because for the leaf nodes' summaries, the maximum number of quadtree cells c depletes before more black quadtree cells can be built ($\rightarrow c = 16$ for QTMBR_1 and QTMBR_2, $c = 32$ for QTMBR_3). With regard to the absolute numbers of indexed areas, especially MBRQT_5, MBRQT_6, and QTMBR_4 exhibit very large numbers of more than 24 million. As most of the indexed areas belong to the leaf node summaries, this means that most of the 25 million data points are indexed by an area of their own.[286] Figure 111 shows an exemplary depiction of a situation where the five data points of a leaf node are indexed with five areas by the corresponding $\text{QTMBR}^{b}_{c,a}$ summary.

[286]Note that this is similar for the MBR-like R-trees in the R_5 scenario but has not been explicitly mentioned in the corresponding assessment.

Table 67: Memory consumption of the internal nodes in the summary-like R-trees in the R_5 scenario. Additionally, the respective amounts of indexed areas are listed.

memory consumption internal nodes → technique ↓	avg [in B]	25%-quant. [in B]	median [in B]	75%-quant. [in B]	number of indexed areas
MBR	2,815.1	2,328	2,736	3,288	6,914,983
MBRQT_1	2,824.8	2,336	2,752	3,296	18,058,902
MBRQT_2	2,822.2	2,336	2,752	3,296	20,896,901
MBRQT_3	2,822.3	2,336	2,752	3,296	20,927,926
MBRQT_4	2,822.4	2,336	2,752	3,296	20,950,566
MBRQT_5	2,820.8	2,336	2,752	3,296	24,087,178
MBRQT_6	2,824.8	2,336	2,752	3,296	24,116,165
QTMBR_1	2,854.8	2,368	2,792	3,344	7,058,995
QTMBR_2	2,821.8	2,328	2,760	3,312	7,017,437
QTMBR_3	2,859.9	2,376	2,800	3,352	10,149,644
QTMBR_4	2,846.9	2,360	2,792	3,336	24,494,878

With respect to computational efficiency, such an indexation is not reasonable as point-to-rectangle distance calculations are more expensive than point-to-point distance calculations. Further drawbacks in comparison to a direct representation of the data points are a costlier deserialization and a lower spatial accuracy because of the description of non-zero-volume rectangles instead of points.[287] The only advantage summaries can offer in such situations is that they might consume less storage space than the direct representation. This would then lead to an increased fanout as the internal nodes above leaf node level could store more leaf node entries. Nevertheless, Table 68 shows that the QTMBR_4 summaries consume more memory on average (33.85 B) than the direct representation of the data points would require (approximately 29.18 B[288])—even when disregarding the byte- and word-alignment which has to be applied for the sum-

[287] Due to the concrete parameterizations of the MBRQT^{c,a} parameterizations (i.e. the composition of the values for the parameters c and a), they do generally not run into the limits of the single precision floating-point number format (as opposed to QTMBR_4). Consequently, MBRQT_5 respectively MBRQT_6 index only 558 respectively 1,545 lines (and no points) in their summary-like R-trees in the R_5 scenario, i.e. the overwhelming majority of the indexed areas are actually rectangles for MBRQT_5 and MBRQT_6. For the summary-like QTMBR_4 R-tree in the R_5 scenario, 109,338 points and 1,189,155 lines are indexed. Hence, the indexing of points and lines is less pronounced for the `R` collection than for the `T` collection, in general. However, even if the indexed rectangles are actually points and lines, they are still described as rectangles and thus, point-to-rectangle distance calculations are conducted.

[288] This value arises as *data_point_size* · (*leaf_node_capacity* · *average_leaf_node_memory_utilization*) or $8\,B \cdot (5 \cdot 0.7295) = 29.18\,B$. Hereby, we calculate with 0.7295 as *average_leaf_node_memory_utilization* because for all the R-trees in the R_5 scenario, the leaf node memory utilization rate is 72.9% or 73.0% (see Table 66), i.e. we take the mean of the two occuring values.

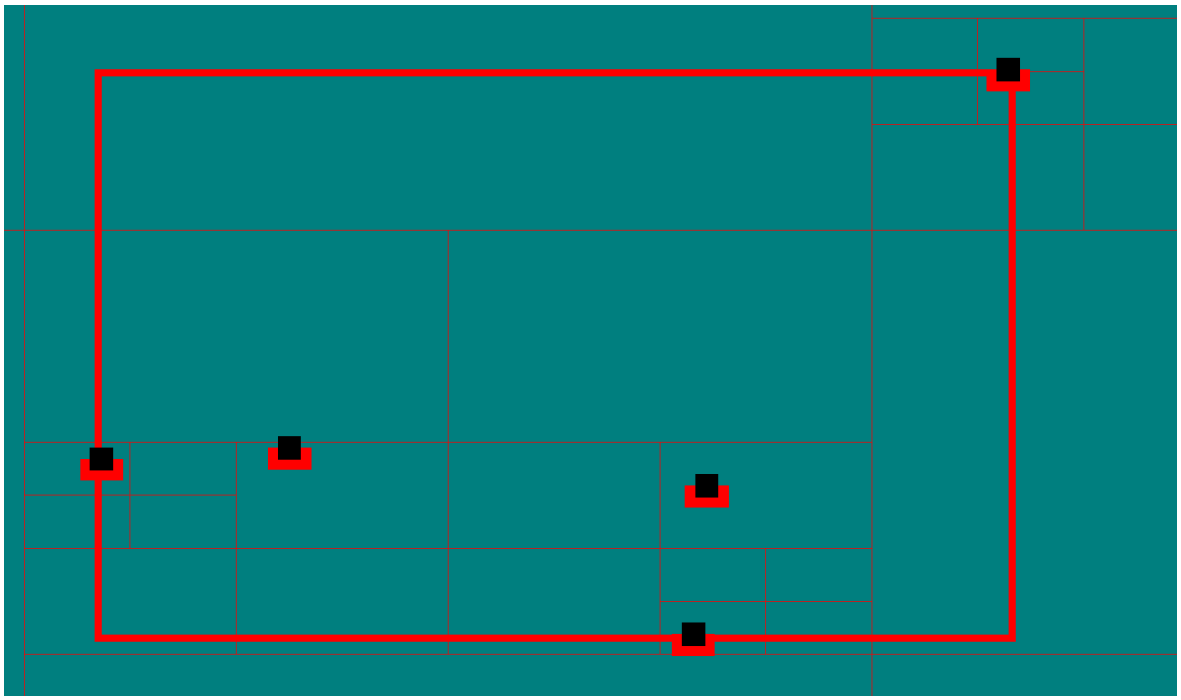


Fig. 111: Visualization of an example leaf node's summary in the R_5 scenario's summary-like QTMBR_4 R-tree. The leaf node stores five data points. The black squares denote the data points' locations. Each data point is delineated with a quantized MBR of its own (small red rectangles adjacent to the points). The border lines of these indexing MBRs have been given a great weight to be recognizable. Actually, each quantized MBR is only a cell of its quadtree region's 255×255 quantization grid, i.e. they index extremely small surface areas—which sum up to 1.15E-8 dsu^2 for the five quantized MBRs. As a reference, the MBR of the five quantized MBRs (red-bordered outer rectangle without fill color) accounts for 5.51E-3 dsu^2, i.e. the quantized MBRs' cumulated indexed surface area is almost 480,000 times smaller.

maries. Whereas QTMBR_3 (28.82 B) is also very close to the 29.18 B of the direct representation, the other techniques (especially QTMBR_1 and QTMBR_2) are capable of reducing the storage space requirements substantially, though. However, these observations show that in future work—similar to the resource descriptions in the distributed application scenario—the integration of the direct representation as an option in certain situations could be examined (in particular, when the leaf node capacity is low).

Again, it is difficult to compare the cumulated indexed surface areas of the internal nodes' levels (level 1 to level 3) due to the varying amount of internal nodes in the respective R-trees. Nevertheless, in general, there are no apparent surprises in the results (see Table 69): As always, no substantial improvements in the indexed surface areas per internal node level can be achieved over the MBR approach. Especially at level 2 and level 3, oftentimes significantly larger indexed surface areas occur for our summarization approaches. This is even though the MBR approach does not

Table 68: Overview of the average memory consumption of the leaf nodes' summaries in the summary-like R-trees in the R_5 scenario. These numbers do *not* take the byte- and word-alignment into account. The direct representation of the data points would account for 29.18 B on average.

technique →	MBR	MBRQT_1	MBRQT_2	MBRQT_3	MBRQT_4	MBRQT_5	MBRQT_6	QTMBR_1	QTMBR_2	QTMBR_3	QTMBR_4
avg mem. consumption [in B]	16.00	17.89	18.49	18.50	19.75	20.81	20.82	7.20	9.23	28.82	33.85

necessarily have the smallest amount of nodes per level (see the corresponding values for QTMBR_1 to QTMBR_3 in Table 70). As an example for non-substantial improvements, consider the MBR approach and QTMBR_4 which both have the same number of nodes (4) at level 1. Nevertheless, the indexed surface areas are almost identical, only minimally in favor of QTMBR_4 (68,214.4 dsu^2 instead of 68,590.1 dsu^2). Anew, this underlines that especially for data collections with regularly distributed data points, no improvements with regard to the *internal nodes*' cumulated indexed surface areas can be achieved over the MBR approach—at least without investing exorbitant amounts of storage space. For the $\text{QTMBR}^b_{c,a}$ parameterizations, again, it shows that due to the specifications of the summary-from-summaries calculation, a more detailed quantization is beneficial for internal nodes as QTMBR_2 has significantly better results than the other three parameterizations, in particular at level 3.

For the leaf nodes, the cumulated indexed surface areas are more easily comparable as the amount of leaf nodes is fairly similar between the techniques. Overall, most noteworthy is that in comparison with the T collection's scenarios, the summary-like R-trees of QTMBR_1 to QTMBR_3 now exhibit much less favorable results with regard to the cumulated indexed surface areas. As learned in the other scenarios' assessments, this key figure does not say much without considering the distribution of the indexed surface area per node. Nevertheless, since the corresponding assessement for the MBR-like R-trees in the R_5 scenario showed that for the R collection, no classical long tail distributions occur, it can be assumed that the listed values are an appropriate representation of the techniques' distributions. This means that we can presume that e.g. most of the QTMBR_4 R-tree's leaf nodes have smaller indexed surface areas than the leaf nodes of the MBR approach's R-trees. However, as just discussed in the structural analysis of the T_5 scenario's summary-like R-trees, the following has to be considered: For summary-like R-trees, the distribution of the indexed surface areas per node is much less meaningful than for MBR-like R-trees

as the assignment of data points to leaf nodes differs for the respective summary-like R-trees.
In Figure 112, the distributions of the leaf nodes' spatial scattering with regard to their stored data points are depicted for the usual techniques. As mentioned before, the amount of leaf nodes is virtualy identical for all techniques and therefore, the respective curves are well comparable. It shows that all curves exhibit very similar courses: The inferiority of the QTMBR$^{b}_{c,a}$ parameterizations is much less pronounced now. However, the curves of QTMBR_1 and QTMBR_4 are still slightly above those of the MBR approach and the MBRQT^{c,a} parameterizations. Hence, also for regular data point distributions, the possibly adverse effects of the SELECTBEST$(.)$-method's inappropriateness are greatest for QTMBR$^{b}_{c,a}$. Hereby, the QTMBR_4 curve is marginally above that of QTMBR_1. The other three parameterizations exhibit virtually identical curves and only in an extreme zoom level, it shows that with regard to the spatial scattering, it applies that MBRQT_1 $>$ MBRQT_6 $>$ MBR approach. However, it has to be noted that these differences are far from being significant.

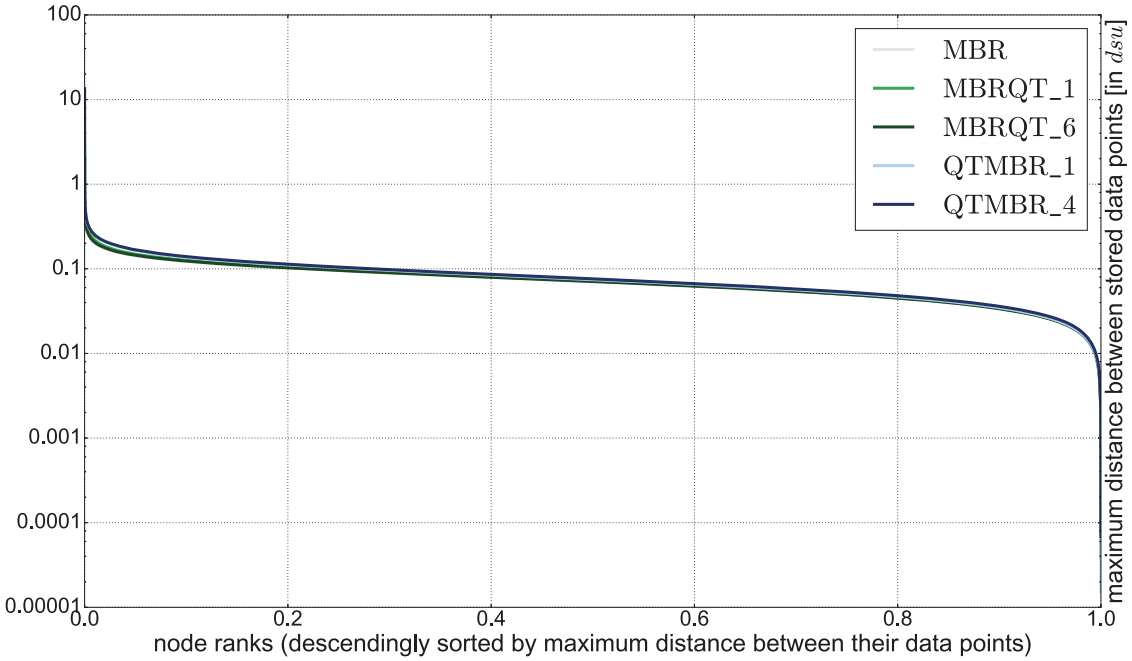


Fig. 112: Leaf nodes sorted in descending order by maximum distance between stored data points (for the R_5 scenario's summary-like R-trees). The node ranks are normalized. The x-axis is linear-scaled whereas the y-axis is log-scaled.

Table 69: Cumulated indexed surface areas of the summary-like R-trees in the R_5 scenario (all values in dsu^2).

technique	level 1	level 2	level 3	level 4
MBR	68,590.1	78,371.1	84,223.3	21,505.1
MBRQT_1	70,354.0	84,717.3	90,528.2	14,015.8
MBRQT_2	71,490.3	85,232.3	91,442.2	13,685.2
MBRQT_3	68,766.7	79,319.3	85,125.4	12,782.4
MBRQT_4	70,691.0	84,403.3	89,862.7	5,396.5
MBRQT_5	68,828.2	80,883.4	86,176.2	4,437.6
MBRQT_6	68,579.5	78,525.8	83,370.8	4,305.0
QTMBR_1	65,959.4	81,984.8	133,465.9	56,771.1
QTMBR_2	66,449.7	80,001.7	93,575.7	24,404.8
QTMBR_3	65,546.5	84,831.6	137,406.3	16,491.1
QTMBR_4	68,281.4	87,601.4	123,579.3	73.0

Table 70: Nodes per level of the summary-like R-trees in the R_5 scenario.

technique	level 1	level 2	level 3	level 4
MBR	4	488	58,460	6,856,031
MBRQT_1	8	865	77,647	6,852,902
MBRQT_2	9	868	77,718	6,853,139
MBRQT_3	10	886	77,691	6,852,612
MBRQT_4	10	878	77,640	6,847,677
MBRQT_5	9	870	77,701	6,848,297
MBRQT_6	11	864	77,574	6,846,736
QTMBR_1	2	223	39,061	6,921,365
QTMBR_2	2	341	58,603	6,859,530
QTMBR_3	2	368	65,414	6,856,805
QTMBR_4	4	613	107,191	6,849,230

Structural Analysis for the Summary-like R-trees in the R_25 scenario

For the summary-like R-trees in the R_25 scenario, different heights result—similar to the T_25 scenario. The heights are the same as there, i.e. 4 for the MBR approach and QTMBR_1 to QTMBR_3 while being 5 for the other techniques. The memory utilization rates of both internal and leaf nodes are nearly identical for all techniques (slightly above 69%). We see the slightly greater memory utilization rates here in compari-

son to the T_25 scenario as indication that in the latter scenario, also the summary-like MBR and MBRQT^{c,a} R-trees might be slightly affected by the non-randomness of our implementation for resolving ties in the SELECTBEST(.)-method—which is *only* relevant for the assignment of newly inserted data points located in regions of *very high point density*. Compared to the summary-like R-trees in the T_25 scenario, the values of the respective fanouts remain fairly stable. The only substantial change occurs for QTMBR_4 which now has an even lower fanout of 38.0. This is easily explainable by the greater spatial spread of the data points in the leaf nodes due to the regular data point distribution ($\rightarrow$ greater average leaf node summary size $\rightarrow$ lower fanout).

As usual, the amounts of indexed areas are significantly lower in the R_25 scenario compared to the corresponding R_5 scenario. Hence, for the higher MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ parameterizations, the numbers are not as extreme anymore, and not almost each data point is described by an area of its own. Nevertheless, the three MBRQT^{16,a} parameterizations (each with 11.9 million areas) and QTMBR_4 (15.2 million areas) still describe the leaf nodes in extreme detail—although the leaf nodes now store up to 25 data points, each. For comparison: The MBR approach indexes 1.45 million areas.

Similar to the corresponding *MBR-like* R-trees, also for the *summary-like* R-trees, the cumulated indexed surface areas at leaf node level are by far the greatest in the R_25 scenario. Here, in a comparison with these MBR-like R-trees, in most cases, slight increases occur for the summary-like R-trees. The increases are all between 1% and 6%. The exceptions are QTMBR_1 (-3.9%), QTMBR_3 (+15.0%), and QTMBR_4 (+30.8%). While the changes for QTMBR_1 as well as for the other techniques are in the range of being noise, the changes for QTMBR_3 and QTMBR_4 are noteworthy. These substantial increases—although overall, the same data point set is indexed and the average memory utilization rates in the leaf nodes are basically the same[289]—are a further indication that at least for QTMBR$^{b}_{c,a}$ summaries, the applied SELECTBEST(.)-method does not work as targeted as with MBR summaries. In this context, Figure 113 displays the distributions of the spatial scattering of the leaf nodes' data points for the usual techniques. Again, the curves are well comparable due to the virtually identical leaf node memory utilization rates. It shows that also this time, QTMBR_4 exhibits the greatest spatial scattering of all techniques.

[289] The leaf node memory utilization numbers are:

- MBR-like R-trees in the R_25 scenario:
  - QTMBR_3: 69.6%
  - QTMBR_4: 69.6%
- summary-like R-trees in the R_25 scenario:
  - QTMBR_3: 69.7%
  - QTMBR_4: 70.3%

The other techniques' curves all have nearly the same course, only the MBRQT_6 curve is slightly above those of the other techniques. Overall, the differences between the techniques are now a little more pronounced than in the R_5 scenario. Again, this implies that with increasing complexity of the leaf node summaries, the differences between the techniques become more evident. Once more, the techniques which should offer a greater spatial accuracy in theory are a bit worse with regard to the leaf nodes' spatial scattering.
In total, this means that the techniques' differences in the spatial scattering of the data points stored in the leaf nodes are very significant in the T_5 scenario and the T_25 scenario whereas in the R_5 scenario and the R_25 scenario, the adverse effects are mitigated. In the subsequent evaluation of the query results, we keep track of these observations and try to elaborate the reasons behind these phenomena.

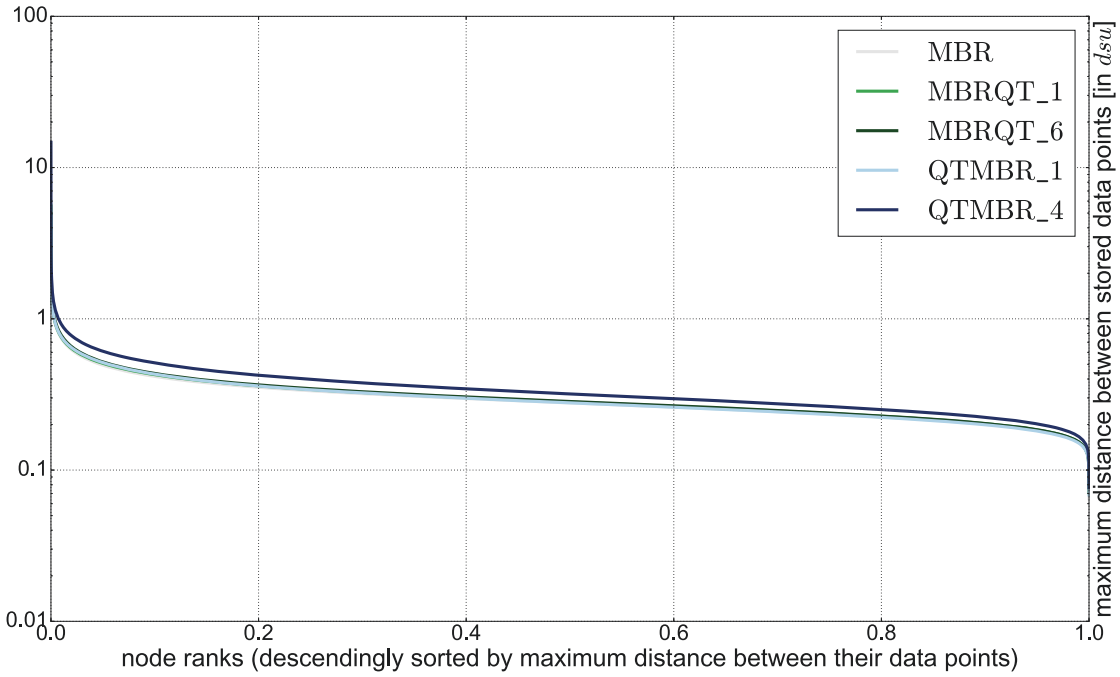


Fig. 113: Leaf nodes sorted in descending order by maximum distance between stored data points (for the R_25 scenario's summary-like R-trees). The node ranks are normalized. The x-axis is linear-scaled whereas the y-axis is log-scaled.

13.3. Assessment of the Query Results

In this section, the query results are evaluated. Hereby, we change the sequence of the assessment and proceed scenario-wise: At first, the T_5 scenario is investigated (section 13.3.1), followed by the T_25 scenario (section 13.3.2), the R_5 scenario (section 13.3.3), and the R_25 scenario (section 13.3.4). In general, the query result tables contain the number of point-to-rectangle distance calculations ('#Dist2Rect'-row), the number of distance calculations between a query point and a summary instance ('#Dist2Sum'-row), the total amount of page accesses ('#PA total'-row), the amount of page accesses to internal nodes ('#PA IN'-row), and the amount of page accesses to leaf nodes ('#PA LN'-row).

13.3.1. RESULTS FOR THE T_5 SCENARIO. In the following, we evaluate the query results for the T_5 scenario. Hereby, we start with assessing the results for the MBR-like R-trees. They serve to gauge the potential for improvement over the MBR approach. Afterwards, the summary-like R-trees are examined.

Query Results for the MBR-like R-trees in the T_5 scenario

Table 71 depicts the kNN query results, for both $k = 10$ and $k = 1,000$. At first, we focus on the results for $k = 10$. On average, the MBR approach requires 18.3 page accesses in total from which 12.5 are to internal nodes and 5.8 are to leaf nodes. The MBRQT^{c,a} parameterizations slightly improve the MBR approach with regard to page accesses to both internal as well as leaf nodes. Out of the MBRQT^{c,a} parameterizations, MBRQT_6 achieves the greatest reductions with 5.5% less total page accesses than the MBR approach. More specifically, the internal node accesses can be reduced by 3.9% while for the leaf node accesses, larger improvements of 8.7% are evident.

Table 71: kNN query results for the MBR-like R-trees in the T_5 scenario.

	MBR	MBRQT_1	MBRQT_2	MBRQT_3	MBRQT_4	MBRQT_5	MBRQT_6	QTMBR_1	QTMBR_2	QTMBR_3	QTMBR_4
k=10											
#Dist2Rect	1,401.5	1,496.0	1,687.3	1,975.8	3,363.3	4,104.6	4,478.9	13,378.4	1,990.3	12,364.0	11,786.7
#Dist2Sum	1,401.5	1,389.5	1,389.5	1,389.5	1,342.5	1,341.3	1,341.3	13,255.7	1,883.7	12,141.5	6,129.3
#PA total	18.3	18.2	18.1	18.0	17.8	17.6	17.3	656.4	43.5	122.9	57.6
#PA IN	12.51	12.40	12.40	12.40	12.02	12.02	12.02	113.74	16.66	104.22	52.76
#PA LN	5.83	5.77	5.68	5.57	5.74	5.57	5.32	542.63	26.87	18.66	4.82
k=1,000											
#Dist2Rect	2,247.4	2,356.3	2,674.5	3,328.9	4,323.8	5,264.1	6,084.2	15,612.3	3,038.0	14,589.1	15,762.8
#Dist2Sum	2,247.4	2,235.5	2,235.5	2,235.5	2,186.2	2,184.4	2,184.4	15,486.8	2,930.5	14,343.3	8,116.4
#PA total	321.5	321.2	320.8	320.0	320.8	320.1	318.9	1,456.6	389.2	457.0	368.6
#PA IN	19.70	19.60	19.60	19.60	19.20	19.18	19.18	132.89	25.56	123.11	69.66
#PA LN	301.74	301.63	301.20	300.40	301.57	300.93	299.73	1,323.70	363.65	333.89	298.90

For the internal nodes, the results for the three MBRQT^{8,a} parameterizations respectively the three MBRQT^{16,a} parameterizations are the same (after being rounded to two decimal places). This is no surprise since within each of these two groups, the results for the cumulated indexed surface areas are very similar (see Table 46 on page 324). Hence, parameter c is obviously the limiting factor here. For the leaf nodes, parameter a has more impact on the results. For example, MBRQT_4 (with $c = 16$ and $a = 0.001$) is slightly worse than MBRQT_2 (with $c = 8$ and $a = 1.0E{-}5$). In Table 72, the corresponding descriptive statistic values for the leaf node accesses are listed. Except the mean, the values are identical for both MBRQT_2 and MBRQT_4. This proves that the slight superiority of MBRQT_2 is not caused by single outlier values. The results are easily explainable, anyway: As shown in the structural analysis of the MBR-like R-trees in the T_5 scenario (section 13.2.1), the number of nodes with a *non-refined* basic MBR is greater for MBRQT_4 (roughly 7.6 million nodes) than for MBRQT_2 (roughly 6.9 million nodes). Thus, the basic MBRs of the leaf nodes are more often refined for MBRQT_2 than for MBRQT_4. This yields *slight* advantages with regard to the leaf node access performance, obviously. For example, for query 187 ($x = -2.67$, $y = 53.53$, located in Wigan between Liverpool and Manchester), MBRQT_2 requires only 5 leaf node accesses whereas MBRQT_4 requires 7 leaf node accesses to determine the top $k = 10$ data points. However, in a comparison of the performances for the 200 individual queries, the differences in the leaf node accesses are never greater than 2, i.e. they are not very pronounced. In fact, MBRQT_2 is better for 17 queries whereas MBRQT_4 is better for 8 queries. For the remaining 175 queries, the results are identical. Despite worse results for the leaf node accesses, the cumulated indexed surface areas at leaf node level are clearly in favor of MBRQT_4 (3,003.9 dsu^2 as opposed to 9,869.4 dsu^2, see Table 46). This underlines the importance of keeping the long tail distribution of the indexed surface area per leaf node in mind and proves that the cumulated indexed surface areas are unreliable indicators with regard to the query performance. Overall, MBRQT_6 is not only best of the MBRQT^{c,a} parameterizations but also of *all* tested techniques.

This is because the QTMBR$^{b}_{c,a}$ parameterizations are significantly worse than the MBR approach and the MBRQT^{c,a} parameterizations with regard to both internal node accesses as well as leaf node accesses. The only exception is QTMBR_4 at *leaf node level* where it is clearly the best performing technique, reducing the leaf node accesses by 9.4% relative to MBRQT_6 and by 17.4% in comparison to the MBR approach. Hereby, QTMBR_4 is slightly inferior with regard to the 'min'- and 'max'-values for the leaf node accesses (i.e. the outlier values, for which it offers 3 respectively 13 in contrast to 2 respectively 12, see Table 72) but superior for the majority of the queries: In a juxtaposition of the individual queries, it is better than MBRQT_6 for 79 queries. Only 8 times, it is worse. Overall, with regard to zero volume summaries, these results show that the frequent indexing

Table 72: Descriptive statistic values of the leaf node page accesses for the MBR-like R-trees in the T_5 scenario.

technique → #PA LN ↓	MBR	MBRQT_1	MBRQT_2	MBRQT_3	MBRQT_4	MBRQT_5	MBRQT_6	QTMBR_1	QTMBR_2	QTMBR_3	QTMBR_4
mean	5.83	5.77	5.68	5.57	5.74	5.57	5.32	542.63	26.87	18.66	4.82
min	2	2	2	2	2	2	2	4	3	3	3
25%-quant.	5	4	4	4	4	4	4	82.8	8	7	4
median	5	5	5	5	5	5	5	256.5	15	10	5
75%-quant.	7	7	7	6	7	6	6	656.5	28	19	5
max	12	12	12	12	12	12	12	4,455	348	329	13

of line-like rectangles for QTMBR_4 has only little (if any) adverse effects in comparison to the mostly point-like rectangles for the MBR approach and the MBRQT^{c,a} parameterizations. This is reasonable as these lines are usually *extremely short* intervals.

In the structural analysis, we suspected that with regard to leaf node accesses, QTMBR_1 to QTMBR_3 might face problems for queries located in regions of high data point density since their maximum spatial accuracy is bound to their comparatively coarse smallest indexable spatial unit. This assumption can be confirmed. Table 73 lists the techniques' leaf node accesses for three exemplarily selected queries:

— Query 171 ($x = 16.44$, $y = -21.90$), located in the metropolitan area 'Tyne And Wear' (including Newcastle and Sunderland), a region of high data point density,
— query 131 ($x = -95.36$, $y = 29.76$), located in Houston, Texas, a region of high data point density, and
— query 76 ($x = -1.44$, $y = 54.99$), located approximately 100 km northwest of Windhoek, Namibia, a region of low data point density.

The results of the three queries clearly show that QTMBR_1 to QTMBR_3 exhibit significantly greater amounts of leaf node accesses for query 171 and query 131 which are both located in regions of high data point density (also see Figure 92a on page 300).[290] In contrast, for query 76 located in the low-density region, they offer the same performance as the other techniques.

On basis of the structural analysis, the QTMBR$^{b}_{c,a}$ parameterizations' bad results for the internal node accesses are easily explainable: Table 46 on page 324 showed that at the internal nodes' levels in the MBR-like R-trees, the QTMBR$^{b}_{c,a}$ summaries achieve at best minor reductions for the *cumu-*

[290] This is even though QTMBR_2 seems to profit from its greater quantization accuracy for query 131 as it only requires 32 leaf node accesses there, which is comparatively few.

Table 73: Leaf node access performances for examplarily selected 10NN queries (of the MBR-like R-trees in the T_5 scenario).

technique → query ↓	MBR	MBRQT_1	MBRQT_2	MBRQT_3	MBRQT_4	MBRQT_5	MBRQT_6	QTMBR_1	QTMBR_2	QTMBR_3	QTMBR_4
query 171	7	7	7	7	7	7	7	785	348	329	6
query 131	7	7	7	5	7	7	5	1,663	32	291	5
query 76	4	4	4	4	4	4	4	4	4	4	4

lated indexed surface areas.[291] Due to the restrictions imposed by target depth td, the internal nodes' $\text{QTMBR}^b_{c,a}$ summaries oftentimes only *approximate* the full-precision MBRs (especially at level 3). In such cases, the $\text{QTMBR}^b_{c,a}$ summaries obviously *must* be coarser than the corresponding MBR summaries. Since in general, long tail distributions arise for the indexed surface areas, it is conceivable that this is also the case for the internal nodes. Overall, this implies that for the $\text{QTMBR}^b_{c,a}$ parameterizations, the minor reductions of the *cumulated* indexed surface areas are achieved by improving rather few internal nodes with very large indexed surface areas. For the majority of the internal nodes, especially those close to the leaf node level, we suspect that the $\text{QTMBR}^b_{c,a}$ summaries index a greater surface area as their summaries can actually be no more than *approximations* of the corresponding full-precision MBRs. The fact that QTMBR_2 is the best $\text{QTMBR}^b_{c,a}$ parameterization with regard to internal node accesses by far reaffirms this assumption.[292] On basis of our analyses so far, it is obvious that the $\text{QTMBR}^b_{c,a}$ parameterizations' problems with the internal node accesses should primarily arise for the nodes at level 3 (i.e. the level immediately above leaf node level). In Table 74, the average numbers of page accesses to internal nodes are displayed per level. Indeed, the serious problems of the $\text{QTMBR}^b_{c,a}$ parameterizations become evident at level 3.
As a final proof of our considerations' correctness, Figure 114 depicts the level-3-nodes sorted in descending order by their indexed surface areas. The showcased techniques are the MBR approach, QTMBR_1, QTMBR_2, and QTMBR_4. It shows that in fact, a long-tail distribution is prevalent for the indexed surface areas of the level-3-nodes. Beyond roughly rank 1,200, the MBR approach has the smallest indexed surface areas of the depicted techniques. It is also the only listed technique for which zero volume surface areas arise—for 200 nodes. From these nodes' MBRs, 183 are actually points and 17 are actually iso-oriented lines. Hence, for extremely dense regions, there are even level-3-nodes with zero volume surfaces. Overall,

[291] At level 3, QTMBR_1 and QTMBR_3 are actually even worse.

[292] A greater target depth td for the initial quadtrees also helps: compare the results of QTMBR_1 and QTMBR_3 ($td = 2$) to those of QTMBR_4 ($td = 3$). However, a detailed quantization is much more valuable.

Table 74: Average amount of internal node accesses for the MBR-like R-trees in the T_5 scenario. The internal node accesses are displayed level-wise.

	MBR	MBRQT_1	MBRQT_2	MBRQT_3	MBRQT_4	MBRQT_5	MBRQT_6	QTMBR_1	QTMBR_2	QTMBR_3	QTMBR_4
level 0	1	1	1	1	1	1	1	1	1	1	1
level 1	2.01	2.01	2.01	2.01	2.01	2.01	2.01	1.88	1.85	1.88	1.80
level 2	4.62	4.62	4.62	4.62	4.49	4.49	4.49	6.42	4.41	6.37	5.39
level 3	4.88	4.78	4.78	4.78	4.53	4.52	4.52	104.45	9.40	94.97	44.58

it is clear now why the $\text{QTMBR}^{b}_{c,a}$ summaries cannot compete anymore at level 3. Furthermore, Figure 114 also shows that QTMBR_2 has smaller indexed surface areas than QTMBR_4 for all ranks. For QTMBR_4, the minimally occuring value of a level-3-nodes' indexed surface area is 0.01557 dsu^2 while for QTMBR_2, it is 0.00024 dsu^2. Overall, this proves that our stipulations—'the $\text{QTMBR}^{b}_{c,a}$ summaries only approximate the MBR summaries in the majority of cases' and 'the smallest indexable spatial unit of QTMBR_2 is smaller than that of QTMBR_4'—are correct.

Again, it shows that the cumulated indexed surface area is only partially suitable as key performance indicator since also for the internal nodes, a long tail distribution of the indexed surface area per node exists[293] and therefore, the resulting values are dominated by the 'short head'. In general, at lower levels of the R-tree, it is much more important how accurate the majority of the summaries are—especially in regions of high point density where also most of the queries are located.

This is even more true for the leaf nodes. For example, QTMBR_3 reduces the leaf nodes' cumulated indexed surface area by 97.4% in comparison to the MBR approach. Nevertheless, 220% *more* leaf node accesses are made for QTMBR_3. Similarly, the $\text{MBRQT}^{c,a}$ parameterizations offer only comparatively small improvements versus the MBR approach with regard to the leaf node accesses—despite solid reductions of the cumulated indexed surface areas at leaf node level. On the one hand, this is attributable to the long tail distribution of the indexed surface areas which results in that for the majority of the leaf nodes, the MBR summaries and the $\text{MBRQT}^{c,a}$ summaries are the same. On the other hand, it is also because the MBR approach is already very good with regard to the leaf node accesses, leaving merely room for improvement: 'Mathematically' determined, on average,

[293] At least, this is the case for the T collection, i.e. in the T_5 and T_25 scenarios.

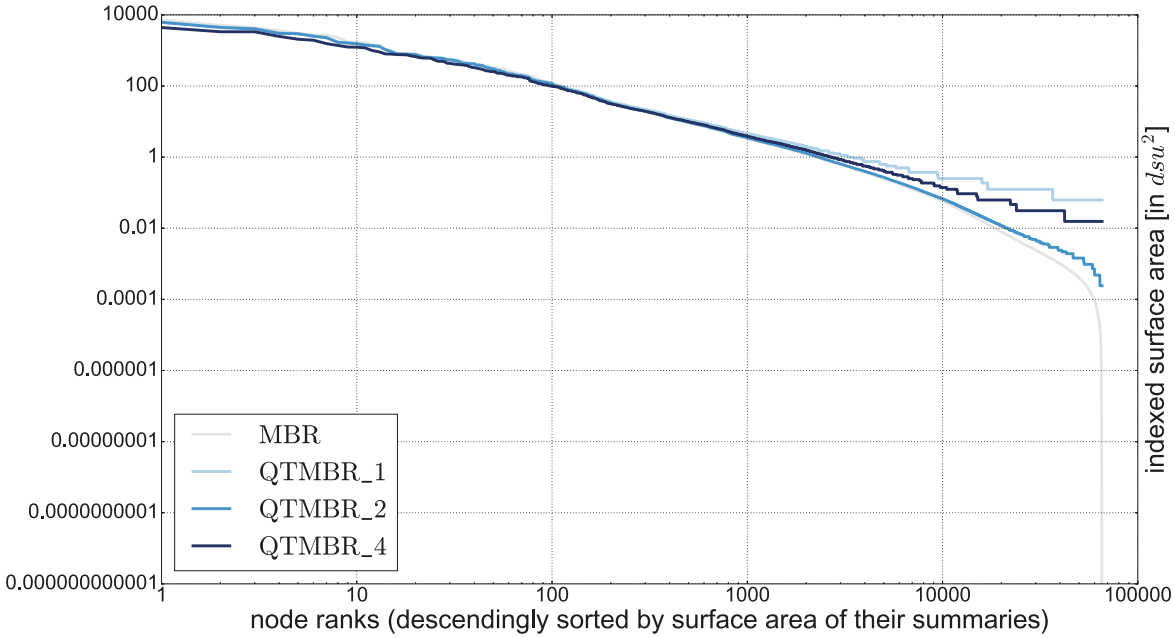


Fig. 114: Internal nodes at level 3 of selected MBR-like R-trees in the T_5 scenario, sorted in descending order by their indexed surface areas. Both axes are log-scaled.

only about 20.38 data points[294] are examined if they are one of the $k = 10$ closest neighbors to the given query point during the *entire* query processing. Hence, the achievable improvements are by no means comparable to the distributed application scenario. Therefore, the leaf node access results of the MBRQT^{c,a} parameterizations and in particular QTMBR_4 are actually fairly solid.

The average number of distance calculations to summary instances[295] ('#Dist2Sum'-row in Table 71) obviously correlates with the number of page accesses as the structures of all MBR-like R-trees are identical. Thus, we do not further consider these values for the analyses of the MBR-like R-trees' query results. The average number of point-to-rectangle distance calculations ('#Dist2Rect'-row in Table 71) is dependent on two aspects:[296]

— The amount of page accesses.
— The amount of indexed areas per summary.

[294] 'Mathematically' means that the given value is not determined empirically but by means of a calculation. The 'mathematically determined' value arises from the average amount of leaf node accesses (#PA), the leaf node capacity (CAP), and the average memory utilization of the leaf nodes (MU) as $\#PA \cdot (CAP \cdot MU)$, or concretely, as $5.83 \cdot (5 \cdot 0.699) = 20.38$.

[295] Remember that a summary might consist of several rectangular areas for both the MBRQT^{c,a} as well as the QTMBR$^{b}_{c,a}$ approach.

[296] For *summary-like* R-trees, a third aspect to consider is the average fanout of the internal nodes. Nevertheless, the fanout does not play a role for the currently assessed MBR-like R-trees because of their identical structures.

Overall, for $k = 10$, the MBR approach offers the lowest average number of distance calculations. This has also been our a priori assumption as we considered a greater number of distance calculations to be the price to pay for being able to reduce the number of page accesses (see section 10.5). Hence, this assumption is confirmed by the results. In general, the average number of distance calculations varies widely for the other techniques, reaching from 1,496.0 (MBRQT_1, +6.7% compared to the MBR approach) up to 13,378.4 (QTMBR_1, +855%). Despite the huge differences in the numbers, the introduced runtime overhead is negligible: For 100,000 point-to-rectangle distance calculations, we measured a runtime of 3.99 ms.[297] As can be seen in Table 71, for all techniques, much less than 100,000 distance calculations are conducted. Hence, for the overall runtime[298], the distance calculations are a non-factor in comparison to the page accesses: For *each* page access, 9 to 15 ms are required. As depicted in Table 71, even when $k = 10$, the average number of page accesses is at least 17.3. Consequently, in the further course of the evaluation, our focus is on the page accesses as the overhead on distance calculations is non-critical with regard to the overall runtime. For completeness, the '#Dist2Rect'- and the '#Dist2Sum'-rows are still included in the respective query result tables, though.
For $k = 10$, the total number of page accesses is dominated by the number of accesses to internal nodes: for e.g. the MBR approach, the ratio between internal node accesses and leaf node accesses is about 2:1. Hence, the prerequisites for our summarization approaches may have been not the best from the start as the internal node accesses have been suspected to be more difficult to improve in the structural analysis. The ratio between the accesses to internal nodes and leaf nodes turns completely for $k = 1,000$: For the MBR approach, it is now 1:15.3. Thus, is this finally the time for our summarization approaches to shine? The answer is disappointing as the results are very similar to those for $k = 10$ (see Table 71). It still applies that for the *absolute* amounts of page accesses, the MBRQT^{c,a} parameterizations only minimally improve the MBR approach for both internal and leaf node accesses. Therefore, the *relative* page access reductions are even significantly lower now. For example, MBRQT_6 achieves an overall reduction of 0.81% (which has been 5.5% for $k = 10$) that can be split into

[297] The measurement was conducted as follows: We loaded the MBR-like MBR R-tree into memory and extracted the MBRs from each 50,000 leaf node and internal node summaries, constituting the 100,000 rectangles for the runtime tests. As a data point for the point-to-rectangle distance calculations, we selected the first query point of the T collection. Then, the time measurements started: We conducted ten runs of distance calculations between the data point and the 100,000 rectangles. Finally, we took the average of the resulting runtimes. This led to the stated average runtime of 3.99 ms per 100,000 point-to-rectangle distance calculations.

[298] Technically, also the deserialization overhead (which was identified to be the bottleneck of our implementation in section 13.1) has to be taken into account for the overall runtime of a query. Nevertheless, the development of a solution to this issue and a subsequent overall evaluation are part of future work.

2.6% (3.9%) for the internal nodes and 0.67% (8.7%) for the leaf nodes, i.e. its *relative* advantageousness especially with regard to the latter strongly diminished. This is no surprise: The mathematically determined number of examined data points to retrieve the $k = 1,000$ nearest neighbors is 1,054.6 for the MBR approach, i.e. the share of non-relevant data points which are considered until the query result is ascertained is only 5.5%. For such numbers, it is illusionary to expect relevant improvements in the *relative* results. Consequently, even the utilization of QTMBR_4 leads to considering an *absolute* amount of merely 10 data points less than when applying the MBR approach. In general, the *relative* differences between the techniques diminish for $k = 1,000$: The QTMBR$^b_{c,a}$ parameterizations are no longer as inferior as for $k = 10$ whereas the MBR approach catches up to the better techniques with regard to the leaf node accesses. Overall, when comparing the techniques with each other, the single noteworthy change in the absolute results is that QTMBR_4 is now slightly better than QTMBR_2 in terms of the total amount of page accesses. This is due to the greater significance of the leaf node accesses for the total amount.

Table 75: Range query results for the MBR-like R-trees in the T_5 scenario.

	MBR	MBRQT_1	MBRQT_2	MBRQT_3	MBRQT_4	MBRQT_5	MBRQT_6	QTMBR_1	QTMBR_2	QTMBR_3	QTMBR_4
r=10	result size = 21,533.0										
#Dist2Rect	9,338.6	9,462.2	10,238.9	14,001.7	11,636.4	13,295.1	17,890.7	25,564.7	10,539.2	24,541.2	34,590.8
#Dist2Sum	9,338.6	9,325.0	9,324.6	9,324.6	9,263.3	9,250.0	9,250.0	25,436.6	10,429.2	24,236.7	17,752.9
#PA total	6,268.7	6,268.4	6,267.2	6,265.4	6,267.8	6,265.8	6,262.9	8,521.7	6,397.1	6,459.4	6,334.1
#PA IN	80.2	80.1	80.1	80.1	79.6	79.4	79.4	217.9	89.4	207.7	152.3
#PA LN	6,188.5	6,188.4	6,187.1	6,185.4	6,188.3	6,186.3	6,183.5	8,303.8	6,307.7	6,251.7	6,181.8
r=1,000	result size = 256,137.0										
#Dist2Rect	79,009.1	79,489.5	87,214.2	120,032.6	83,635.0	94,674.3	134,756.7	91,883.4	79,886.6	91,867.6	168,496.6
#Dist2Sum	79,009.1	78,990.1	78,990.1	78,990.1	78,867.6	78,860.8	78,860.1	91,717.7	79,745.5	90,883.0	84,973.0
#PA total	73,924.7	73,923.6	73,920.6	73,919.2	73,922.0	73,917.6	73,915.4	75,598.6	74,029.8	74,074.1	73,965.5
#PA IN	676.8	676.7	676.7	676.7	675.7	675.6	675.6	785.7	683.1	778.6	727.9
#PA LN	73,247.9	73,246.9	73,244.0	73,242.5	73,246.3	73,242.0	73,239.8	74,812.9	73,346.6	73,295.5	73,237.6

Table 75 outlines the results for the range queries. Due to the erratic data point distribution of the T collection, the average sizes of the result sets are much greater than the target radii assuming a regular spatial distribution forecasted: For the radius targeting at 10 data points ($r = 10$), the average result set size is 21,533.0. Hereby, the greatest result set's size is 211,695 whereas the size of the smallest is 1. In total, only three queries have less than 10 data points in their result set. For the radius targeting at 1,000 data points ($r = 1,000$), the average result set size is 256,137.0 with the greatest respectively smallest set having a size of 1,111,566 respectively 4. Moreover, only for two queries, less than 1,000 data points are retrieved. Once again, these numbers show that for the T collection, most query points are located in regions of the data space where enormous amounts of data points are concentrated.

The large result set sizes lead to that the total amount of page accesses is *extremely* dominated by the leaf node accesses. However, the results are basically the same as for the kNN queries: The MBR approach is minimally improved by the MBRQT^{c,a} parameterizations whereas the QTMBR$^{b}_{c,a}$ parameterizations are inferior, overall. As always, QTMBR_4 is best with regard to the leaf node accesses. Nevertheless, this fact cannot compensate its inherent shortcomings concerning the internal nodes' accesses. Again, it is evident that there is not much room for improvement over the MBR approach with regard to the leaf node accesses. Mathematically, for $r = 10$, an optimal search would require $21,533.0/(5 \cdot 0.699) = 6,161.1$ leaf node accesses on average. For the MBR approach, 6,188.5 leaf node accesses are conducted on average, i.e. mathematically, it is only 0.44% away from an optimal selection.

The range queries' numbers of point-to-rectangle distance calculations provide proof that the greater the result sets, the lower the relative computational overhead over the MBR approach for our summarization approaches. For example, for $k = 10$, the overhead for QTMBR_4 is $(11,768.7/1,401.5) - 100\% = 740\%$. For $k = 1,000$, its overhead decreases to 601% while for $r = 10$ (result set size of 21,533) and $r = 1,000$ (result set size of 256,137), the numbers are 270% and 113%. Hence, also for larger result sets, the computational costs of the point-to-rectangle distance calculations have only negligible relevance for the overall runtimes: For instance, the 168,496.6 distance calculations for $r = 1,000$ and QTMBR_4 would account for about 6.72 ms (if the average 3.99 ms per 100,000 point-to-rectangle distance calculations are used as a benchmark). As a comparison: Assuming 9 ms per page access, the 73,294.7 page accesses for $r = 1,000$ and the MBR approach accumulate to 659.7 seconds.

Overall, it can be stated that the results show that the partly enormous reductions in the cumulated indexed surface areas do not really pay off in the page access performances. Since all the query results of the T_5 scenario's MBR-like R-trees have been assessed, at this point, we revisit two assumptions made in the corresponding structural analysis.

One presumption was that it seems more likely to reduce leaf node accesses than internal node accesses. Obviously, this is a rather fuzzy statement. Moreover, it can only be meaningfully investigated for the MBRQT^{c,a} parameterizations because for the QTMBR$^{b}_{c,a}$ parameterizations, diverse problems with regard to both internal node accesses as well as leaf node accesses became apparent. In Table 76, the relative internal node and leaf node access performances of the MBR approach (which is the benchmark) and the MBRQT^{c,a} parameterizations are depicted.

In principle, our assumption is only true for $k = 10$: There, the MBRQT^{c,a} parameterizations' relative improvements for the leaf node accesses are greater than for the internal node accesses.[299] For example, MBRQT_3 re-

[299] The only exception for $k = 10$ is MBRQT_4.

Table 76: Comparison of the relative internal node (IN) and leaf node (LN) access performances for the MBR-like R-trees in the T_5 scenario. The MBR approach is the benchmark and thus, its amount of internal node respectively leaf node accesses corresponds to 100% for each row. In case the leaf node accesses are improved to a greater extent than the corresponding internal node accesses, the corresponding cells are highlighted in dark-grey.

	MBR	MBRQT_1	MBRQT_2	MBRQT_3	MBRQT_4	MBRQT_5	MBRQT_6
k=10							
IN accesses	12.51	99.2%	99.2%	99.2%	96.2%	96.1%	96.1%
LN accesses	5.83	99.0%	97.4%	95.6%	98.4%	95.6%	91.3%
k=1,000							
IN accesses	19.70	99.5%	99.5%	99.5%	97.4%	97.4%	97.4%
LN accesses	301.74	100.0%	99.8%	99.6%	99.9%	99.7%	99.3%
r=10							
IN accesses	80.2	99.9%	99.9%	99.9%	99.2%	99.1%	99.1%
LN accesses	6,188.5	100.0%	100.0%	99.9%	100.0%	100.0%	99.9%
r=1,000							
IN accesses	676.8	100.0%	100.0%	100.0%	99.7%	99.7%	99.7%
LN accesses	73,247.9	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%

quires 99.2% of the MBR approach's number of internal node accesses but only 95.6% of its number of leaf node accesses. With respect to larger result set sizes (i.e. $k = 1,000$, $r = 10$, and $r = 1,000$), the MBRQT^{c,a} parameterizations' relative improvements over the MBR approach diminish *in general*. This is easily comprehensible because also the MBR approach gets closer and closer to the optimal result with regard to the *relative* numbers. Furthermore, for the queries with larger result set sizes, the relative improvements are now actually greater for the internal node accesses. However, it is debatable whether measuring relative improvements is a suitable means of comparison at all in the given context. This is because the absolute numbers of internal node accesses and the leaf node accesses always differ widely. Consequently, we only want to state that the improvements with regard to the internal node and leaf node accesses are equally marginal—which was not necessarily to be expected on basis of the structural analysis.

As a second assumption to revisit, we anticipated that for MBRQT_1, only small improvements over the MBR approach are expectable for the *leaf node accesses* as in its R-tree, only 130,182 leaf nodes feature a smaller indexed surface area than in the MBR R-tree. Overall, the query results confirm this stipulation (1.0% improvement for $k = 10$, and less than 0.1%

for $k = 1,000$, $r = 10$, and $r = 1,000$). In this context, we further assumed that the results might be much more favorable for MBRQT_6 because it not only reduces the cumulated indexed surface area at leaf node level by 84% but also, 4,556,761 or 59.5% of the leaf nodes feature a smaller indexed surface area. Nevertheless, the improvements achieved for the leaf node accesses are only 8.7% ($k = 10$), 0.67% ($k = 1,000$), respectively less than 0.1% ($r = 10$ and $r = 1,000$). As noted before, the greater the result set sizes, the harder it is to achieve improvements in the relative results. However, even for $k = 10$, the improvements for MBRQT_6 are way smaller than one could have hoped for after the structural analysis. Therefore, once again, we want to analyze the reasons why the leaf node access performance of the MBR approach is only slightly improvable (despite apparently good prerequisites in some cases such as for MBRQT_6)—but this time from a *theoretical point of view*.

For once, as already showcased, it is because the results for the MBR approach are already so good that there is almost no room for further improvements. Obviously, the applied SELECTBEST(.)-method and the R*-split are capable of achieving such an assignment of data points to leaf nodes that the MBRs of the leaf nodes are largely free from overlap. A measurement shows that for the T_5 scenario's basic R-tree, the total mutual overlap between the leaf nodes' MBR summaries is 1,808.2 dsu^2. Given a cumulated indexed surface area of 18,751.4 dsu^2 and an amount of 7.65 million leaf nodes, this is a fairly respectable result (the evaluation of the distributed application scenario has shown to which immense amounts the overlaps of MBR summaries can accumulate). When such a state is achieved, MBRs are a sufficiently accurate description of a leaf node's spatial footprint. It does not matter much anymore that our summarization approaches might be able to greatly reduce the indexed surface areas in comparison to the MBR approach.

See Figure 115 for an illustrating example with a given query point and four leaf nodes in the vicinity. These nodes are described by MBRQT^{c,a} summaries whose basic MBRs are completely free of mutual overlap. Overall, for these four nodes, the MBRQT^{c,a} summaries reduce the indexed surface areas by 13/16 in comparison to the basic MBRs. Now, assume an kNN query with $k = 2$ is conducted. In the query processing, at first, *node 3* is accessed because its summary has the lowest MINDIST to the query point. After the data points of *node 3* have been examined, the query radius is set to the distance of the currently second nearest neighbor (blue circle, upper right quadrant of the figure). Next, *node 2* is accessed and the new second nearest neighbor is the data point stored in *node 2* which is closer to the query point than the previous second nearest neighbor. The query radius is updated accordingly. Then, *node 1* has to be investigated since the query ball slightly intersects one of *node 1*'s indexed areas. However, no closer data points than the current two nearest neighbors are found in *node 1*. Since the query ball does not intersect any unconsidered nodes anymore,

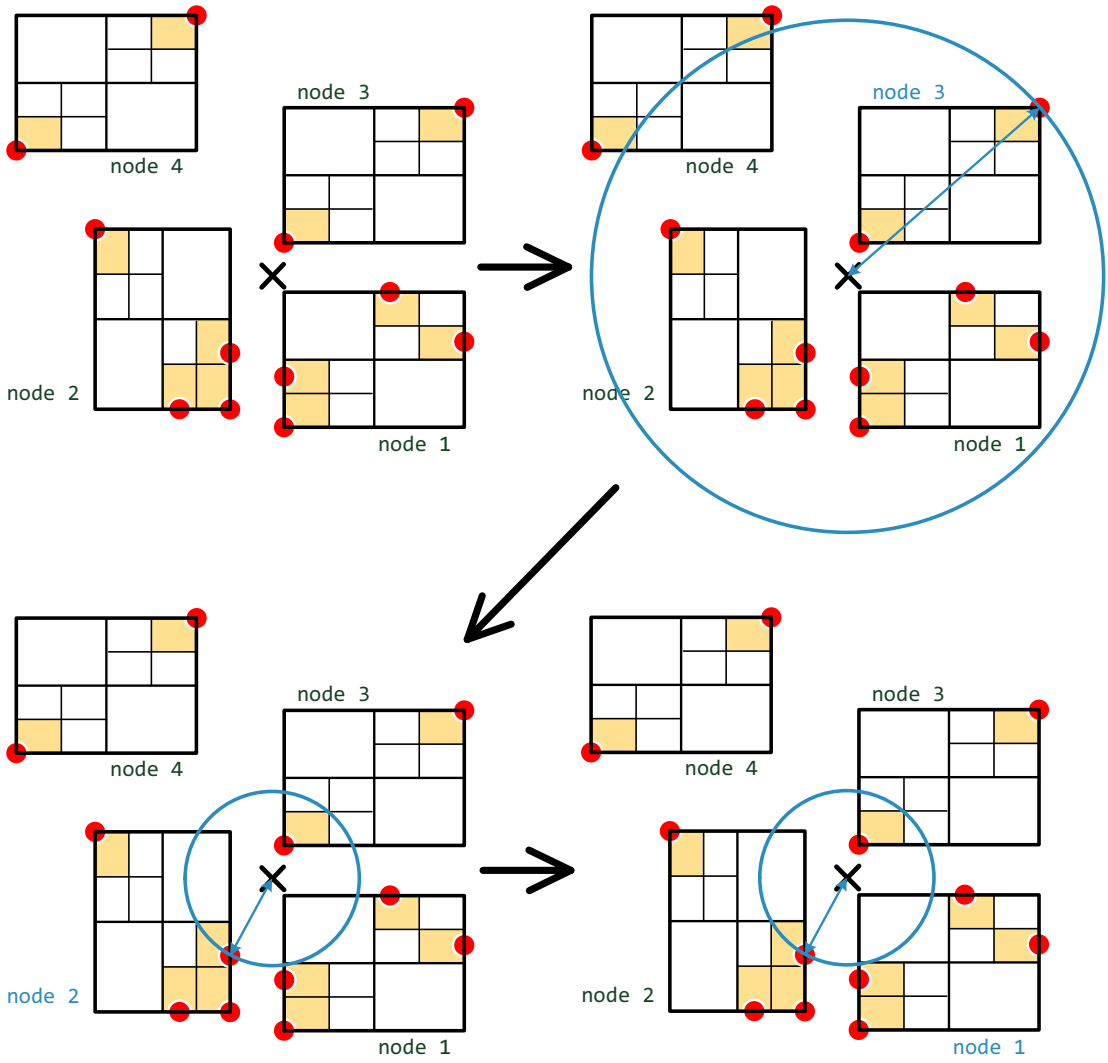


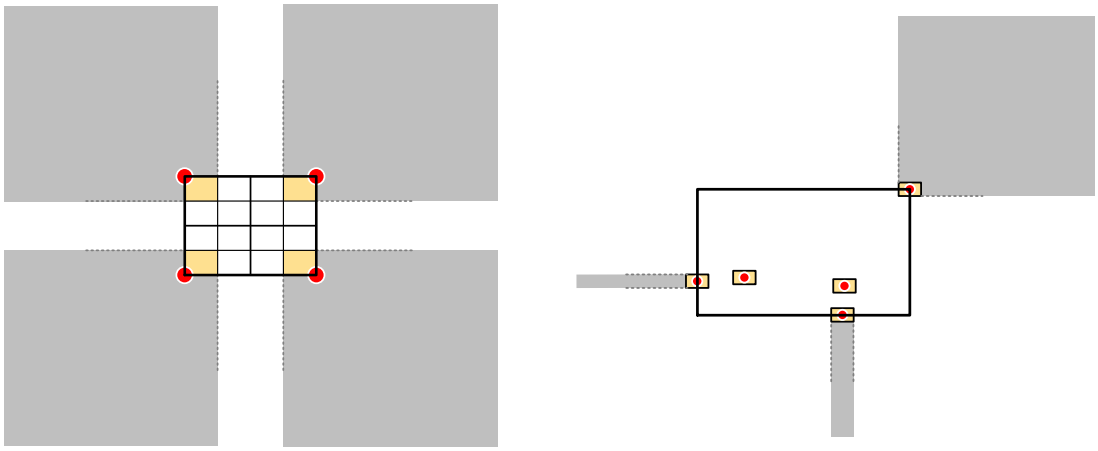
Fig. 115: Exemplary query processing involving four leaf nodes described by MBRQT^{c,a} summaries. The black cross depicts the location of the query point.

the query terminates and the $k = 2$ nearest neighbors have been found. If the nodes would have been described with MBR summaries, the access order would have been different (*node 1* $\rightarrow$ *node 3* $\rightarrow$ *node 2* instead of *node 3* $\rightarrow$ *node 2* $\rightarrow$ *node 1*) but the total number of leaf node accesses would have been the same (3), still. Of course, this is a constructed example. However, it illustrates how the importance of the indexed surface areas diminishes when the nodes are free from overlap. As has been shown in the evaluation of the distributed application scenario, the mutual overlap between *resources* would be a better key figure than the indexed surface area[300] to assess the expectable query performances.[301] The absence of overlap en-

[300] Captured in form of the *data space coverage* in the distributed application scenario.

[301] Note though that the mutual overlap is *extremely* expensive in its calculation. Therefore, we did not determine the overlaps for each technique in the centralized application scenario.

sures that (despite the comparative coarseness of the MBR summaries) the ordering of the *resources* in the query processor's priority queue works very well. As illustrated in the pathological example of Figure 115, the access order may not always be optimal for the MBR approach due to its deficiencies in spatial accuracy. Nevertheless, there will be many cases in which the final set of accessed *resources* is exactly the same for the different techniques, despite possibly varying access orders.



(a) $\text{MBRQT}^{c,a}$ (b) $\text{QTMBR}^{b}_{c,a}$

Fig. 116: Depiction of the 'outward appearances' of exemplary summaries.

Furthermore, the 'outward appearance' of the summaries has to be considered. In the query processing, *only* the MINDIST is calculated between the query point and a summary. Hence, the MINDIST is the only thing known to the query processor. Metaphorically, the only thing a query processor 'sees' of an entire summary is its closest point to the query point—which is the 'outward appearance' of the summary to the query processor. This outward appearance is heavily dependent on the relative positions of query point and summary. Figure 116a exemplifies this for an $\text{MBRQT}^{c,a}$ summary which achieves a surface area reduction of 75% as opposed to the corresponding MBR. Nevertheless, for *any point* located in the grey-colored regions, the MINDIST to the $\text{MBRQT}^{c,a}$ summary is the same as to the MBR. Hence, if the query point is located in any of these regions, the outward appearance of both summaries is the same to the query processor. Only in the white regions, the $\text{MBRQT}^{c,a}$ summary makes a difference. Since typically, the MBRs of leaf nodes are spatially very narrow, the favorable white regions make up only very small portions of the entire data space. Again, it is a constructed example but it shows that an $\text{MBRQT}^{c,a}$ summary's 'outward appearance' in comparison to the corresponding MBR summary might not be as favorable as the pure indexed surface area numbers possibly suggest. This is especially the case for $\text{MBRQT}^{c,a}$ summaries as at least two opposing quadrants of the basic MBR are occupied in case a refinement is applied. The $\text{QTMBR}^{b}_{c,a}$ summaries might, overall, fare com-

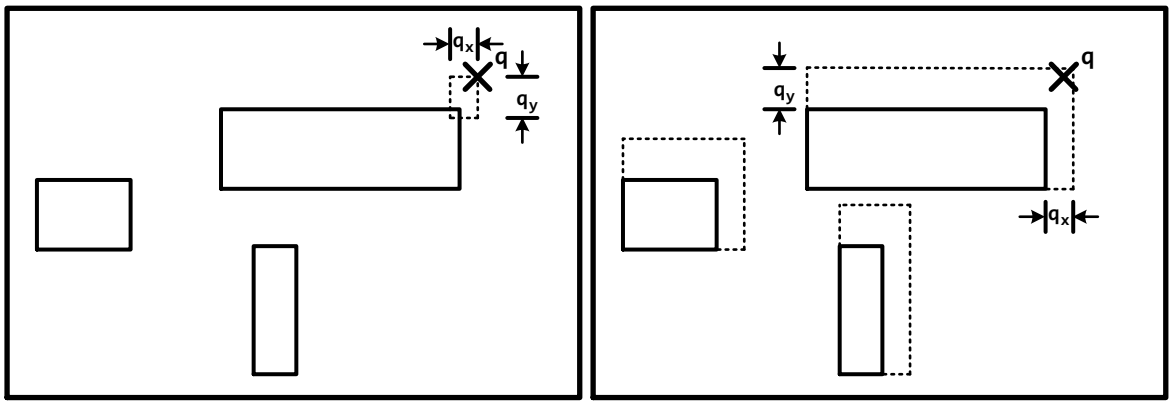
paratively better in this regard. Nevertheless, also their outward appearance is not as favorable as the pure numbers suggest. In Figure 116b, a situation comparable to the one pictured in Figure 111 on page 364 is depicted (in which a QTMBR_4 summary indexed a 480,000 times smaller surface area than the corresponding MBR summary). Although the grey regions at the left and bottom are extremely narrow, the grey region at the upper right, in any case, makes up substantial portions of the entire data space. For any query point located in these regions, the outward appearance of the $\text{QTMBR}^b_{c,a}$ summary is not better than the one of the corresponding MBR summary—even though the indexed surface area might be 480,000 times smaller in total.[302]

Related to this, at various points of the work, we stipulated that a node's probability of being accessed corresponds to its indexed surface area. This is true for point queries. However, for queries involving query radii (such as range or kNN queries), this changes as additionally, the radius of the given query has to be taken into account for determining these probabilities. These reflections are based on a model by Kamel and Faloutsos to estimate disc access performances when conducting window queries in R-trees [Kamel and Faloutsos 1993, p. 494ff]. They argue that a window query wq (of an origin q, an interval q_x in x-dimension, and an interval q_y in y-dimension) can be reduced to a point query if the MBRs of the R-tree's nodes are 'inflated' by q_x in the x-dimension and q_y in the y-dimension. Figure 117 illustrates the inflation process: In Figure 117a, the three example nodes' MBRs and the original window query wq (specified by q, q_x, and q_y) are depicted. In Figure 117b, the inflated MBRs and q (which now corresponds to the location of the point query) are showcased. Since wq has been reduced to a point query, the stipulation 'surface area equals access probability' is applicable, again. Due to the inflation, the MBRs' surface areas are significantly increased and therefore, their probabilities of being accessed are much higher than their actually indexed surface areas.[303] Assuming very small indexed areas and a comparatively large window query, it is also obvious that the overwhelming part of an inflated MBR's surface area is introduced by the inflation, i.e. for its access probability, its actually indexed surface area is negligible.

The inflation of MBRs (or more generally rectangles) is also transferable to query types operating with a query radius q_{rad} (such as range queries and kNN queries). For these, the inflation can be conducted as follows: *On*

[302] In fact, due to the quantization, there is even a high probability that in the grey regions, the outward appearance of the $\text{QTMBR}^b_{c,a}$ summary is slightly worse than the MBR summary's outward appearance.

[303] The model of Kamel and Faloutsos includes an equation for calculating the expectable disk access performance of an R-tree. This equation lays a theoretical foundation for the importance of minimizing the perimeters of the nodes' MBRs. The larger the query, the more important the perimeter becomes [Kamel and Faloutsos 1993, p. 495]. These theoretical foundations are also intuitively comprehensible by considering the depictions in Figure 117.



(a) before inflation (b) after inflation

Fig. 117: Illustration of transforming a window query to a point query by inflating the nodes' MBRs. The illustration is based on Figure 8 in [Kamel and Faloutsos 1993].

the shortest line between a rectangle r and the query point q, the point p has to be determined which is in distance q_{rad} to r. This point p has to be part of the face of r's closest side to q after inflation.[304] Hence, r has to be enlarged or 'inflated' accordingly. Figure 118 illustrates this inflation for one example rectangle and two example queries. Overall, these considerations attest that a node's probability of being accessed is not only dependent on its actually indexed surface area but also on the query radius q_{rad}. Consequently, the improvements of our summarization approaches with regard to the indexed surface areas are additionally diminished. Arguably, our summarization approaches should be able to reduce the kNN query radius faster compared to the MBR approach in situations as depicted in Figure 115. However, there is still one aspect to consider which we already discussed before:

The erratic data point distribution of the T collection also comes favorably into play for the MBR approach. In densely populated regions, the up to five data points of a *leaf node* are often so narrowly packed that for our MBRQT^{c,a} parameterizations, the basic MBRs are not refined anymore (i.e. their summaries correspond to a plain MBR) while QTMBR_1 to QTMBR_3 suffer from that their summaries are bound to their smallest indexable spatial units. With regard to the *internal nodes*, the nodes' subtrees administer such a number of data points that the associated data space is so densely populated with data points that only very rarely, regions of dead space can be excluded by our summarization approaches. For MBRQT^{c,a}, this generally leads to solely minimally improved performances

[304] Note that therefore, a rectangle's inflation for radii-based queries is orientation-sensitive, i.e. it is dependent on the relative positions of query point and rectangle. Consequently, in contrast to the orientation-*insensitive* inflation for window queries, it is not possible to formulate an equation which *generally* estimates the disc access performance for radii-based queries.

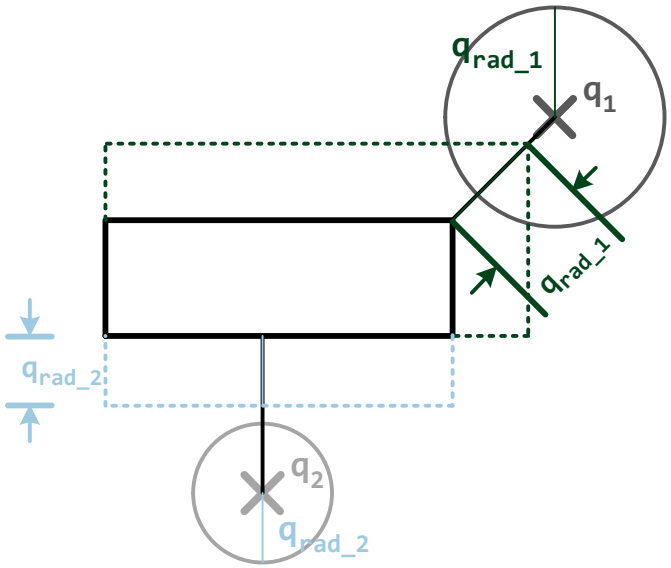


Fig. 118: Inflation of a rectangle for queries based on a query point and a query radius.

for the internal node accesses whereas the QTMBR$^b_{c,a}$ parameterizations, again, suffer from inaccuracies due to their (in the end) quantized approximation of the full-precision MBRs (see Figure 114 on page 375). Table 74 shows that at level 1 of the R-tree, all QTMBR$^b_{c,a}$ parameterizations are better with regard to page accesses than the MBR approach and at level 2, QTMBR_2 (with $b = 12$) is still best. As previously outlined, the problems for QTMBR$^b_{c,a}$ become massive at level 3. Due to the analyses so far, these phenomena caused by the erratic data point distribution of the T collection are easily comprehensible, though. For the R collection, one might expect this change a little, i.e. the QTMBR$^b_{c,a}$ parameterizations might be less favorable at level 1 and level 2 but more competitive at level 3. We come back to this in the evaluation of the R_5 scenario's query results.
Overall, all these outlined aspects, i.e.

— the already excellent performance of the MBR approach,
— the freedom of overlap between the nodes at leaf node level,
— the outward appearance of the summaries,
— the access probability being impacted by the query radii,
— and the erratic data distribution of the T collection

lead to the following outcome: Despite identical R-tree structures and (partly) significant amounts of additionally invested storage space, for the MBR-like R-trees, the MBR approach is hardly (MBRQT^{c,a} parameterizations), only partly (QTMBR_4, for the leaf nodes accesses), or not at all (remaining QTMBR$^b_{c,a}$ parameterizations) improved by our summarization approaches. Hence, since already the MBR-like R-trees hardly profit from our summarization approaches, the probability of achieving improvements

for the corresponding summary-like R-trees is rather low. The results for the summary-like R-trees in the T_5 scenario are outlined in the following.

Query Results for the Summary-like R-trees in the T_5 scenario

Table 77 depicts the kNN query results for the T_5 scenario's summary-like R-trees. For $k = 10$, the MBR approach now requires 18.6 page accesses on average (from which 12.5 are to internal nodes and 6.1 are to leaf nodes), i.e. the results are slightly worse in comparison to its corresponding MBR-like R-tree (18.3 page accesses on average). Thus, the relative query performances between both are the exact opposite of the cumulated indexed surface areas' indication (the summary-like MBR R-tree has better values for all levels), once again showcasing the unreliability of this key figure.

Table 77: kNN query results for the summary-like R-trees in the T_5 scenario.

	MBR	MBRQT_1	MBRQT_2	MBRQT_3	MBRQT_4	MBRQT_5	MBRQT_6	QTMBR_1	QTMBR_2	QTMBR_3	QTMBR_4
k=10											
#Dist2Rect	1,389.2	1,351.0	1,717.5	1,944.4	3,430.1	4,106.5	5,126.3	20,714.8	2,625.4	14,322.2	15,961.0
#Dist2Sum	1,389.2	1,228.6	1,289.5	1,281.5	1,299.5	1,219.4	1,406.4	20,600.7	2,483.4	13,840.7	6,894.7
#PA total	18.6	20.2	20.9	21.1	21.1	20.3	22.2	958.4	52.5	181.5	117.7
#PA IN	12.52	14.11	14.74	15.01	14.80	14.39	16.18	141.02	18.65	148.21	109.30
#PA LN	6.05	6.07	6.12	6.14	6.31	5.86	6.06	817.42	33.90	33.33	8.43
k=1,000											
#Dist2Rect	2,228.1	2,137.4	2,658.1	3,244.8	4,322.6	5,314.1	6,796.4	24,370.3	3,779.0	16,977.5	20,888.3
#Dist2Sum	2,228.1	1,998.5	2,063.6	2,072.5	2,065.9	2,038.3	2,222.2	24,252.5	3,635.2	16,420.0	8,867.7
#PA total	321.5	324.8	325.6	325.7	327.1	325.8	327.5	2,130.7	461.8	571.8	469.8
#PA IN	19.63	22.78	23.45	23.95	23.50	23.64	25.40	166.02	28.46	176.55	143.91
#PA LN	301.89	302.00	302.18	301.72	303.64	302.20	302.06	1,964.67	433.38	395.29	325.84

However, for the summary-like R-trees in the T_5 scenario, the MBR approach is now the best technique. All the MBRQT^{c,a} parameterizations are worse with regard to the overall result. This also applies to internal node accesses *and* leaf node accesses except in one case: MBRQT_5 requires only 5.86 leaf node accesses on average which makes it the best of all techniques in this regard. The outcome for the internal node accesses is not surprising, though, because the summary-like MBRQT^{c,a} R-trees feature roughly 33% more internal nodes than the summary-like MBR R-tree (see Table 56 on page 346). Also, the fanout of the MBRQT^{c,a} R-trees is lower than for the MBR R-tree (roughly 87.7 as opposed to 116.8). In general, there are four aspects to consider when analyzing the internal node access performances of summary-like R-trees:[305]

[305]For the MBR-like R-trees, only the spatial accuracy of the summaries is relevant as the other aspects are identical.

a) The height of the R-tree (the lower, the more favorable).
b) The spatial accuracy of the summaries (which includes the possibility of unsuitable assignments of data points to nodes, i.e. the eventual degeneration of the R-tree).
c) The number of internal nodes in the R-tree (the lower, the more favorable).
d) The fanout of the R-tree (the greater, the more favorable).

Obviously, the aspects a), b), and d) are fairly closely linked: The lower the fanout the greater the amount of internal nodes. The more internal nodes, the greater the height (in the long run). Nevertheless, they cannot be reduced to one key figure as e.g. the fanout could be misleading in case of that the leaf node summaries and the internal node summaries differ strongly with regard to their memory consumption rates—which might well be due to the different summary calculation processes for internal node summaries and leaf node summaries (especially because of target depth td which is used for the internal nodes' summary-from-summaries calculation). Overall, it is not possible to assess which aspect has how much impact on the final results. Thus, we cannot attribute worse internal node access performance solely to degenerations of the respective R-trees.[306] However, the query results clearly show that the overall composition of the summary-like MBR R-tree (which in general has a definitive edge over the summary-like MBRQT^{c,a} R-trees for a), c), and d)) is more suitable.
When comparing the amounts of internal node accesses of the MBRQT^{c,a} parameterizations with each other, it is interesting that the higher the MBRQT^{c,a} summaries are parameterized (i.e. the greater their spatial accuracy should be), the worse their internal node access performances tend to be. For example, MBRQT_1 requires only 14.11 internal node accesses on average whereas MBRQT_6 necessitates 16.18. When reassessing the cumulated indexed surface areas of the summary-like R-trees in the T_5 scenario (see Table 59 on page 352), it is also evident that the lower MBRQT^{c,a} parameterizations, by tendency, exhibit slightly better results than the higher MBRQT^{c,a} parameterizations.[307] Even though the unreliability of this key figure has been demonstrated on several occasions before, these results are unexpected because the different MBRQT^{c,a} parameterizations

[306] The case is different for the leaf node access performance as it is basically only dependent on the spatial accuracy of the summaries and the average amount of data points in the leaf nodes, i.e. the leaf nodes' average memory utilization. Hence, in case the leaf nodes' memory utilization rates are comparable for a specific techniques' MBR-like and summary-like R-trees, deteriorations of the leaf node access performance have to be caused by a degeneration of the summary-like R-tree, i.e. unsuitable assignments of data points to leaf nodes.

[307] Note that during the corresponding structural analysis, we did not go into detail for the resulting cumulated indexed surface areas to avoid an overinterpretation of the results (because of the apparent influence of randomness on the key figures). Therefore, this anomaly has not been considered in the corresponding structural analysis.

are instances of the same approach, i.e. they are identical in character.[308] For the MBRQT^{c,a} R-trees, height, number of internal nodes, and fanout are virtually identical. In theory, the spatial accuracy of the summaries should be in favor of the higher MBRQT^{c,a} parameterizations—which nevertheless show worse results. Therefore, we think that both phenomena (internal node access performances and cumulated indexed surface areas) have the same origin: The SELECTBEST$(.)$-method's inappropriateness which has already been indicated by the distributions of the spatial scattering of the leaf nodes' data points in the structural analysis (see Figure 110 on page 361). Naturally, not only the leaf node access performance will suffer from degenerations of the R-tree (i.e. inappropriate assignments of data points to nodes) but also the internal node access performance.

Consider the example depicted in Figure 119 as an illustration for the SELECTBEST$(.)$-method's inappropriateness with regard to MBRQT^{c,a} parameterizations: There are two MBRQT^{c,a} summaries for which the basic MBRs are refined by an MBR-interior quadtree. For both refinement quadtrees, three cells are 'black', i.e. each summary has three indexed areas. The newly inserted data point dp is not contained in any of the indexed areas. In accordance with the SELECTBEST$(.)$-algorithm, the distances between dp and the respective basic MBRs' center points have to be determined and used as a decision criterion.[309] In this example, the green summary's center point is closer to dp and therefore, dp is assigned to the green summary's associated node. However, when looking at the respective indexed areas, an assignment of dp to the blue summary's node would certainly be more appropriate. The more detailed the refining quadtree (especially the more quadtree cells can be built, i.e. the greater parameter c), the more likely suchlike misassignments. Thus, we think the anomalies of the MBRQT^{c,a} parameterizations' intra-approach results with respect to the internal node accesses and the cumulated indexed surface areas are caused by the SELECTBEST$(.)$-method because all the other aspects (height, number of internal nodes, fanout) are virtually identical for all the MBRQT^{c,a} parameterizations' R-trees. However, this means even though e.g. MBRQT_6 did not appear to be at a massive disadvantage with regard to the leaf nodes' spatial scattering (see Figure 109 on page 356), the consequences are obviously still measurable.

The leaf nodes' average memory utilization rates are identical for the MBRQT^{c,a} parameterizations (see Table 56 on page 346). Hence, like for

[308] Therefore, for *inter*-approach comparisons of the MBRQT^{c,a} parameterizations, the cumulated indexed surface areas should be a more suitable indicator of the nodes' average spatial accuracy compared to *intra*-approach assessments such as between e.g. the MBR approach and QTMBR$^{b}_{c,a}$ parameterizations.

[309] As outlined in section 10.2.1, actually, the MBR of its *indexed areas* has to be determined for each node. The center point of this MBR is utilized for the assignment decisions. However, for MBRQT^{c,a}, the MBR of the indexed areas corresponds to the basic MBR.

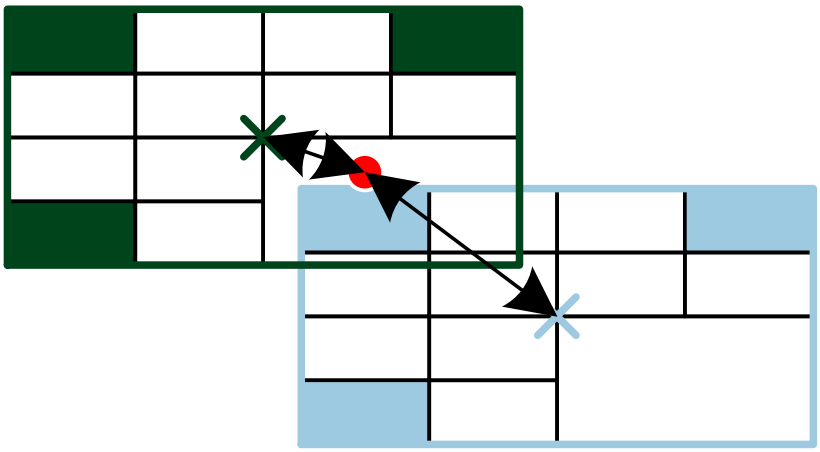


Fig. 119: Example visualization illustrating the inappropriateness of the applied SELECTBEST(.)-method for MBRQT^{c,a} summaries.

the internal node accesses, the prerequisits for the leaf node accesses are the same for all MBRQT^{c,a} parameterizations. Therefore, it is no surprise that the leaf node access results show similar patterns to the results of the internal nodes. However, the MBRQT^{16,a} parameterizations (i.e. MBRQT_4 to MBRQT_6) now also seem to benefit a little from their greater amount of quadtree cells c (yielding a greater spatial accuracy of the summaries through a more detailed refinement of the basic MBRs—which partially outweighs the misassignments). Nevertheless, when comparing e.g. the leaf node access results of MBRQT_5 and MBRQT_6 (5.86 and 6.06, both with $c = 16$) *or* of MBRQT_1 and MBRQT_3 (6.07 and 6.14, both with $c = 8$), the relative results of the MBRQT^{c,a} parameterizations featuring the same c value do still not meet the expectations. In total, a greater fuzziness in the results of the leaf node accesses can be noted for MBRQT^{c,a}. As mentioned before, there is certainly a trade-off between the general spatial accuracy of the summaries and the induced degeneration of the corresponding R-tree. The *overall* performances of the summary-like MBRQT^{c,a} R-trees deteriorate by between 11% (MBRQT_1) and 28% (MBRQT_6) in comparison to their MBR-like counterparts.
For the summary-like QTMBR$^{b}_{c,a}$ R-trees, the results are *significantly* worse than for their corresponding MBR-like R-trees. The total amount of page accesses increases by between 21% (QTMBR_2) and 104% (QTMBR_4). Hence, QTMBR_4 now requires more than twice as many page accesses on average (117.7 instead of 57.6). The deterioration in performance applies to both internal node accesses (109.3 instead of 52.8) and leaf node accesses (8.43 instead of 4.82). For the internal nodes, it is not unexpected as the number of internal nodes is significantly greater in the summary-like QTMBR_4 R-tree (91% more internal nodes than in the cor-

[310] This is because the average leaf node memory utilization of the summary-like QTMBR_4 R-tree is slightly greater (70.1% as opposed to 69.9% for the MBR-like QTMBR_4 R-tree).

responding MBR-like QTMBR_4 R-tree) while simultaneously, the fanout is clearly lower (61.1 as opposed to 116.7). Nevertheless, the number of leaf nodes is even slightly lower for the summary-like QTMBR_4 R-tree.[310] Consequently, its poor *leaf node* access performance is not caused by a low leaf node memory utilization rate and is certainly also not explainable by a surplus on *internal nodes*. As already mentioned several times in the structural analysis, we suspect the summary-like QTMBR$^b_{c,a}$ R-trees to suffer significantly from unsuitable assignments of newly inserted data points to nodes. The unsuitability of the SELECTBEST(.)-method for QTMBR$^b_{c,a}$ can be illustrated by using a simple example.

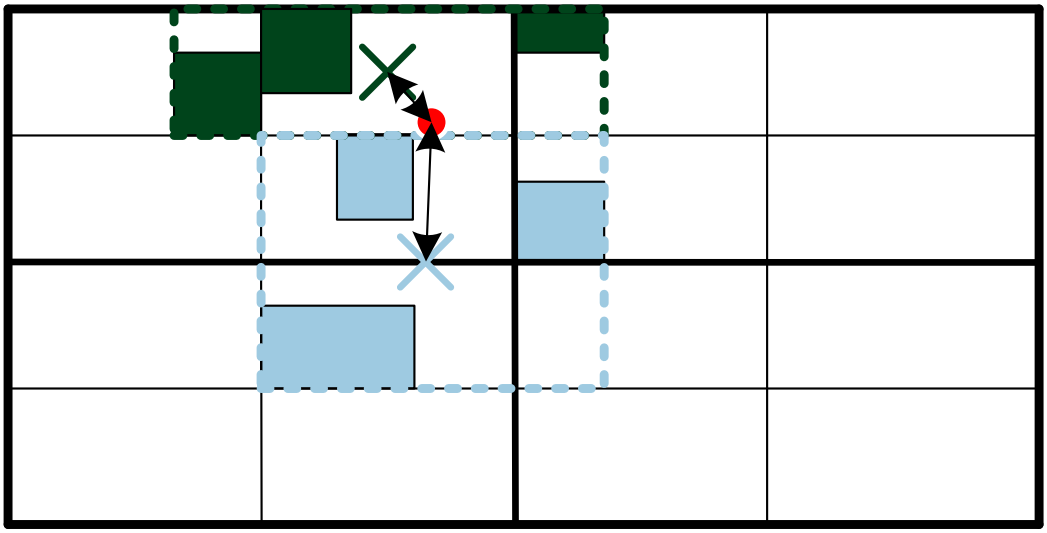


Fig. 120: Example visualization illustrating the unsuitability of the applied SELECTBEST(.)-method for QTMBR$^b_{c,a}$ summaries.

In Figure 120, two QTMBR$^b_{c,a}$ summaries are depicted. Again, neither covers dp with its indexed areas. Anew, dp's distance to the center points of the respective indexed areas' MBRs has to be utilized as decision criterion. Consequently, dp is assigned to the node associated with the green summary—which is similarly inappropriate as in the situation depicted in Figure 119. For QTMBR$^b_{c,a}$, we also presume that the more quadtree cells can be built theoretically (i.e. the greater parameter c or, for the internal node summaries, the greater target depth td), the more likely such misassignments are.[311] Since at the end of the corresponding R-trees' construction phases, the internal nodes' MBRQT^{c,a} respectively QTMBR$^b_{c,a}$ summaries often correspond to the equivalent MBR summaries respectively approximations of them, we suspect that the main problems occur dur-

[311]Note that in Figure 109 on page 356, the curve of QTMBR_1 showed the greatest spatial scattering for the leaf nodes and therefore implied that the degeneration of the QTMBR_1 R-tree is the greatest. However, it is difficult to make comparisons whether QTMBR_1 or QTMBR_4 suffers more as the results for both are too different. Therefore, we only want to state that it appears reasonable to us that more accurate QTMBR$^b_{c,a}$ parameterizations should suffer more from degenerations of their summary-like R-trees.

ing the early construction stages. However, since we did not capture intermediate results during the R-trees' constructions, it is hard to assess to which extent the *internal node* access performances of the summary-like MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ R-trees suffer from the degeneration of their R-trees.[312]

For QTMBR$^{b}_{c,a}$, the following anomalies and consequential conclusions are obvious from the results of the summary-like QTMBR$^{b}_{c,a}$ R-trees:

— Based on the results of QTMBR_4 (8.43 leaf node accesses for its summary-like R-tree as opposed to 4.82 leaf node accesses for its MBR-like R-tree, *despite comparable leaf node memory utilization rates*), it is indisputable that in general, the assignment of data points to leaf nodes is deteriorated when using QTMBR$^{b}_{c,a}$ summaries in the R-tree construction phase. This is underlined by the fact that with regard to the leaf node accesses, all summary-like QTMBR$^{b}_{c,a}$ R-trees are significantly worse than their MBR-like counterparts. Since the deteriorations in leaf node access performance are significantly greater than for the MBRQT^{c,a} parameterizations (see Table 78), we think that the QTMBR$^{b}_{c,a}$ parameterizations suffer to a significantly larger extent from the unsuitability of the SELECTBEST(.)-method. This is also evident from the figures displaying the distribution of the spatial scattering of the leaf nodes' data points.

Table 78: Deterioration of the summary-like R-trees compared to their corresponding MBR-like R-trees with regard to the leaf node access performance (T_5 scenario).

technique	MBR	MBRQT_1	MBRQT_2	MBRQT_3	MBRQT_4	MBRQT_5	MBRQT_6	QTMBR_1	QTMBR_2	QTMBR_3	QTMBR_4
deterioration	3.8%	5.2%	7.7%	10.0%	10.0%	5.1%	13.8%	50.6%	26.1%	78.6%	74.9%

— Also, all summary-like QTMBR$^{b}_{c,a}$ R-trees are now significantly worse with regard to internal node accesses. Hereby, when specifically comparing the summary-like and the MBR-like QTMBR_1 R-tree (141.0 internal node accesses for the former as opposed to 113.7 for the latter), it is derivable that the degeneration has to play a significant role as the changes in the numbers of internal nodes (72,618 instead of 66,169) and the fanout (151.1 instead of 116.7) do not appear to be unfavorable for the summary-like QTMBR_1 R-tree (the height is 5 for both R-trees).

— In total, out of the QTMBR$^{b}_{c,a}$ parameterizations, QTMBR_4 suffers most when juxtaposing the *relative* overall results of the summary-like and the MBR-like R-trees. QTMBR_2 holds up best in the relative results

[312]For the leaf nodes, in case similar memory utilization rates are given, it is easily assessable.

with 'only' 21% more total page accesses on average (QTMBR_1: 46%, QTMBR_3: 48%, QTMBR_4: 104%)—which is still significant. For the summary-like and the MBR-like QTMBR_2 R-trees, height (both 5) and fanout (115.4 as opposed to 116.7) are comparable, only the number of internal nodes is 19.5% greater for the summary-like QTMBR_2 R-tree. This is fairly similar to its increase in total page accesses. Nevertheless, the total amount of page accesses for QTMBR_2 is dominated by the amount of leaf node accesses (33.9) and *not* by the amount of internal node accesses (18.7). Consequently, the performance deterioration is mainly caused by the leaf node accesses (for which the MBR-like QTMBR_2 R-tree requires 26.9 accesses on average) and *not* by internal node accesses (16.7).

— Due to the aspects discussed in the latest two indents, it seems very likely that for $\text{QTMBR}^{b}_{c,a}$, in general, most of the performance deteriorations are caused by the degeneration of their summary-like R-trees. Therefore, we think that the unsuitability of the SELECTBEST(.)-method has significant impacts on all $\text{QTMBR}^{b}_{c,a}$ parameterizations.

— Considering the comparatively small deteriorations of the summary-like QTMBR_2 R-tree's performance (for both internal node accesses as well as leaf node accesses) in comparison to its MBR-like counterpart, the negative effects of the unsuitable SELECTBEST(.)-method are obviously mitigatable (but not completely avoidable) by utilizing appropriate parameterizations.

— As indicated in the structural analyses and already evident for the MBR-like R-trees, a high quantization accuracy (as used for QTMBR_2 with $b = 12$) proves to be advantageous for the internal node accesses when simultaneously, a target depth td is used for the internal nodes' summary-from-summaries calculation.

— Interestingly, despite its greater average leaf node memory utilization (66.4%), QTMBR_3 requires about the same amount of leaf node accesses (33.33) as QTMBR_2 (59.1%, 33.90). Also, QTMBR_3 requires more accesses to internal nodes than QTMBR_1 (148.2 as opposed to 141.0) despite the same prerequisits ($td = 2$ and $b = 8$).[313] This might be an effect caused by the differences in their R-tree structures as the QTMBR_1 R-tree is better than the QTMBR_3 R-tree with regard to fanout (151.1 as opposed to 95.0) and number of internal nodes (72,618 as opposed to 85,687). Hence, QTMBR_3 would be a very suitable test subject to in-

[313] Not to mention in comparison QTMBR_2, although the latter admittedly has the best internal node memory utilization rate of all $\text{QTMBR}^{b}_{c,a}$ parameterizations—which certainly also plays a role in the comparative advantageousness of QTMBR_2 (in addition to its evidentially smallest indexable spatial units for internal node summaries). Note, though, that we do not consider the internal nodes' memory utilization rate as a factor with *direct* influence on the internal node access performance because it derives immediately from the average memory consumption of the summaries (which is correlated with the summaries' spatial accuracy) and the fanout.

vestigate eventual degenerations of the QTMBR$^{b}_{c,a}$ R-trees during the construction phase.

In a comparison with the MBR approach and the MBRQT^{c,a} parameterizations, for the QTMBR$^{b}_{c,a}$ parameterizations, the subpar memory utilization rates for both the internal nodes (QTMBR_1, QTMBR_3, QTMBR_4) as well as the leaf nodes (QTMBR_1, QTMBR_2, slightly for QTMBR_3) have to be taken into account in addition. Especially the latter is disadvantageous as a lower average amount of data points stored in the leaf nodes implies that usually, the query result's data point set is dispersed across more leaf nodes. Hence, a lower memory utilization rate automatically leads to a greater amount of leaf node accesses (irrespective of the summaries' spatial accuracy).
To get a sense of how large this effect can be, consider Table 79. There, the leaf node accesses are contrasted with the mathematically determined number of considered data points during query processing. It can be seen that for the techniques featuring low leaf node memory utilization rates (QTMBR_1, QTMBR_2, QTMBR_3), the relative results with regard to the leaf node accesses are significantly worse compared to the (already bad) results for the considered data points. For example, QTMBR_2 considers 473.8% of the data points of the MBR approach (assuming that the MBR approach's amount of considered data points equals 100%) but conducts 560.3% of the MBR approach's leaf node accesses.

Table 79: Comparison of the techniques' absolute and relative results with regard to amount of leaf node accesses and the number of considered data points (summary-like R-trees in the T_5 scenario). For the relative results, the MBR approach is the benchmark (i.e. its results correspond to 100%).

	MBR	MBRQT_1	MBRQT_2	MBRQT_3	MBRQT_4	MBRQT_5	MBRQT_6	QTMBR_1	QTMBR_2	QTMBR_3	QTMBR_4
avg mem util. leaf nodes	69.9%	69.9%	69.9%	69.9%	69.9%	69.9%	69.9%	49.1%	59.1%	66.4%	70.1%
leaf node accesses											
absolute	6.05	6.07	6.12	6.14	6.31	5.86	6.06	817.4	33.90	33.33	8.43
relative	100%	100.3%	101.2%	101.5%	104.3%	96.9%	100.2%	13,511%	560.3%	550.9%	139.3%
considered data points (mathematically determined)											
absolute	21.14	21.21	21.39	21.46	22.05	20.48	21.18	2,006.8	100.2	110.7	29.55
relative	100%	100.3%	101.2%	101.5%	104.3%	96.9%	100.2%	9,491%	473.8%	523.3%	139.7%

A definitive clarification of the reasons behind all the observed anomalies would require additional experiments. We refrain from these at this point as it is evident that the applied SELECTBEST$(.)$-method is unsuitable for QTMBR$^{b}_{c,a}$. Therefore, the method's adaption to QTMBR$^{b}_{c,a}$ and a subsequent evaluation is much more reasonable. This is also part of future work, though.

It is difficult to provide a definitive proof for the apparently occuring misassignment of data points to nodes during the construction phase of the summary-like QTMBR$^{b}_{c,a}$ R-trees. Nevertheless, in the following, we additionally present some visualizations that support this hypothesis as circumstantial evidence. For the visualizations, the summary-like QTMBR_4 R-tree has been loaded and its QTMBR_4 summaries were replaced with MBR summaries. As a benchmark, also the summary-like MBR R-tree was loaded but its MBR summaries were retained. Both R-trees were utilized to create comparative images which showcase the degeneration of the summary-like QTMBR_4 R-tree. Of course, they are difficult to compare as already at level 2 of the respective R-trees, the numbers of internal nodes differ (see Table 60 on page 352). However, several interesting aspects are observable in the respective images. As a first example, Figure 121 shows a set of nodes at level 2 of the MBR R-tree (left) respectively of the QTMBR_4 R-tree (right). For both, these nodes are the children of one specific level-1-node covering most of the western hemisphere. It is easily recognizable that for the summary-like MBR R-tree (left), the level-2-nodes' MBRs actually reproduce the shape of the United States in a fairly well-defined fashion. In contrast, for the summary-like QTMBR_4 R-tree (right), the shape of the United States is not nearly as well discernible on basis of the nodes' MBRs. Also, Hawaii has a node of its own in the MBR-like R-tree (highlighted by the yellow circle in the left image of Figure 121) which is not the case for the QTMBR_4 R-tree. Hence, in the summary-like MBR R-tree, also larger 'clusters' of data points at the *internal nodes*' levels are obviously held together better (which was already the conclusion regarding the techniques' results for the spatial scatterings of the *leaf nodes*' data points). We consider this as first circumstantial evidence for our assertion regarding the misassignment of data points to *internal nodes* for the summary-like QTMBR_4 R-tree.

In Figure 122, the MBRs of some other level-2-nodes are showcased. Again, all nodes are children of a specific node at level 1 in the summary-like MBR R-tree (top) respectively the summary-like QTMBR_4 R-tree (bottom). In both R-trees, this specific level-1-node's MBR covers the entire eastern hemisphere and large parts of the western hemisphere.[314] In this comparison, three noteworthy differences are observable:

— In the QTMBR_4 R-tree, there is more overlap between the nodes' MBRs (the darker the blue, the greater the amount of MBRs which overlap at the corresponding locations).
— In specific regions of increased data point density, the MBRs of the MBR R-tree are clearly more concise, i.e. the corresponding nodes' data points

[314] However, both summary-like R-trees feature not only the two nodes whose children are depicted in Figure 121 and Figure 122 but a total of five nodes at level 1 (see Table 60 on page 352). In general, in both R-trees, the MBRs of the level-1-nodes are very similarly arranged to those in Figure 96 on page 325.

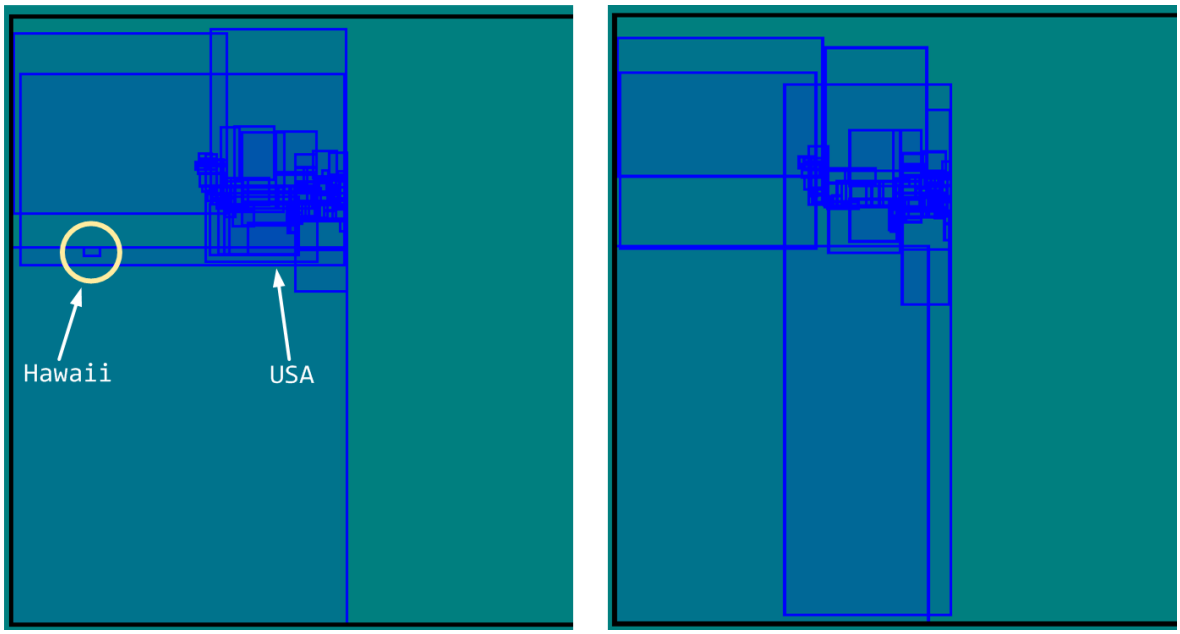


Fig. 121: MBRs of selected nodes at level 2 of the summary-like MBR R-tree (left) and the summary-like QTMBR_4 R-tree (right).

are more cleanly separated from each other. Some example regions are highlighted by yellow circles.

— The MBRs of the QTMBR_4 R-tree's nodes extend significantly further into the western hemisphere. This is indicated by the dashed, vertical red line which overspans both images of Figure 122. The obviously less sharp demarcation between high-level nodes in the QTMBR_4 R-tree is already minimally present in Figure 121. There, the MBRs of the QTMBR_4 R-tree extend *slightly* further east (hardly visible in the images but verified by additional measurements). Hence, the two specific level-1-nodes overlap each other to substantial extents in the QTMBR_4 R-tree. In contrast, in the MBR R-tree, the two corresponding level-1-nodes overlap only minimally and are separated from each other much more cleanly.

All these observations underpin our underlying hypothesis. Finally, Figure 123 shows snippets of the images in Figure 122. There, the increased overlap of the MBRs of the QTMBR_4 R-tree's nodes is even better observable than in Figure 122. Furthermore, the regions in which the demarcations between the MBRs are cleaner and the MBRs are delineated much more concisely in the MBR R-tree as opposed to the QTMBR_4 R-tree are highlighted by yellow circles. Hence, Figure 122 and Figure 123 indicate, anew, that the assignment of data points to internal nodes is much more targeted and differentiated for the MBR approach. Overall, the results of the structural analyses, the query results, the theoretical considerations regarding the SELECTBEST(.)-method, and the just discussed circumstantial evidence derivable from the visualizations of both example R-trees lead us to the conclusion that the apparently occuring misassignment of data

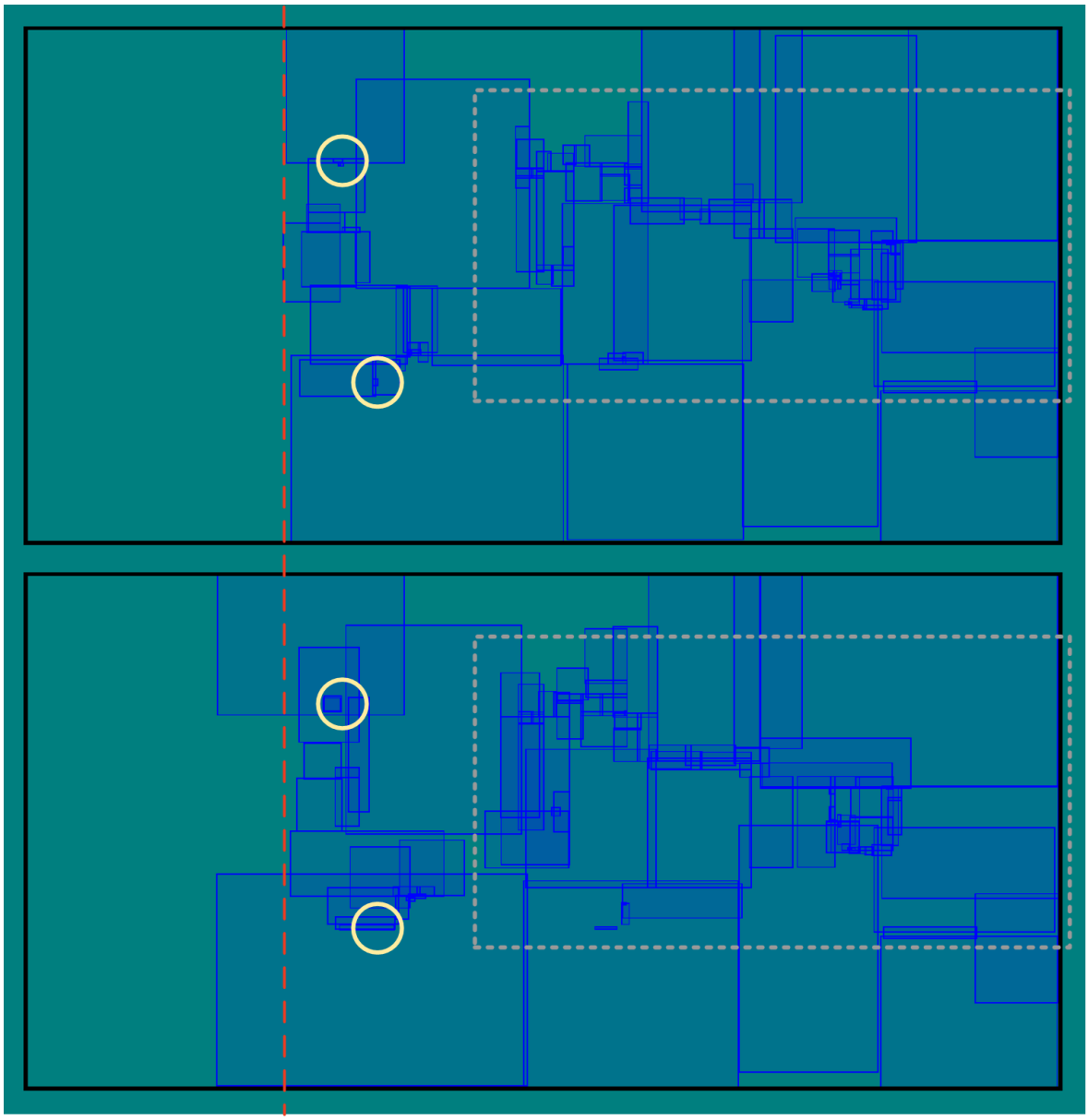


Fig. 122: MBRs of selected nodes at level 2 of the summary-like MBR R-tree (top) and the summary-like QTMBR_4 R-tree (bottom). Additionally, diverse apparent differences between both R-trees (yellow circles) as well as the snippet regions of Figure 123 (dashed, grey rectangles) are indicated.

points to nodes is a big problem for the summary-like $\text{QTMBR}^{b}_{c,a}$ R-trees. It is also problematic for the summary-like $\text{MBRQT}^{c,a}$ R-trees even though the adverse effects are much less pronounced. Hence, we want to emphasize again that future work should highly prioritize the development of appropriate $\textsc{selectbest}(.)$-algorithms for $\text{QTMBR}^{b}_{c,a}$ and $\text{MBRQT}^{c,a}$.
With regard to the distance calculations, except MBRQT_6, each $\text{MBRQT}^{c,a}$ parameterization requires less distance calculations to summary instances

[315] In general, the number of *considered* nodes (i.e. nodes whose summaries are used for distance calculations in the query processing) can also be *estimated* by the number of accesses to internal nodes multiplied with the fanout of the corresponding R-tree.

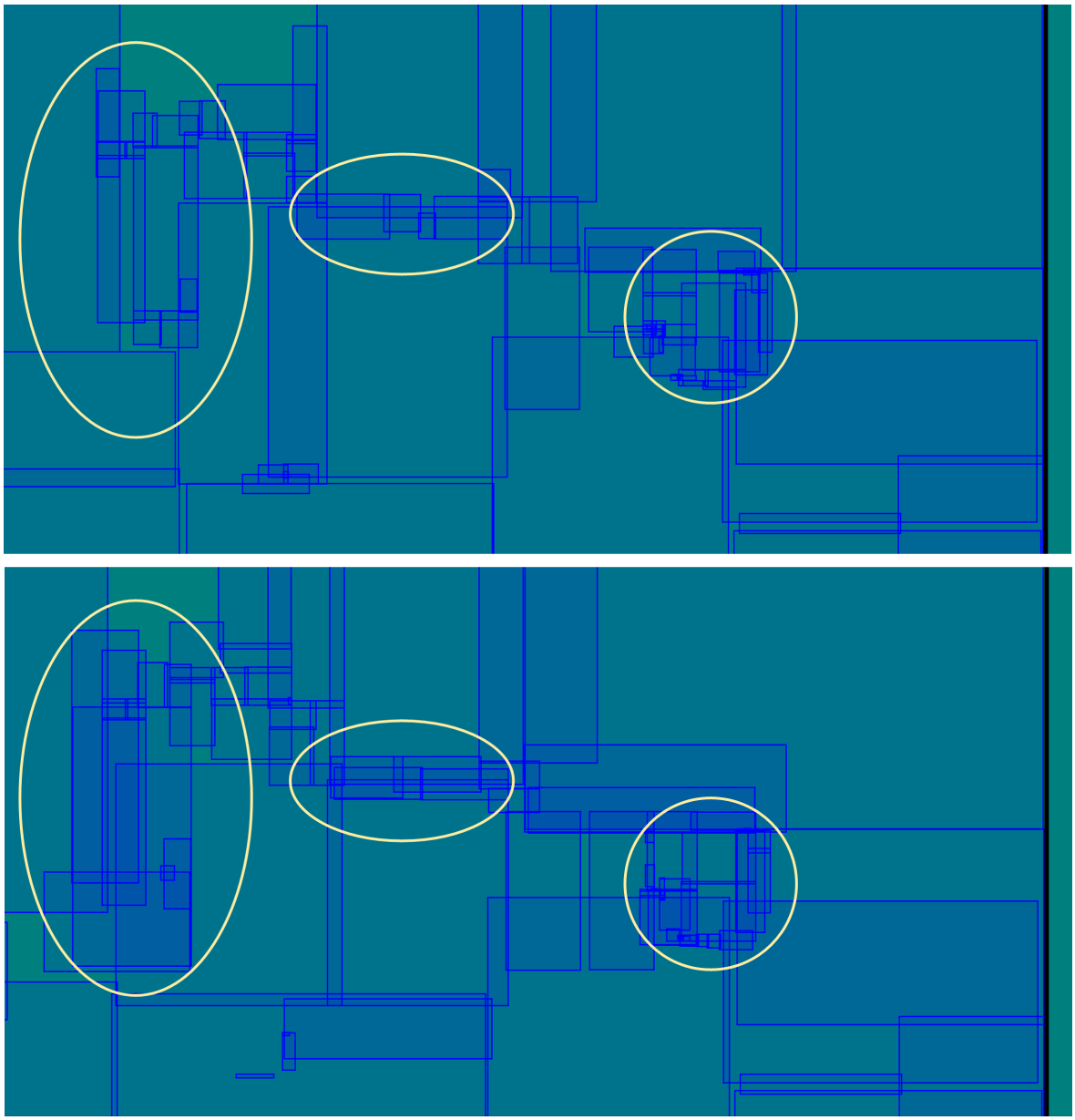


Fig. 123: Snippet of Figure 122 showing the MBRs of diverse level-2-nodes for the summary-like MBR R-tree (top) and the summary-like QTMBR_4 R-tree (bottom). Again, apparent differences are highlighted.

than the MBR approach (see '#Dist2Sum'-row in Table 77 on page 386). This means that in total, less nodes are *considered*[315] during the query processing—despite the greater amount of page accesses to internal nodes for the MBRQT^{c,a} parameterizations. The reason for this is the lower fanout of the summary-like MBRQT^{c,a} R-trees: On average, an internal node in the summary-like MBRQT^{c,a} R-trees has fewer child node entries (about 87.7) than an internal node in the summary-like MBR R-tree (116.8). A lower amount of considered nodes in comparison to the MBR approach has already been observed for the *MBR-like* MBRQT^{c,a} R-trees (see Ta-

ble 71 on page 370).[316] Interestingly, for MBRQT_1, the total amount of point-to-rectangle distance calculations ('#Dist2Rect'-row) is now slightly lower than for the MBR approach (1,351.0 as opposed to 1,389.2). This is a very unexpected result. However, as the distance calculations are non-significant with regard to the overall runtime and additionally, our a priori assumption 'less page accesses are only achievable by means of a greater amount of distance calculations' still holds (although it now applies to the MBR approach, i.e. the other way round than originally assumed), this is just a side note. In general, for the remaining MBRQT^{c,a} parameterizations, the number of distance calculations is *slightly* increased while for the QTMBR$^{b}_{c,a}$ parameterizations, it is *significantly* increased (both compared to the results of the corresponding MBR-like R-trees).
Overall, in comparison to the MBR-like R-trees, the page access results are slightly worse for the MBR approach (1.6% more page accesses on average). This change is within the expectable, natural variations of results. In contrast, all MBRQT^{c,a} parameterizations are at least 11% worse. However, the QTMBR$^{b}_{c,a}$ parameterizations suffer to a significantly greater extent (at least 21% deterioration), and are now even further behind the other techniques.
For $k = 1,000$, the observations made for $k = 10$ are pretty much confirmed, there are no apparent discrepancies to the results just discussed.

Table 80: Range query results for the summary-like R-trees in the T_5 scenario.

	MBR	MBRQT_1	MBRQT_2	MBRQT_3	MBRQT_4	MBRQT_5	MBRQT_6	QTMBR_1	QTMBR_2	QTMBR_3	QTMBR_4
r=10	result size = 21,533.0										
#Dist2Rect	9,338.2	9,140.6	10,130.9	13,966.1	11,614.2	13,643.8	18,950.3	38,123.8	12,795.3	27,043.6	45,485.6
#Dist2Sum	9,338.2	8,980.2	9,027.6	9,057.1	9,045.2	9,066.3	9,210.9	37,998.2	12,647.9	26,229.2	18,440.0
#PA total	6,273.0	6,291.8	6,292.6	6,289.6	6,296.2	6,290.0	6,288.1	12,349.5	7,822.3	6,756.5	6,523.8
#PA IN	80.4	102.2	102.6	103.4	102.8	103.3	104.7	261.7	108.1	285.5	319.1
#PA LN	6,192.5	6,189.5	6,190.0	6,186.2	6,193.4	6,186.7	6,183.4	12,087.8	7,714.3	6,471.1	6,204.8
r=1,000	result size = 256,137.0										
#Dist2Rect	78,992.4	79,259.6	87,871.3	122,415.6	84,778.4	97,906.7	142,664.4	132,093.6	95,890.1	96,997.1	205,031.6
#Dist2Sum	78,992.4	78,659.9	78,803.8	78,929.4	79,014.8	78,651.4	79,314.6	131,915.6	95,712.3	94,413.6	85,103.4
#PA total	73,947.7	74,159.8	74,180.2	74,161.5	74,179.9	74,129.3	74,154.4	108,471.8	89,110.5	76,704.8	74,383.4
#PA IN	676.4	895.9	898.5	899.5	900.4	898.0	902.7	894.6	829.3	1,019.8	1,439.6
#PA LN	73,271.3	73,263.9	73,281.7	73,262.0	73,279.5	73,231.3	73,251.7	107,577.2	88,281.2	75,685.0	72,943.7

For the range query results (see Table 80), in comparison to the kNN query results, the MBR approach is now more often outperformed with regard to the *leaf node* accesses—by diverse MBRQT^{c,a} parameterizations (for both $r = 10$ and $r = 1,000$) and by QTMBR_4 (for $r = 1,000$). This is reasonable since due to the large result set sizes, more leaf nodes contain relevant data points and therefore, the suboptimal assignment of data points

[316] However, the reason there is the slightly lower amount of internal node accesses for the MBRQT^{c,a} parameterizations (since the fanout of all MBR-like R-trees is the same).

to leaf nodes is not as severe anymore: More leaf nodes need to be accessed anyway (i.e. the greater dispersion of the query results' data points has mitigated consequences) and furthermore, the misassignment can be compensated by the greater spatial accuracy of the respective techniques' leaf node summaries. Nevertheless, there is no change for the total amount of page accesses since the superiority of the MBR approach for *internal node* accesses is substantially greater in comparison to the kNN queries: For the range queries, the other techniques require at least 27% ($r = 10$) respectively 32% ($r = 1,000$) more accesses to internal nodes than the MBR approach. Due to the generally large amount of internal node accesses as well as the a) significantly larger amount of internal nodes in and b) the (in most cases) lower fanout of the MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ R-trees, this is no surprise, though (even when ignoring the degeneration of the MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ R-trees). At this point, we conclude the evaluation of the query results for the T_5 scenario and move on to the T_25 scenario.

13.3.2. RESULTS FOR THE T_25 SCENARIO. In this section, the evaluation of the query results for the T_25 scenario is conducted. Analogous to the T_5 scenario, first, the results for the MBR-like R-trees are analyzed. Then, we proceed with assessing the outcomes for the summary-like R-trees.

Query Results for the MBR-like R-trees in the T_25 scenario

Table 81 shows the results for the kNN queries. Again, we initially assess the results for $k = 10$. In comparison to the MBR-like R-trees in the T_5 scenario, in the T_25 scenario, the total amount of page accesses is generally lower. Furthermore, this applies to both internal node accesses as well as leaf node accesses. As an example, consider the MBR approach: The total amount of page accesses (13.7 instead of 18.3), the amount of internal node accesses (8.7 instead of 12.5), and the amount of leaf node accesses (4.95 instead of 5.83) are all substantially lower in the T_25 scenario. For both internal and leaf nodes, the general developments of the results are easily explained:

Table 81: kNN query results for the MBR-like R-trees in the T_25 scenario.

	MBR	MBRQT_1	MBRQT_2	MBRQT_3	MBRQT_4	MBRQT_5	MBRQT_6	QTMBR_1	QTMBR_2	QTMBR_3	QTMBR_4
k=10											
#Dist2Rect	1,077.1	1,407.2	2,204.2	2,536.0	2,616.1	4,002.3	4,561.1	3,976.1	1,290.4	3,995.9	8,582.0
#Dist2Sum	1,077.1	1,076.5	1,076.5	1,076.5	1,056.9	1,056.9	1,056.9	3,868.8	1,196.1	3,653.9	2,293.6
#PA total	13.7	13.6	13.5	13.5	13.3	13.0	13.0	154.7	19.8	38.3	22.5
#PA IN	8.72	8.71	8.71	8.71	8.55	8.55	8.55	32.37	9.78	30.58	18.97
#PA LN	4.95	4.85	4.76	4.75	4.70	4.48	4.41	122.33	10.06	7.70	3.56
k=1,000											
#Dist2Rect	1,366.4	1,760.3	2,904.0	3,495.0	3,024.7	4,991.5	5,984.8	4,505.8	1,604.7	4,534.4	10,506.7
#Dist2Sum	1,366.4	1,365.8	1,365.8	1,365.8	1,344.9	1,344.6	1,344.6	4,396.3	1,509.3	4,162.0	2,751.0
#PA total	87.1	86.8	86.5	86.3	86.5	85.6	85.2	328.9	100.6	116.6	95.7
#PA IN	11.10	11.10	11.10	11.10	10.93	10.92	10.92	36.81	12.37	34.85	22.84
#PA LN	75.96	75.75	75.43	75.15	75.57	74.72	74.30	292.05	88.24	81.74	72.83

— There are significantly fewer internal nodes in the MBR-like R-trees of the T_25 scenario than in those of the T_5 scenario. Also, the heights of the former are lower (only 4 instead of 5).
— The maximum leaf node capacity is now 25. Since the leaf nodes' memory utilization rates are comparable in both scenarios, this means that more data points are stored in the leaf nodes on average. Consequently, the query result is usually dispersed across fewer leaf nodes.

Overall, the *relative* performances between the techniques are very similar to those of the MBR-like R-trees in the T_5 scenario: All the MBRQT^{c,a} parameterizations still slightly improve the MBR approach for both internal

as well as leaf node accesses, and MBRQT_6 is yet the best technique of all. As before, the $\text{QTMBR}^{b}_{c,a}$ parameterizations suffer with regard to the internal node accesses, and QTMBR_1 to QTMBR_3 are far behind for the leaf node accesses. Nevertheless, with regard to the latter, QTMBR_4 is still the best technique and its superiority (19% improvement compared to MBRQT_6 and 28% compared to the MBR approach) is now even significantly greater than in the T_5 scenario (where it was 9% and 17%). This shows that with the given assignment of data points to leaf nodes, the MBR approach can be substantially improved with regard to the retrieval performance by using more accurate summaries. However, in this context, it has to be noted that the average summary size of QTMBR_4 increased from 27.6 B in the T_5 scenario to 40.9 B in the T_25 scenario while those of the other techniques remained fairly stable.

Table 82: Comparison of the MBR-like R-trees' relative leaf node access performances for the kNN queries in the T_25 scenario and the T_5 scenario. For all values, the MBR approach serves as a benchmark, i.e. its average amount of leaf node accesses corresponds to 100%.

	MBR	MBRQT_1	MBRQT_2	MBRQT_3	MBRQT_4	MBRQT_5	MBRQT_6	QTMBR_1	QTMBR_2	QTMBR_3	QTMBR_4
T_25 scenario											
k=10	4.95	98%	96%	96%	95%	90%	89%	2,469%	203%	155%	72%
k=1,000	75.96	100%	99%	99%	99%	98%	98%	384%	116%	108%	96%
T_5 scenario											
k=10	5.83	99%	97%	96%	98%	96%	91%	9,308%	461%	320%	83%
k=1,000	301.74	100%	100%	100%	100%	100%	99%	439%	121%	111%	99%

In contrast to the T_5 scenario, with respect to the *leaf node accesses*, the relative performances of the $\text{MBRQT}^{c,a}$ and $\text{QTMBR}^{b}_{c,a}$ parameterizations in comparison to the MBR approach are in general definitely better in the T_25 scenario. Table 82 outlines these relative performances for both scenarios. For example, MBRQT_6 requires 'only' 89% ($k = 10$) respectively 98% ($k = 1,000$) of the MBR approach's leaf node accesses in the T_25 scenario whereas in the T_5 scenario, it is 91% respectively 99%. This is a reasonable outcome: Now, more data points are stored in the leaf nodes on average which leads to a greater spatial spread of the leaf nodes' data point sets. Additionally, not as many zero volume leaf nodes arise. Consequently, more often, regions of dead space can be excluded by the $\text{MBRQT}^{c,a}$ respectively $\text{QTMBR}^{b}_{c,a}$ summaries, or, the other way around, the disadvantages of the MBR summaries' spatial coarseness are more pronounced. This also shows in the numbers of considered data points per query: Mathematically, for the 10NN queries and the MBR approach, 84.77 data points[317] are examined until the query result is unambiguously determined. In the T_5

[317] The given value arises as $4.95 \cdot (25 \cdot 0.685) = 84.77$.

scenario, only 20.38 data points have been considered. Hence, the *potential* for improvements in the T_25 scenario should be greater than in the T_5 scenario.[318] Note that even though QTMBR_1 to QTMBR_3 are still worse than the MBR approach (due to the problems with their smallest indexable spatial units by virtue of the T collection's erratic data point distribution), their *relative* leaf node access performances are significantly better now (see Table 82).

As a universal insight with regard to improving the leaf node access performance of the MBR approach, it has to be recognized that the MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summaries cannot realize their potential like in the distributed application scenario because of two simultaneously applying, but conflicting aspects:

— Generally, the greater the amount of leaf nodes which store relevant data points, the more likely it is that improvements with regard to the leaf node accesses can be achieved.[319] Hence, from a viewpoint considering the R-tree structure, it is advantageous if the leaf node capacity is *low*.
— The more data points are stored in a leaf node, the greater their spatial spread. As a consequence, the ability of the MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summaries to reduce the indexed surface areas and to exclude dead space can be exploited more profitably. Thus, from a viewpoint considering the summaries, it is advantageous if the leaf node capacity is *high*.

Obviously, both aspects do not fit together. Since the relative improvements over the MBR approach are greater in the T_25 scenario, it seems like the second aspect is more important with regard to reducing leaf node accesses. Nevertheless, it shows that our summarization approaches, in some sense, are caught up in a conceptual dilemma.

The query results for $k = 1,000$ exhibit no unexpected changes. The full-precision MBR-based techniques' general convergence for the amount of internal node accesses when querying for large result sets is slightly amplifying. As a consequence, the MBR approach now features the same average values as MBRQT_1 to MBRQT_3. Apart from this, there are no noteworthy changes in the results.

The range query results do not show any big surprises either.[320] In general, the amount of leaf node accesses is about 4.8 to 4.9 times lower than in the T_5 scenario which roughly matches the upgrade of the leaf node capac-

[318] Of course, it has to be considered that for the MBR-like R-trees in the T_25 scenario, a leaf node stores $25 \cdot 68.5\% = 17.13$ data points on average, and that the query result size is only $k = 10$. Hence, even if only two leaf nodes are accessed while processing the query, it is extremely likely that way more than 20.38 data points are stored in these two leaf nodes. Consequently, with regard to the numbers of examined data points, achieving similar results as in the T_5 scenario is illusionary.

[319] This is because then, more leaf nodes have to be accessed mandatorily, and therefore, it is more likely that unnecessary accesses are performed because of the spatial coarseness of the MBR summaries.

[320] Note that therefore, the corresponding range query result table is not depicted.

ity. Most noteworthy, for $r = 1,000$, QTMBR_4 (with its total amount of 15,172.0 page accesses on average) is now better than the MBR approach and MBRQT_1 in the *total amount of page accesses*—by 1.7 respectively 0.2 page accesses.[321] This is due to the drastically diminished share of internal node accesses in the total amount of page accesses for $r = 1,000$. Therefore, for the first time, the advantages of QTMBR_4 at leaf node level can fully compensate its shortcomings concerning the internal node accesses. However, for the range queries, the *relative* page access performances between the MBR approach, the MBRQT^{c,a} parameterizations, and QTMBR_4 are very similar and always within the range of less than 1% difference.
For the point-to-recangle distance calculations, the overhead of the MBRQT^{c,a} parameterizations in comparison to the MBR approach is now significantly greater than in the T_5 scenario. This is just as expected since the structural analysis revealed that due to the greater spatial spread of the leaf nodes' data points, the average number of indexed areas per summary is also increased for all MBRQT^{c,a} parameterizations. For the QTMBR$^{b}_{c,a}$ parameterizations, the corresponding developments of the distance calculation results between the T_5 and the T_25 scenario are not reasonably comparable because the changes in the total amount of page accesses in relation to the MBR approach are too large. In general, the QTMBR$^{b}_{c,a}$ parameterizations still require the greatest amounts of point-to-rectangle distance calculations. Nevertheless, all these numbers are in ranges which are absolutely uncritical for the overall query runtimes.

Query Results for the Summary-like R-trees in the T_25 scenario

In the following, the query results for the T_25 scenario's summary-like R-trees are evaluated. As always, at the beginning, we consider the kNN query results for $k = 10$. The MBR approach requires 13.6 page accesses on average from which 8.7 are to internal nodes and 4.9 are to leaf nodes (see Table 83). With these results, the MBR approach is the best technique for the summary-like R-trees in the T_25 scenario. Anew, this also applies to both internal node accesses as well as leaf node accesses with the single exception of MBRQT_5 being minimally better for leaf node accesses (4.84 as opposed to 4.89 for the MBR approach). Thus, in this regard, the results are exactly the same as for the summary-like R-trees in the T_5 scenario.
In general, the superiority of the MBR approach over the MBRQT^{c,a} parameterizations for the internal node accesses is now more pronounced than in the corresponding T_5 scenario. The reason is that the height of the summary-like MBR R-tree in the T_25 scenario is only 4 whereas all summary-like MBRQT^{c,a} R-trees exhibit a height of 5 (see Table 61 on page

[321] It has to be noted, though, that these minimal improvements in page access performance are completely out of proportion to the internal nodes' respective memory consumption: QTMBR_4 requires 5,722.6 B on average whereas the MBR approach respectively MBRQT_1 require 50.9% respectively 34.5% less (see Table 49 on page 334).

Table 83: kNN query results for the summary-like R-trees in the T_25 scenario.

	MBR	MBRQT_1	MBRQT_2	MBRQT_3	MBRQT_4	MBRQT_5	MBRQT_6	QTMBR_1	QTMBR_2	QTMBR_3	QTMBR_4
k=10											
#Dist2Rect	1,074.4	1,296.9	2,527.6	2,662.3	2,520.8	3,722.0	4,850.0	5,266.7	1,645.8	4,719.3	10,406.5
#Dist2Sum	1,074.4	926.9	1,181.7	1,107.4	937.2	935.3	1,071.1	5,141.6	1,538.0	4,191.0	2,621.4
#PA total	13.6	16.3	19.7	19.3	16.3	15.9	18.0	204.7	22.3	55.0	52.8
#PA IN	8.74	11.02	13.53	13.52	11.13	11.04	12.48	33.54	11.35	40.28	44.78
#PA LN	4.89	5.27	6.14	5.75	5.20	4.84	5.53	171.18	10.92	14.68	8.05
k=1,000											
#Dist2Rect	1,340.8	1,613.7	3,259.3	3,635.1	2,900.6	4,679.6	6,315.5	5,943.2	1,948.8	5,355.2	12,776.3
#Dist2Sum	1,340.8	1,181.6	1,474.3	1,393.6	1,201.8	1,203.7	1,357.3	5,815.7	1,839.7	4,765.0	3,078.7
#PA total	86.1	90.4	95.4	94.7	91.2	89.7	93.7	424.3	106.8	148.0	147.0
#PA IN	11.01	13.86	16.82	16.78	14.05	14.05	15.66	37.93	13.86	46.15	54.97
#PA LN	75.11	76.57	78.58	77.96	77.13	75.64	78.00	386.33	92.95	101.83	92.07

357). Additionally, the MBR R-tree still has a substantially greater fanout than the MBRQT^{c,a} R-trees (116.5 as opposed to roughly 87.8) and significantly less internal nodes, in general. Thus, the evolution of these results is easily comprehensible. Varying R-tree heights also play a role when comparing the QTMBR$^{b}_{c,a}$ parameterizations with each other: QTMBR_4 is now the worst technique with regard to the internal node accesses (both out of the QTMBR$^{b}_{c,a}$ parameterizations as well as overall). Again, it is because the QTMBR_4 R-tree has a height of 5 whereas the R-trees of all other QTMBR$^{b}_{c,a}$ parameterizations feature a height of 4. Furthermore, the fanout of the QTMBR_4 R-tree is the lowest by far (only 47.7, while being at least 100.7 for the other QTMBR$^{b}_{c,a}$ parameterizations) and even significantly lower than for the summary-like QTMBR_4 R-tree in the T_5 scenario (where it is 61.3).
Aside from this, the results show more of the already known patterns:

— The spatially more accurate the summaries of the diverse MBRQT^{c,a} parameterizations' should be (from a theoretical point of view), the worse their page access performances tend to be. The inferiority of the theoretically more accurate techniques is a bit greater now than in the T_5 scenario's summary-like R-trees. For example, in the T_25 scenario, MBRQT_1 requires only 16.3 total page accesses whereas MBRQT_3 requires 19.3 accesses. In the T_5 scenario, the numbers were 20.2 (MBRQT_1) and 21.1 (MBRQT_3). This is in line with the distribution of the spatial scatterings of the leaf nodes' data points where the differences between the techniques increased from the T_5 scenario (see Figure 109 on page 356) to the T_25 scenario (see Figure 110 on page 361).
— QTMBR_2 is the best QTMBR$^{b}_{c,a}$ parameterization with regard to the internal node accesses. QTMBR_4 is the best QTMBR$^{b}_{c,a}$ parameterization

with regard to the leaf node accesses. Overall, QTMBR_2 is now best in its peer group *by far* (additionally amplified by the greater height of the QTMBR_4 R-tree). However, it is still significantly worse than the MBR approach and all MBRQT^{c,a} parameterizations.

— QTMBR_3 exhibits a worse leaf node access performance than QTMBR_2 (14.7 as opposed to 10.9) despite a—in theory—greater spatial accuracy of its leaf node summaries (6,755.6 dsu^2 cumulated indexed surface area at leaf node level as opposed to 25,338.6 dsu^2, see Table 63 on page 359) and a greater leaf node memory utilization rate (67.2% as opposed to 65.1%, see Table 61 on page 357). Furthermore, its internal node access performance is only 10% better than for QTMBR_4 even though its R-tree has a 25% less *internal node levels* (3 as opposed to 4), a significantly greater fanout (100.7 as opposed to 47.7), and a lower number of internal nodes (15,960 as opposed to 33,500). Hence, within its peer group, the infeasibility of QTMBR_3 (respectively the combination of its concrete parameter values) still seems to be especially pronounced for summary-like R-trees.

— In comparison to their MBR-like R-trees in the T_25 scenario, the total amounts of leaf node accesses are significantly increased for all MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ parameterizations. Anew, this indicates the general degeneration of summary-like MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ R-trees.

For $k = 1,000$, $r = 10$, and $r = 1,000$, the MBR approach is also clearly the best performing technique. All variations in the results of the remaining techniques are within the expectations. Therefore, these specific results are not further elaborated. On a side note, the MBR approach has the best leaf node access performances for $k = 1,000$ and $r = 10$ whereas for $r = 1,000$, MBRQT_5 is minimally better.

In total, it turns out that the MBR approach is unmatched for the summary-like R-trees and the T collection (embracing both the T_5 scenario and the T_25 scenario). This is *always* the case for internal node accesses, too. Although the MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ parameterizations are capable of reducing the cumulated indexed surface areas at leaf node level by a significant magnitude (see Table 59 on page 352 and Table 63 on page 359), also their *leaf node* access performances generally cannot improve that of the MBR approach (with extremely rare exceptions).

In the following, the main reasons for this are outlined, once again, as a conclusion of the evaluation of the summary-like R-trees in the T collection's scenarios: For one thing, it is due to the misassignment of data points to leaf nodes in the summary-like MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ R-trees which leads to those R-trees' degeneration. Consequently, the relevant data points are usually scattered across more leaf nodes. A second aspect is that the cumulated indexed surface areas per node exhibit a pronounced long tail distribution, i.e. the majority of the reductions are achieved for very few nodes with very large indexed surface areas—which

has no relevance for the majority of the nodes. This is the case for both the T_5 scenario (see Figure 106 on page 353) *and* the T_25 scenario (see Figure 124)—despite the latter scenario's greater spatial spread of the leaf nodes' data points (because of the increased leaf node capacity of 25). Due to the varying assignment of data points to leaf nodes in the different summary-like R-trees, the nodes at the same ranks for the distributions of the indexed surface area per node are not comparable, anyway. A further aspect is that in section 13.3.1, it has been shown that regardless of this, the indexed surface areas are not linearly transferable into node access performances, after all. Overall, this means that indexed surface areas are unsuitable indexing performance indicators for summary-like R-trees (at least for erratic data). Therefore, the spatial scatterings of the leaf nodes' data points have to be considered which are generally unfavorable for the MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ parameterizations—and simultaneously prove the misassignment of data points to leaf nodes for the summary-like MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ R-trees (in varying extents). In combination, all these aspects lead to the ermerging results for the leaf node access performances. For the *internal node access performances* of the *summary-like* R-trees, in addition, the diverging fanouts, the differing absolute amounts of internal nodes, and the different heights (only in the T_25 scenario) of the respective R-trees are of relevance—which are almost always in favor of the MBR approach. Also, the cumulated indexed surface areas reveal that the MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summaries' ability to reduce the indexed surface areas at the internal nodes' levels is far less pronounced. For the QTMBR$^{b}_{c,a}$ summaries, this is in large parts attributable to the restrictions imposed by target depth td for the summary-from-summaries calculation. For the MBRQT^{c,a} parameterizations, this restriction is generally not as severe because they feature rather low values for parameter c anyway.

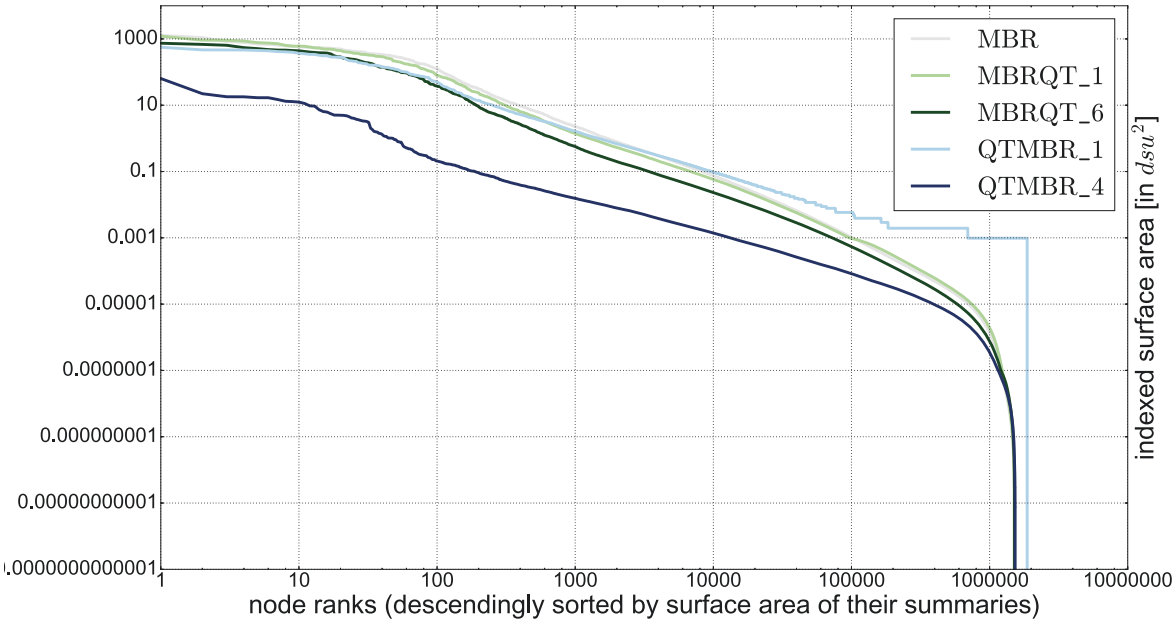


Fig. 124: Leaf nodes sorted in descending order by their indexed surface areas (summary-like R-trees in the T_25 scenario). Both axes are log-scaled.

13.3.3. RESULTS FOR THE R_5 SCENARIO. Since the T collection's scenarios have been entirely evaluated, we now commence with the evaluation of the R collection's scenarios. Hereby, we start anew with the assessment of the scenario featuring a low leaf node capacity: the R_5 scenario. As usual, the query results of the MBR-like R-trees are examined first before proceeding to the query results for the summary-like R-trees.

Query Results for the MBR-like R-trees in the R_5 scenario

Table 84 depicts the kNN query results for the MBR-like R-trees in the R_5 scenario. As usual, we start with considering the results for $k = 10$. The MBR approach now requires 10.5 page accesses in total from which 5.14 are to internal nodes while 5.33 are to leaf nodes.

Table 84: kNN query results for the MBR-like R-trees in the R_5 scenario.

	MBR	MBRQT_1	MBRQT_2	MBRQT_3	MBRQT_4	MBRQT_5	MBRQT_6	QTMBR_1	QTMBR_2	QTMBR_3	QTMBR_4
k=10											
#Dist2Rect	517.2	880.7	969.6	970.0	985.7	1,084.5	1,085.3	714.0	599.6	846.0	1,459.9
#Dist2Sum	517.2	517.2	517.2	517.2	516.3	516.3	516.3	630.7	522.3	623.2	567.6
#PA total	10.5	10.3	10.3	10.2	10.1	10.1	10.1	13.1	10.6	11.2	10.3
#PA IN	5.14	5.14	5.14	5.14	5.14	5.14	5.14	6.11	5.19	6.05	5.57
#PA LN	5.33	5.12	5.11	5.09	4.97	4.93	4.92	6.97	5.40	5.16	4.68
k=1,000											
#Dist2Rect	1,215.1	2,662.3	3,025.8	3,030.0	3,042.7	3,449.9	3,453.8	1,505.6	1,313.4	1,945.9	3,987.6
#Dist2Sum	1,215.1	1,215.1	1,215.1	1,215.1	1,215.1	1,215.1	1,215.1	1,409.4	1,225.3	1,397.6	1,303.5
#PA total	307.2	305.6	305.4	305.4	304.5	304.2	304.2	321.9	308.1	308.1	303.5
#PA IN	11.04	11.04	11.04	11.04	11.04	11.04	11.04	12.70	11.13	12.59	11.78
#PA LN	296.17	294.59	294.33	294.35	293.51	293.13	293.12	309.18	296.95	295.51	291.77

In a comparison of the MBR approach's results to those of the MBRQT^{c,a} parameterizations, the familiar picture (with respect to the MBR-like R-trees) shows: The MBRQT^{c,a} parameterizations slightly improve the MBR approach, overall. Hereby, it is noteworthy that the page access reductions are exclusively achieved at leaf node level, and no improvements arise for the internal node accesses. This is easily comprehensible: Since a lot of data points are administered in the subtrees of the *internal nodes*[322] and a regular spatial distribution of the data points is present, the respective maximum cell numbers of the refining quadtrees ($c = 8$ for MBRQT_1 to MBRQT_3, $c = 16$ for MBRQT_4 to MBRQT_6) are too low to exclude

[322]As there are 58,468 level-3-nodes, the average amount of data points in a level-3-nodes' subtree is $25,000,000/58,468 = 427.6$ data points. With the information listed in Table 52 (page 338), this amount could also have been *estimated*: Since the fanout of the basic R-tree is 117.3 and the leaf node memory utilization rate is 72.9%, the estimated average amount of data points administered in the subtree of a level-3-node is $117.3 \cdot (5 \cdot 0.729) = 427.6$ data points (i.e. the same value results).

regions of dead space from indexation. For the leaf node accesses, slight and continuous improvements arise from MBRQT_1 (5.12) to MBRQT_6 (4.92). Thereby, the MBRQT^{16,a} parameterizations are all better than the MBRQT^{8,a} parameterizations. The results show that parameter c is of greater significance for the leaf node accesses of the R_5 scenario's MBR-like R-trees than parameter a. It is also a logical consequence of the techniques' respective parameterizations and the regular spatial distribution of the R collection's data points.[323]

With regard to the QTMBR$^{b}_{c,a}$ parameterizations, QTMBR_1 to QTMBR_3 are overall still worse than the MBR approach but the deficiencies are much smaller now. QTMBR_4 is even slightly better with regard to the total amount of page accesses (10.3 as opposed to 10.5 for the MBR approach). This is in particular because the QTMBR$^{b}_{c,a}$ parameterizations are generally much more competitive in terms of internal node accesses now: In comparison to the MBR approach, the deteriorations are in the range between 1.0% (for QTMBR_2) to 18.9% (for QTMBR_1)—which is much less than for the T_5 scenario's MBR-like R-trees (where they have been 33.2% *at least*). It is obvious that because there are no extreme agglomerations of data points in the R collection, at the internal nodes' levels, the disadvantages of the QTMBR$^{b}_{c,a}$ parameterizations ($\rightarrow$ size of the smallest indexable spatial units and the mere approximation of the full-precision MBRs, both due to the specification of the summary-from-summaries calculation process) are not as severe anymore. Nevertheless, they are yet considerable at level 3 (see Table 85). Still, with regard to the overall results and in comparison to the MBR approach, QTMBR_4 is capable of compensating these deficiencies with its superior leaf node access performance. Here, as for all MBR-like R-trees, QTMBR_4 is the best technique and requires only 4.68 leaf node accesses on average. It is an improvement of 12.2% over the MBR approach respectively 4.9% over MBRQT_6. Notably, for the first time, also QTMBR_3 (5.16 leaf node accesses on average) is better with regard to the amount of leaf node accesses than the MBR approach (5.33). This is line with the significant increase in the number of indexed areas for QTMBR_3 which has been noticed in the structural analysis of the R_5 scenario's MBR-like R-trees. The reason is that for QTMBR_3, the number of quadtree cells c does not deplete as often prematurely anymore respectively the threshold surface area a is undercut less frequently before more black cells can be built in the summaries' basic quadtrees (due to the greater spatial spread of the data points stored in the leaf nodes).

In the following, we compare the results of the given *R_5 scenario*'s MBR-like R-trees to those of the *T_5 scenario*'s MBR-like R-trees. In general, the amounts of page accesses are significantly smaller in the R_5 scenario. Consider the MBR approach as an example: The total amount of page accesses is now 10.5 instead of 18.3. This underlines that a regular data point distri-

[323] In the T_5 scenario with its erratic data point distribution, it is parameter a which is more important for the MBRQT^{c,a} parameterizations' leaf node access results.

Table 85: Average amount of internal node accesses for the MBR-like R-trees in the R_5 scenario. The internal node accesses are displayed level-wise.

	MBR	MBRQT_1	MBRQT_2	MBRQT_3	MBRQT_4	MBRQT_5	MBRQT_6	QTMBR_1	QTMBR_2	QTMBR_3	QTMBR_4
level 0	1	1	1	1	1	1	1	1	1	1	1
level 1	1.06	1.06	1.06	1.06	1.06	1.06	1.06	1.05	1.05	1.05	1.05
level 2	1.27	1.27	1.27	1.27	1.27	1.27	1.27	1.34	1.27	1.34	1.30
level 3	1.82	1.82	1.82	1.82	1.82	1.82	1.82	2.72	1.87	2.66	2.23

bution is per se an easier retrieval situation than an erratic one—extreme agglomerations of data points (such as they occur in specific data space regions for the T collection) are much harder to deal with. It is noteworthy that the page accesses primarily decrease because there are much less internal node accesses now:[324] For example, for the MBR approach, their *absolute* number diminishes from 12.51 in the T_5 scenario to 5.14 in the R_5 scenario. In *relative* numbers, internal node accesses are about 58% fewer for the MBR approach *and* all MBRQT^{c,a} parameterizations. At leaf node level, the decrease is only between 7.6% (MBRQT_6, corresponding to an absolute amount of 0.40 accesses) and 13.5% (MBRQT_4, 0.77 accesses). With respect to the internal node accesses, neither fanout (117.3 in the R_5 scenario as opposed to 116.7 in the T_5 scenario), nor the number of internal nodes (58,961 as opposed to 66,169, i.e. a decrease of 10.9% in the R_5 scenario), or the height (5 in both scenarios) of the respective basic R-trees are eligible as explanations that the internal node accesses decrease by roughly 58%. Hence, it must be related to the summaries. Comparing the respective internal node accesses per level (see Table 85 for the R_5 scenario and Table 74 on page 374 for the T_5 scenario), it can be seen that for all techniques, there are significant differences at all levels. For example, in the R_5 scenario, the MBR approach requires 1.06 accesses at level 1, 1.27 accesses at level 2, and 1.82 accesses at level 3. In the T_5 scenario, it is 2.01, 4.62, and 4.88 accesses. In Figure 125, the MBR summaries of the nodes at level 1, level 2, and level 3 of the R_5 scenario's basic R-tree (left) and the T_5 scenario's basic R-tree (right) are juxtaposed. It is obvious that at all levels, there is significantly more overlap between the MBR summaries of the latter's nodes. Even though the R_5 scenario's basic R-tree is not completely free from overlap, it is evident that the ap-

[324] Note that the QTMBR$^b_{c,a}$ parameterizations are excluded from the consideration of the results' developments between the T_5 scenario and the R_5 scenario. This is because in the previous evaluations, diverse problems were encountered for the QTMBR$^b_{c,a}$ parameterizations with regard to both internal node accesses as well as leaf node accesses (whose origin always were the extremely densely populated data regions which occur for the T collection).

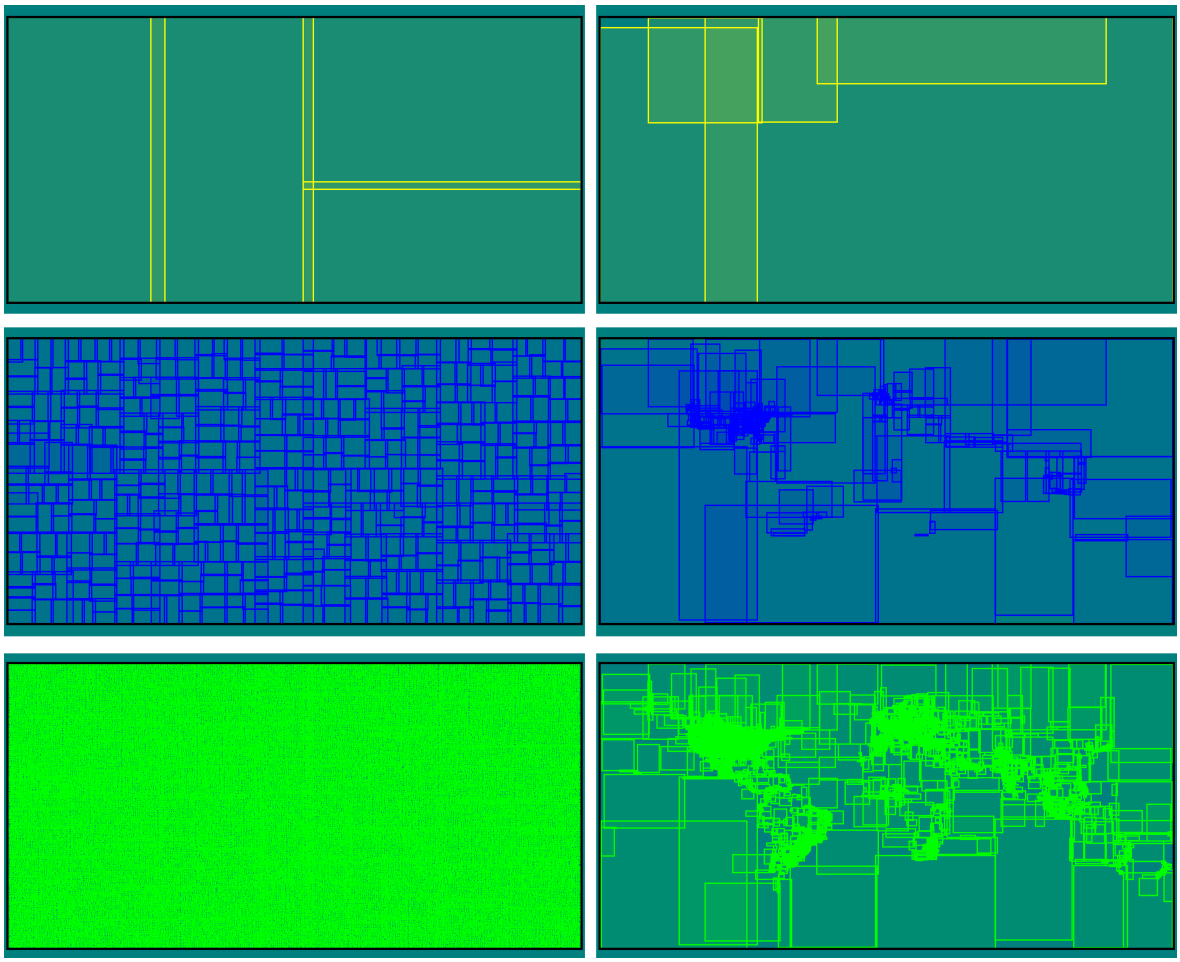


Fig. 125: Comparison of the basic MBR-like R-tree in the R_5 scenario (left) respectively in the T_5 scenario (right). For both R-trees, the MBR summaries of the nodes at level 1 (top), level 2 (middle), and level 3 (bottom) are displayed.

plied SELECTBEST(.)-method and the R*-split work very well with regard to keeping the nodes' data point clouds cleanly separated.
Of course, this is massively favored by the regular spatial distribution of the R collection's data points which is much easier to handle than the erratic spatial distribution of the T collection's data points. Although it can be attested that even for the T collection, the SELECTBEST(.)-method and the R*-split work satisfactorily well (at least when MBR summaries are used in the construction phase), it is self-evident that over time, the R-trees degenerate to a certain extent if no reinsertions or similar operations are applied. Therefore, it is easily comprehensible why the internal node accesses are so much lower in the R_5 scenario: There is simply much less overlap between the internal nodes' MBRs. The numbers in Table 85 show that at least at level 1 and at level 2, only one node is accessed in the majority of the cases. In contrast, in the T_5 scenario, already more than two level-1-node accesses are conducted on average (see Table 74 on page 374)—which is plausible when considering the top right image of Figure 125.

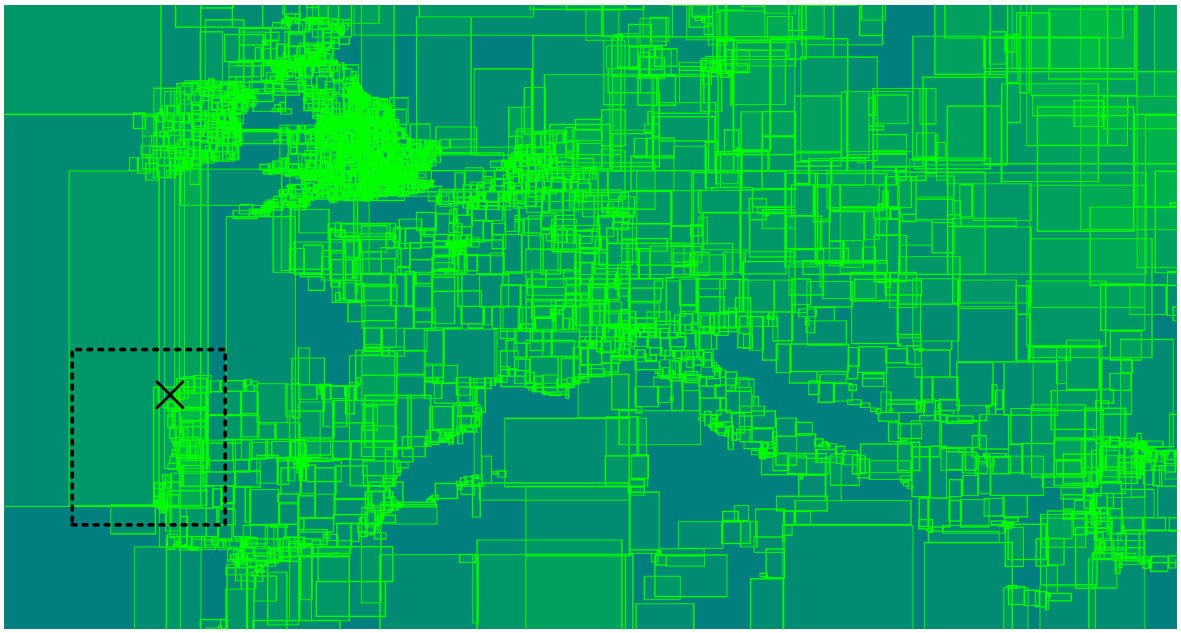


Fig. 126: Visualization of the MBR summaries of a set of level-3-nodes in the T_5 scenario's MBR-like R-tree. It is easily recognizable that the visualization is zoomed into the data space region where Europe is located. At the location marked with the black cross, the MBRs of four level-3-nodes overlap. The dashed, black rectangle indicates the snippet region of Figure 127.

For the MBR approach and the MBRQT^{c,a} parameterizations, the decreases in the amount of leaf node accesses from the T_5 scenario to the R_5 scenario are comparatively low. For example, the MBR approach requires 5.83 leaf node accesses in the T_5 scenario which decrease to 5.33 accesses in the R_5 scenario. This is because also in the *T_5 scenario*'s basic R-tree, the MBR summaries of the leaf nodes are largely free from overlap—in spite of the substantial amounts of overlap at the internal nodes' levels. See Figure 126, Figure 127, and Figure 128 for example visualizations illustrating this. In Figure 126, we select a specific location in the data space at which four MBR summaries of level-3-nodes in the *T_5 scenario*'s basic R-tree overlap. Level 3 is the level above leaf node level. The four level-3-nodes are separately depicted once again in the left image of Figure 127. In the right image of Figure 127, the MBR summaries of the *leaf nodes* which are associated with the four level-3-nodes are displayed. Already in this image, it is evident that despite the huge overlap between the level-3-nodes' MBRs, the leaf nodes' MBRs are largely free from overlap. Finally, the *right* image of Figure 128 presents a detailed depiction of the region in which *all* MBRs of the four level-3-nodes actually overlap. Here, it shows anew that the mutual overlaps between the leaf nodes' MBRs are surprisingly low considering the constellation of the four level-3-nodes. In the *left* image of Figure 128, the MBR summaries of some leaf nodes in the currently assessed *R_5 scenario*'s basic R-tree are displayed. When comparing both images of Figure 128, it is evident that the leaf nodes' MBRs in

the R_5 scenario's basic R-tree generally have much more even sizes. They also feature less mutual overlap. On the other hand, concerning the mutual overlaps, it is not as much of a difference as one might have suspected on basis of Figure 125—the leaf nodes' MBRs in both images are fairly well separated. As outlined before, the mutual overlap between the leaf nodes' MBR summaries of the T_5 scenario's basic R-tree is 1,808.2 dsu^2 for a cumulated indexed surface area of 18,751.4 dsu^2 at leaf node level and 7.65 million leaf nodes. For the MBR summaries of the R_5 scenario's basic R-tree's leaf nodes, the mutual overlap is 1,486.5 dsu^2 given a cumulated indexed surface area of 21,706.2 dsu^2 for 6.91 million leaf nodes. We already mentioned in the evaluation of the query results for the T_5 scenario's MBR-like R-tree that the freedom from mutual overlap is the most important asset in spatial indexing. When comparing the cumulated indexed surface areas for both scenarios (see Table 55 on page 341 for the R_5 scenario and Table 46 on page 324 for the T_5 scenario), it is apparent that the R_5 scenario's basic R-tree has greater cumulated indexed surface areas at all levels except level 1 even though its amount of nodes per level is always lower than in the T_5 scenario's basic R-tree.[325] Nevertheless, the page accesses are significantly fewer in the R_5 scenario which proves that not the surface area which is indexed in total is decisive but the freedom from overlap. Since the T_5 scenario's basic R-tree is evidently not that much worse than the R_5 scenario's basic R-tree with regard to mutual overlap of leaf node summaries, it is comprehensible that the decrease in the amount of leaf node accesses is not as pronounced as for the internal nodes.

Concerning the query results for $k = 1,000$, there is not much change in the outcomes. In general, the QTMBR$^b_{c,a}$ parameterizations are slightly more competitive. Hereby, QTMBR_1 to QTMBR_3 are still marginally worse than the MBR approach and the MBRQT^{c,a} parameterizations, overall. However, QTMBR_4 is the best technique *of all* as it requires 0.7 fewer page accesses than MBRQT_6. In total, the results for $k = 1,000$ are all very close, being between 303.5 (QTMBR_4) and 308.1 (QTMBR_2 and QTMBR_3) for the total amount of page accesses. Only QTMBR_1 is a little behind with 321.9 total page accesses (which is still within 6.1% of QTMBR_4).

With regard to the range queries, due to the regular spatial distribution of the data points, the result set sizes for $r = 10$ (with an average amount of 10.26 data points in its result sets) and $r = 1,000$ (990.76) are very close to the forecasts of the target query radii. As a consequence, the range

[325] The concrete numbers of nodes are as follows:

— R_5 scenario → (level 0: 1), level 1: 4, level 2: 488, level 3: 58,468, leaf node level: 6,855,863, and
— T_5 scenario → (level 0: 1), level 1: 5, level 2: 562, level 3: 65,601, leaf node level: 7,654,433.

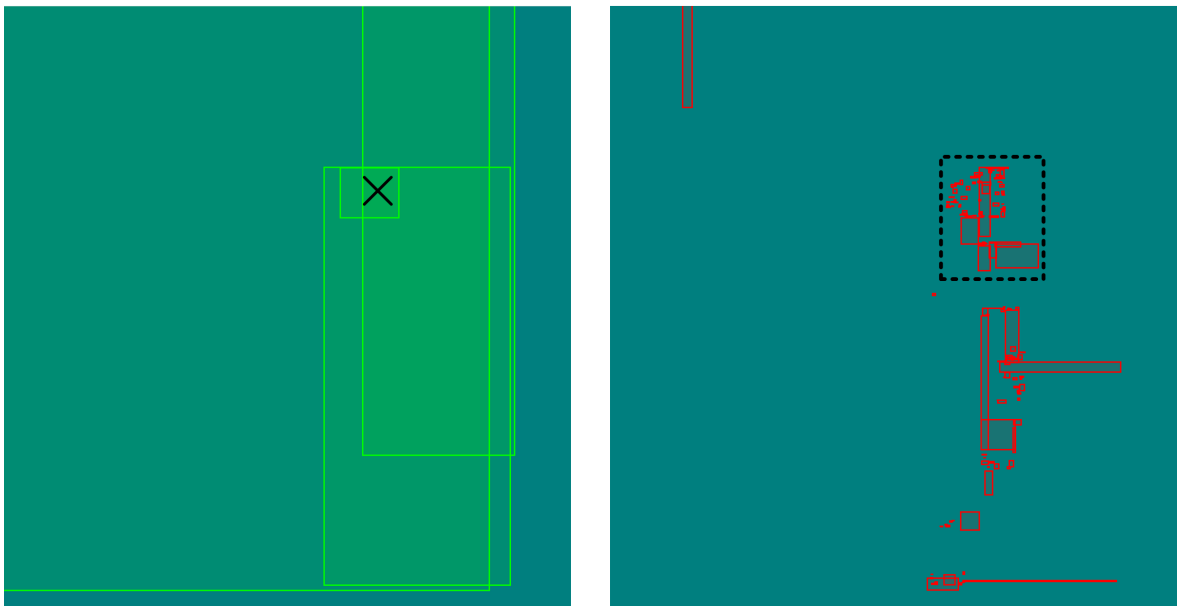


Fig. 127: The left image displays the separate, detailed depiction of the four level-3-nodes whose MBR summaries overlap at the location specified in Figure 126. At the right, their associated leaf nodes' MBR summaries are visualized. The dashed, black rectangle in the right image indicates the snippet region of Figure 128's right image.

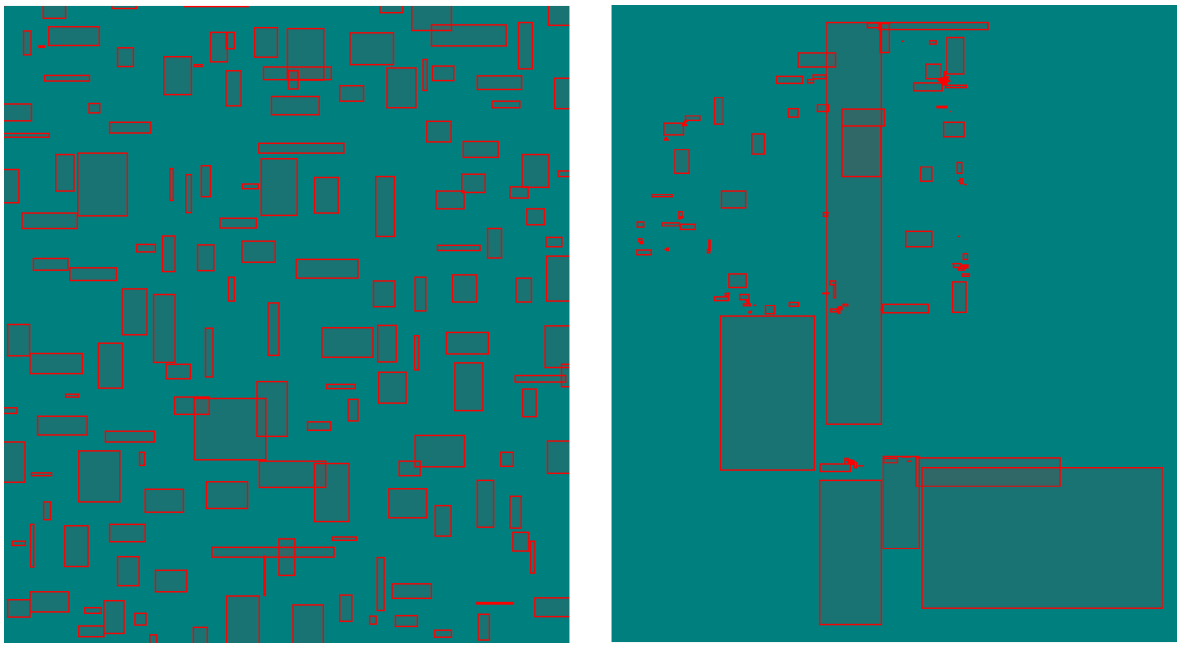


Fig. 128: Comparison of leaf node summaries in the MBR-like MBR R-tree in the R_5 scenario (left) and the MBR-like MBR R-tree in the T_5 scenario (right). The right image originates from the snippet region outlined in Figure 127. The left image is from a random data space region. Both images have been captured at the same zoom level of the R-tree visualization tool.

query results for $r = 10$ respectively $r = 1,000$ are almost identical to the kNN query results for $k = 10$ respectively $k = 1,000$. For e.g. the MBR approach and $r = 10$, it is 10.3 page accesses on average from which 5.16

are to internal nodes while 5.09 are to leaf nodes—as opposed to 10.5, 5.14, and 5.33 accesses for $k = 10$. Due to the similarity of the kNN and range query results, we refrain from assessing the range query results for both the R_5 scenario as well as the subsequent R_25 scenario unless anything unusual is observed.

At this point, we want to revisit two assertions made in the previous analyses. In the structural analysis of the MBR-like R-trees in the R_5 scenario (section 13.2.1), the chances of improving the MBR approach's total page access performance in the R_5 scenario's MBR-like R-trees as well as in the T_5 scenario's MBR-like R-trees were estimated against each other. We concluded that the chances are better in the R_5 scenario because there, the share of leaf nodes whose indexed surface area is reduced by use of our MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summaries is generally substantially larger than in the T_5 scenario.[326] The query results show that *in some sense*, this prediction might eventually apply to the QTMBR$^{b}_{c,a}$ parameterizations: In the R_5 scenario, their total page access perfomances are not as inferior to those of the other techniques anymore as they were in the T_5 scenario, and QTMBR_4 is now even better than the MBR approach. However, for the MBRQT^{c,a} parameterizations, the *relative* improvements over the MBR approach are lower in the R_5 scenario than in the T_5 scenario. For instance, MBRQT_6 improves the MBR approach by 3.8% in the R_5 scenario but by 5.5% in the T_5 scenario (for $k = 10$). This is because of the general convergence of the results in the R_5 scenario. As just outlined, the convergence is caused by the freedom of overlap between the nodes' summaries and the absence of extremely densely populated regions in the data space. Therefore, the results of our summarization approaches are not as favorable as the distributions of the indexed surface area per node suggest. For example, in the R_5 scenario's MBR-like R-tree, 6,855,242 of 6,855,863 leaf nodes (i.e. more than 99.99%) have a smaller indexed surface area with MBRQT_6 summaries than with MBR summaries. However, for the amount of leaf node accesses, the difference in the 10NN results is solely 0.41 accesses or 7.7% improvement for MBRQT_6. In the T_5 scenario's MBR-like R-trees, only 4,556,761 of 7,654,433 leaf nodes (i.e. only 59.5%) have a smaller indexed surface area with MBRQT_6 summaries. Nevertheless, MBRQT_6 improves the MBR approach by 8.7% with regard to the leaf node accesses. In total, this shows that for the MBR-like R-trees, not only the properties of the nodes' summaries are decisive for the *relative* results between the techniques but also the properties of the underlying data collection.[327] This is a fairly obvious insight which may nonetheless sometimes fall into oblivion.

[326] With regard to the internal nodes, it was implicitly estimated that the chances of improving the MBR approach are equally low in both the R_5 as well as the T_5 scenario. This is because the cumulated indexed surface areas were marginally improved *at best* in both scenarios.

[327] For the summary-like R-trees, also the diverging properties of the techniques' respective R-tree structures are decisive, obviously.

In particular, it is evident that *comparative* predictions for different data collections (such as the one currently being revisited) can hardly be made in a meaningful way as there are too many influencing factors. As a consequence, we refrain from revisiting suchlike predictions in the remaining analyses.
The second assertion was made in the query result analysis of the T_5 scenario's MBR-like R-trees (section 13.3.1). There, it was evident that with regard to the internal node accesses, the $\text{QTMBR}^{b}_{c,a}$ parameterizations were very competitive at level 1 and level 2 but dramatically fell off at level 3 (also see Table 73 on page 373). These phenomena were attributed to

a) the T collection's erratic data point distribution (which from time to time allowed the $\text{QTMBR}^{b}_{c,a}$ parameterizations to exclude dead space from the level-1- and level-2-nodes summaries), and
b) the mere approximation of the corresponding full-precision MBRs by level-3-node $\text{QTMBR}^{b}_{c,a}$ summaries (due to the restrictions for the smallest indexable spatial units introduced by target depth td which led to poor performance in regions of very high data point density).

We concluded that therefore, these phenomena might change for the R_5 scenario's MBR-like R-trees in such way that the results converge, i.e. that the $\text{QTMBR}^{b}_{c,a}$ parameterizations are less competitive at level 1 and level 2 but simultaneously, they are closer to the MBR approach and the $\text{MBRQT}^{c,a}$ parameterizations at level 3. When reassessing the R_5 scenario's results listed in Table 85 (page 410), these presumptions are largely confirmed: Generally, the $\text{QTMBR}^{b}_{c,a}$ parameterizations now only safe 0.01 node accesses at level 1 but are also much closer to the MBR approach and the $\text{MBRQT}^{c,a}$ parameterizations for the level-3-node accesses. For example, QTMBR_1 now requires only 1.0 (or 49%) *more* level-3-node accesses than the MBR approach. In the T_5 scenario, the difference between both was 99.57 level-3-node accesses, corresponding to a *surplus* of 2,040% for QTMBR_1. At level 2, it is a bit mixed as QTMBR_2 is now exactly as good as the MBR approach and the $\text{MBRQT}^{c,a}$ parameterizations. QTMBR_2 was slightly better in the T_5 scenario, i.e. it is less competitive now. In contrast, the remaining $\text{QTMBR}^{b}_{c,a}$ parameterizations are only between 0.03 page accesses (or 2.4%, QTMBR_4) and 0.07 page accesses (or 5.5%, QTMBR_1) worse than the MBR approach. In the T_5 scenario, they were at a disadvantage of *at least* 0.77 page accesses (which corresponds to 16.7%). Thus, QTMBR_1, QTMBR_3, and QTMBR_4 are more competitive now. Overall, the second assertion can therefore be confirmed in large parts. The reasons for the observed developments are that on the one hand, for the level-1- and level-2-nodes, the $\text{QTMBR}^{b}_{c,a}$ summaries can hardly exclude any dead space from indexation anymore (due to the regular data point distribution). On the other hand, as already outlined, due to the non-existence of extreme data point agglomerations and the consequential greater spatial spread

of the data points administered in the level-3-nodes' subtrees, especially the deficiencies with regard to the smallest indexable spatial units for the level-3-nodes' summaries are not as severe anymore.
At this point, we conclude the evaluation of the query results for the R_5 scenario's MBR-like R-trees and continue with assessing the query results of the corresponding summary-like R-trees.

Query Results for the Summary-like R-trees in the R_5 scenario

Table 86 lists the kNN query results for the R_5 scenario's summary-like R-trees. Again, we first assess the results for $k = 10$. The MBR approach requires a total amount of 10.3 page accesses on average which can be split into 5.13 internal node accesses and 5.16 leaf node accesses.[328]

Table 86: kNN query results for the summary-like R-trees in the R_5 scenario.

	MBR	MBRQT_1	MBRQT_2	MBRQT_3	MBRQT_4	MBRQT_5	MBRQT_6	QTMBR_1	QTMBR_2	QTMBR_3	QTMBR_4
k=10											
#Dist2Rect	515.5	701.3	758.8	727.2	782.1	831.8	848.4	920.9	766.9	985.2	1,241.0
#Dist2Sum	515.5	419.0	413.3	390.5	400.3	395.0	394.5	820.0	651.0	725.1	590.7
#PA total	10.3	10.2	10.4	10.1	10.1	10.0	10.2	12.9	10.6	11.4	10.7
#PA IN	5.13	5.25	5.35	5.16	5.28	5.16	5.26	5.88	5.28	6.06	5.95
#PA LN	5.16	4.97	5.06	4.95	4.82	4.82	4.94	7.03	5.32	5.30	4.74
k=1,000											
#Dist2Rect	1,235.3	2,433.8	2,799.4	2,678.7	2,814.7	3,131.6	3,115.1	1,900.3	1,539.2	2,168.9	3,646.9
#Dist2Sum	1,235.3	1,108.3	1,115.9	1,056.1	1,098.5	1,075.8	1,070.6	1,784.5	1,411.6	1,542.8	1,314.9
#PA total	307.0	306.3	307.5	306.5	306.2	304.3	305.0	323.0	309.0	309.9	309.2
#PA IN	11.13	12.96	13.16	12.61	13.08	12.70	12.78	11.21	11.57	13.63	16.38
#PA LN	295.89	293.31	294.36	293.83	293.08	291.58	292.23	311.79	297.43	296.23	292.86

The MBRQT^{c,a} parameterizations marginally improve the MBR approach. The only exception is MBRQT_2 which has 10.4 total page accesses, i.e. 0.1 page accesses more than the MBR approach. MBRQT_5 is the best technique overall, featuring 10.0 page accesses (5.16 to internal nodes, 4.82 to leaf nodes). This is remarkable since for the summary-like R-trees analyzed so far, the MBR approach is unmatched. In general, the internal node access performance of the MBRQT^{c,a} parameterizations is now a little worse than that of the MBR approach. Due to the general prerequisits, it is no surprise, though: The MBRQT^{c,a} R-trees have the same height (5) but a 25% lower fanout and 33% more internal nodes compared to the MBR approach's R-tree. Simultaneously, the regular spatial distribution of the data points prevents the exclusion of dead space from the internal nodes' MBRQT^{c,a} summaries. Hence, the advantageousness of the MBR approach

[328]With these results, the R_5 scenario's *summary-like* MBR R-tree is slightly better than the R_5 scenario's *MBR-like* MBR R-tree which requires 10.5 page accesses.

for the internal node accesses is logical. Nevertheless, the deficiencies of the MBRQT^{c,a} parameterizations are only minor and can be compensated by their superior leaf node access performances—which is another new observation: For the T_5 scenario's summary-like R-trees, only MBRQT_5 had a better leaf node access performance than the MBR approach. For the currently assessed R_5 scenario's summary-like R-trees, all MBRQT^{c,a} parameterizations are *clearly* better than the MBR approach. We think this is because the general inappropriateness of the SELECTBEST(.)-method is strongly mitigated by the R collection's regular data point distribution: Assuming a regular spatial distribution of the data points, situations as depicted in Figure 119 (page 389) should arise less frequently than with erratic data, especially at upper-level nodes. Additionally, Figure 112 (page 366) depicting the distribution of the spatial scattering of the leaf nodes' data points for the R_5 scenario's summary-like R-trees showed that there are almost no differences between the techniques with regard to this measurement of eventual R-tree degenerations.

However, also for the R_5 scenario's summary-like R-trees, the results *amongst* the MBRQT^{c,a} parameterizations are not always as one would expect them from a theoretical consideration of the respective summaries' spatial accuracy: the theoretically more accurate parameterizations partly exhibit worse results. For example, MBRQT_6 (10.2 page accesses on average) is worse than MBRQT_5 (10.0). Assuming equally suitable R-tree structures, MBRQT_6 summaries should lead to better results than MBRQT_5 summaries, especially at leaf node level.[329] Nevertheless, for the corresponding *summary-like* R-trees, MBRQT_6 is slightly but clearly worse, even at leaf node level (4.94 leaf node accesses as opposed to 4.82 for MBRQT_5). Since the summary-like R-trees of the MBRQT^{c,a} parameterizations are basically identical with regard to fanout, number of internal nodes, and height (see Table 66 on page 362), the deficiencies of MBRQT_6 have to be related to the summaries respectively the assignment of data points to nodes (which is based on the respective summaries). Of course, in individual cases, it can also be simply due to randomness because the absolute differences are very small. However, these observations recur (see e.g. MBRQT_4 as opposed to MBRQT_6, or MBRQT_1 as opposed to MBRQT_2). Therefore, on basis of the intra-approach results of the MBRQT^{c,a} parameterizations, we conclude that despite its mitigated significance, the inappropriateness of the SELECTBEST(.)-method still has *slight* adverse effects for regular data point distributions. However, at least for the MBRQT^{c,a} parameterizations, these effects cannot be very pronounced as the overall results of the R_5 scenario's summary-like MBRQT^{c,a} R-trees are almost identical to those of the R_5 scenario's MBR-like MBRQT^{c,a} R-trees. A

[329] This has been the case for the R_5 scenario's *MBR-like* R-trees, i.e. where the structures of all R-trees are identical. See Table 84 on page 408. There, the amounts of leaf node accesses are 4.92 for MBRQT_6 and 4.93 for MBRQT_5, i.e. marginally in favor of MBRQT_6 (the internal node access performances are identical for both).

definitive clarification on this issue would require additional experiments from which we refrain at this point.

The $\text{QTMBR}^{b}_{c,a}$ parameterizations are worse than both the MBR approach as well as the $\text{MBRQT}^{c,a}$ parameterizations. In any case, this applies to internal node accesses. The favorable structure of the QTMBR_1 R-tree (fanout 177.2, 39,287 internal nodes) in comparison to the MBR approach's R-tree (117.3, 58,953) cannot compensate its deficiencies concerning the spatial accuracy of the summaries and the possible degeneration of its R-tree. Also for the leaf node accesses, the $\text{QTMBR}^{b}_{c,a}$ parameterizations are generally worse than the MBR approach. The exception is QTMBR_4 which is the best technique *of all* in terms of leaf node accesses. It is the first time QTMBR_4 is prime for *summary-like* R-trees in this regard. That is another query-result-based indication that the severeness of the SELECTBEST(.)-method's unsuitability is mitigated for regular data point distributions. However, within the $\text{QTMBR}^{b}_{c,a}$ parameterizations, QTMBR_2 is the best technique with 10.6 total page accesses on average which is 0.3 page accesses (or 2.9%) worse than the MBR approach.

Interestingly, the R_5 scenario's *summary-like* QTMBR_1 R-tree is slightly better than its *MBR-like* counterpart: Now, only 12.9 total page accesses are required (as opposed to 13.1). This is a little unexpected since so far, the performances of the summary-like $\text{QTMBR}^{b}_{c,a}$ R-trees have always been worse than those of their MBR-like pendants. When comparing the concrete numbers, it is evident that the summary-like QTMBR_1 R-tree is minimally worse for the leaf node accesses (7.03 as opposed to 6.97) but requires only 5.88 internal node accesses in contrast to 6.11 internal node accesses for the MBR-like QTMBR_1 R-tree. This is a decrease of 3.8%, after all. It is attributable to the greater fanout of the summary-like QTMBR_1 R-tree (177.2 as opposed to 117.3, corresponding to an increase of 51%) and the consequential lower amount of internal nodes (39,287 instead of 58,961, i.e. a decrease by 33%).

Another noteworthy observation is that for QTMBR_2 and also all $\text{MBRQT}^{c,a}$ parameterizations except MBRQT_6, the leaf node access performances are slightly better for their *summary-like* R-trees than for their *MBR-like* R-trees. This is very unexpected: For one, the SELECTBEST(.)-method still seems to have slight adverse effects for our summarization approaches (as evident from the intra-approach comparison of the $\text{MBRQT}^{c,a}$ parameterizations' results). Furthermore, the average leaf node memory utilization rates are basically the same for the summary-like and the MBR-like R-trees of QTMBR_2 respectively the $\text{MBRQT}^{c,a}$ parameterizations (i.e. the respective MBR-like R-trees are not at a disadvantage in this regard).[330] However, it has to be recognized that in the R_5 scenario, for all

[330] The average leaf node memory utilization rate is 72.9% for all of the R_5 scenario's MBR-like R-trees (see Table 52 on page 338). For the corresponding summary-like R-trees, it is 72.9% for QTMBR_2 respectively 73.0% for all $\text{MBRQT}^{c,a}$ parameterizations (see Table 66 on page 362).

techniques, the leaf node access performances are generally *very similar* between their MBR-like and their summary-like R-trees: The greatest absolute difference is 0.17 leaf node accesses—which ironically occurs for the MBR approach, i.e. the technique for which the correspondance between its MBR-like and its summary-like R-tree should be greatest. Hence, the techniques' leaf node access results are obviously all within the range of naturally expectable variations. This is a further indication that the adverse effects of the SELECTBEST(.)-method are not very pronounced anymore even though they are vaguely demonstrable on basis of the intra-approach results for the R_5 scenario's summary-like MBRQT^{c,a} R-trees. With regard to the slightly superior leaf node access performance of the summary-like QTMBR_2 R-tree and most of the summary-like MBRQT^{c,a} R-trees compared to their MBR-like counterparts, one aspect should be kept in mind in addition:

The basic R-tree structure (i.e. the assignment of child nodes to parent nodes and of data points to leaf nodes) is the same for all MBR-like R-trees. As outlined, the leaf node access performance of the *MBR-like* MBR R-tree is clearly worse than that of the *summary-like* MBR R-tree (5.33 leaf node accesses as opposed to 5.16, corresponding to a surplus of 3.3% for the MBR-like MBR R-tree). Hence, the R_5 scenario's basic, MBR-like R-tree appears to be at the worse end of the deviations which can be expected within the natural variations for the R-tree construction with MBR summaries. Therefore, the *MBR-like* MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ R-trees might not have the best *basic* R-tree to start with. Consequently, the occassionally occuring better leaf node access performances of the *summary-like* MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ R-trees in comparison to their corresponding *MBR-like* R-trees might be co-created by the evidently suboptimal *basic* R-tree which defines the structure of all the *MBR-like* R-trees.

For $k = 1,000$, the *total page access* results are basically the same as for $k = 10$. With respect to the internal node accesses, QTMBR_1 and QTMBR_2 display better results than all the MBRQT^{c,a} parameterizations. Obviously, this is because the former exhibit a significantly greater fanout and fewer internal nodes in their R-trees. It shows that for greater result set sizes, the significance of the fanout and the general amount of internal nodes in the R-trees (i.e. the general structural properties) is increasing for the internal node access performance: In comparison to the MBRQT^{c,a} parameterizations, the spatial accuracy of the summaries is certainly not in favor of QTMBR_1 and QTMBR_2 while the possibly occuring adverse effects of the inadequate SELECTBEST(.)-method are not very pronounced for all of these techniques. Therefore, their superiority is exclusively attributable to the fanout respectively the number of internal nodes. For the leaf node accesses, QTMBR_4 (292.9 leaf node accesses on average) is slightly outperformed by MBRQT_5 (291.6) and MBRQT_6 (292.2). Since QTMBR_4 is generally the technique which has the best leaf node access performance in the *MBR-like* R-trees but suffers most from the inappro-

priate SELECTBEST(.)-method for the *summary-like* R-trees, we see this as further evidence that also in the R_5 scenario, the SELECTBEST(.)-method's adverse effects are present to low extents. It has also been shown in Figure 112 (page 366) that QTMBR_4 is slightly worst with regard to the spatial scattering of the leaf nodes' data points. As expected, the range query results do not show any noteworthy divergences to the kNN query results. In total, this means that even though the SELECTBEST(.)-method might have marginal adverse effects on the assignment of data points to nodes in the R_5 scenario, the *overall* results are affected to only minimal amounts. This is derivable from comparing the leaf node access performances between the R_5 scenario's MBR-like and summary-like R-trees.[331] Furthermore, the changes in the internal node access performances between the techniques' MBR-like and summary-like R-trees are mostly attributable to differences in the fanout and the amount of internal nodes. In general, the query results in the R_5 scenario are *all* very similar. Regular data point distributions are simply easier to handle than erratic data point distributions and therefore, the results converge for the different techniques. As a consequence, for the R_5 scenario, it is possible to slightly improve the MBR approach even by utilizing *summary-like* R-trees—although there is still *easily* realizable optimization potential left for our summarization approaches. Examples are an adapted SELECTBEST(.)-method or the increase of the memory utilization rates of the MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ *summaries* (i.e. the reduction of the summaries' share of filling data). However, the MBR approach is already very good for itself, too. As an illustration, consider its results for $k = 10$ once again: As there are four levels of internal nodes (level 0 to level 3), the *absolute minimum* of internal node accesses is at least 4 (in case the data points of the query result are all stored in the subtree of a single level-3-node). Hence, the 5.13 internal node accesses of the MBR approach's R-tree appear to be almost optimal with regard to internal node accesses. Furthermore, the mathematically determined amount of considered data points is $5.16 \cdot (5 \cdot 0.729) = 18.8$, i.e. on average, only 8.8 irrelevant data points are assessed before the $k = 10$ nearest neighbors to the query point have been determined. It is therefore a legitimate question whether the small extent of improvement justifies the effort involved.

[331] As already discussed in footnote[306] on page 387, given comparable leaf node memory utilization rates, the leaf node access performances of a specific technique's MBR-like and summary-like R-trees allow to draw conclusions with regard to the degeneration of the summary-like R-tree's structure.

13.3.4. RESULTS FOR THE R_25 SCENARIO. In this section, as final scenario, the R_25 scenario is evaluated. Again, we commence with assessing the MBR-like R-trees' query results before analyzing those of the summary-like R-trees.

Query Results for the MBR-like R-trees in the R_25 scenario

We now briefly analyze the query results for the R_25 scenario's MBR-like R-trees. Anew, we begin with the 10NN results. Here, the MBR approach requires a total of 7.7. page accesses from which 4.05 are to internal nodes and 3.64 are to leaf nodes (see Table 87).

Table 87: kNN query results for the MBR-like R-trees in the R_25 scenario.

	MBR	MBRQT_1	MBRQT_2	MBRQT_3	MBRQT_4	MBRQT_5	MBRQT_6	QTMBR_1	QTMBR_2	QTMBR_3	QTMBR_4
k=10											
#Dist2Rect	482.7	1,276.4	1,276.4	1,276.4	2,158.1	2,158.1	2,158.1	647.5	597.0	,1061.3	3,135.9
#Dist2Sum	482.7	482.7	482.7	482.7	482.2	482.2	482.2	533.9	486.0	530.7	504.9
#PA total	7.7	7.6	7.6	7.6	7.5	7.5	7.5	8.7	7.7	7.8	7.1
#PA IN	4.05	4.05	4.05	4.05	4.04	4.04	4.04	4.46	4.07	4.44	4.23
#PA LN	3.64	3.57	3.57	3.57	3.42	3.42	3.42	4.28	3.67	3.38	2.86
k=1,000											
#Dist2Rect	772.6	2,519.1	2,519.1	2,519.1	4,442.0	4,442.0	4,442.0	982.4	906.8	1,856.0	6,201.3
#Dist2Sum	772.6	772.6	772.6	772.6	771.4	771.4	771.4	848.9	777.6	845.8	806.3
#PA total	83.5	83.2	83.2	83.2	82.5	82.5	82.5	87.1	83.7	83.2	80.6
#PA IN	6.43	6.43	6.43	6.43	6.42	6.42	6.42	7.05	6.47	7.03	6.70
#PA LN	77.08	76.79	76.79	76.79	76.08	76.08	76.08	80.05	77.25	76.13	73.90

As usual for MBR-like R-trees, the MBRQT^{c,a} parameterizations slightly improve the MBR approach. The MBRQT^{8,a} parameterizations require 7.6 page accesses whereas the MBRQT^{16,a} parameterizations require 7.5 page accesses. Within each of the two groups, the results are identical. Due to the leaf node capacity of 25 and the consequential greater spatial spread of their data points, it now also applies for the leaf node summaries that only parameter c is of relevance: Parameter a is obviously never undercut before the maximum number of quadtree cells c depletes. The congruence of the results within the two groups is expected as already in the structural analysis, the equality of the three MBRQT^{8,a} parameterizations' R-trees respectively the three MBRQT^{16,a} parameterizations' R-trees has been ascertained. The MBRQT^{8,a} parameterizations exhibit the exact same numbers for the internal node accesses as the MBR approach but are slightly superior for the leaf node accesses. In contrast, the MBRQT^{16,a} parameterizations are already minimally better for the internal node accesses at level 2 (see Table 88) and marginally better than the MBRQT^{8,a} parameterizations at leaf node level. Nevertheless, the MBRQT^{16,a} parameterizations'

reduction of the amount of leaf node accesses as opposed to the MBR approach is yet fairly low as it is only 0.22 accesses (which corresponds to 6.0%). Due to the given regular spatial distribution of the data points and the leaf node capacity of 25, the outward appearance of MBRQT^{c,a} summaries is probably rarely more favorable than that of MBR summaries. Additionally, the leaf nodes are still largely free from overlap even when utilizing MBR summaries (see top image of Figure 129). Hence, the solely marginal improvements are easily comprehensible.

Table 88: Average amount of internal node accesses for the MBR-like R-trees in the R_25 scenario. The internal node accesses are displayed level-wise.

	MBR	MBRQT_1	MBRQT_2	MBRQT_3	MBRQT_4	MBRQT_5	MBRQT_6	QTMBR_1	QTMBR_2	QTMBR_3	QTMBR_4
level 0	1	1	1	1	1	1	1	1	1	1	1
level 1	1.23	1.23	1.23	1.23	1.23	1.23	1.23	1.28	1.23	1.28	1.26
level 2	1.82	1.82	1.82	1.82	1.81	1.81	1.81	2.18	1.84	2.16	1.97

With regard to the QTMBR$^{b}_{c,a}$ parameterizations, there are no surprises: QTMBR_2 is best of its peer group concerning internal node accesses, QTMBR_4 is best of all techniques with respect to leaf node accesses. For the leaf node accesses, QTMBR_4 achieves a reduction of 21.4% compared to the MBR approach. In the given situation (large leaf node capacity and large freedom of mutual overlap between leaf nodes), this is a fairly respectable result— which is achieved at the expense of high storage space requirements, though (the average memory consumption of the leaf node summaries is 68.4 B for QTMBR_4 as opposed to 16.0 B for the MBR approach). Due to its great leaf node access performance, QTMBR_4 is also prime with regard to the total amount of page accesses despite the usual weakness of the QTMBR$^{b}_{c,a}$ parameterizations with regard to internal node accesses. In Figure 129, visualizations of MBR summaries (top), MBRQT_6 summaries (middle), and QTMBR_4 summaries (bottom) of a specific, randomly selected leaf node set in the R_25 scenario's basic R-tree are displayed in a comparative view. Considering these, it is easily understandable why the leaf node access results between the MBR approach, MBRQT_6, and QTMBR_4 are as shown in Table 87.
Interestingly, QTMBR_3 (2.16 level-2-node accesses) has a minimally better internal node access performance than QTMBR_1 (2.18). This is remarkable because we analyze MBR-like R-trees (i.e. identical structures for all R-trees) and the prerequisites for the internal nodes' summary calculation are the same for both ($td = 2$ and $b = 8$). The differences yet occur because the *leaf node summaries* are the input for the level-2-nodes' summary-from-summaries calculation. Hereby the QTMBR_3 leaf node

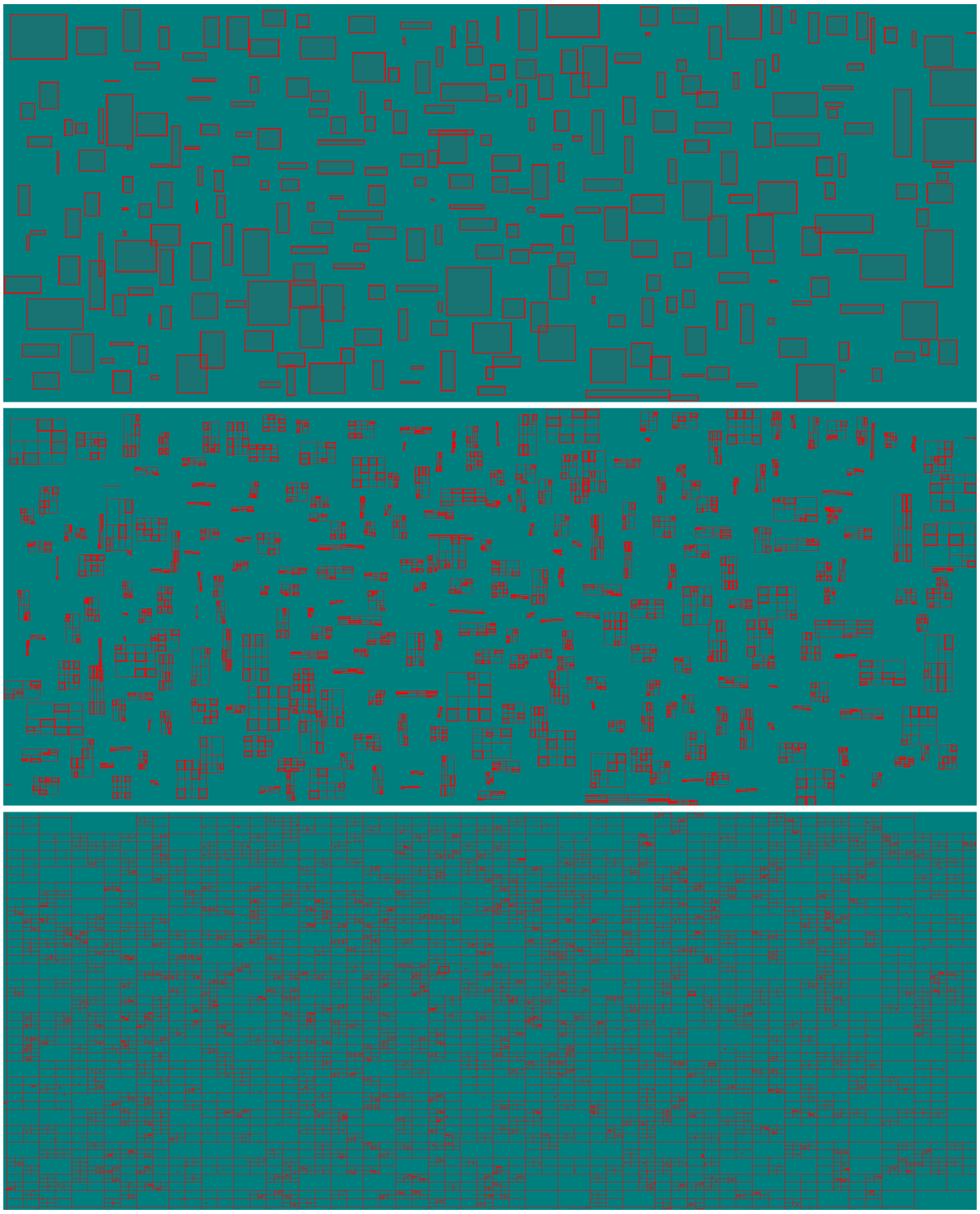


Fig. 129: Comparison of leaf node summaries for the MBR approach (top), MBRQT_6 (middle), and QTMBR_4 (bottom) in the R_25 scenario's basic, MBR-like R-tree. The respectively indexed areas are colored with the darker green.

[332]Note that the slight superiority of QTMBR_3 for the amount of page accesses at the level above leaf node level is already observable in Table 74 (leaf node access results for the MBR-like R-trees in the T_5 scenario, on page 374) and Table 85 (leaf node access results for the MBR-like R-trees in the R_5 scenario, on page 410) but has not been discussed in the corresponding analyses.

summaries are significantly more accurate than those of QTMBR_1 (see results for the leaf node accesses). As a consequence, slight advantages for QTMBR_3 result at level 2.[332]
For $k = 1,000$, basically the same results as for $k = 10$ are evident. Solely the techniques which are good with regard to the leaf node access performance are a little more competitive now as the total page access results are dominated by the leaf node accesses. For example, QTMBR_3 (83.2 total page accesses) is now minimally better than QTMBR_2 (83.7). However, this pattern has already been observed for the query results in previously evaluated scenarios.
In general, the amounts of page accesses for MBR-like R-trees are the lowest in the R_25 scenario: In comparison to the R_5 scenario, the greater leaf node capacity leads to significant decreases. Furthermore, the height of the R_25 scenario's basic R-tree is lower. Relative to the T_25 scenario, it has a regular data point distribution as opposed to an erratic one and is thus much easier to handle, i.e. the nodes can be kept largely free from overlap which allows for a significantly lower amount of page accesses—irrespective of the utilized summaries.

Query Results for the Summary-like R-trees in the R_25 scenario

In Table 89, the kNN query results of the R_25 scenario's summary-like R-trees are displayed. For $k = 10$, the MBR approach now requires 7.6 total page accesses from which 4.03 are to internal nodes and 3.62 are to leaf nodes. That makes it the best technique of all, here.

Table 89: kNN query results for the summary-like R-trees in the R_25 scenario.

	MBR	MBRQT_1	MBRQT_2	MBRQT_3	MBRQT_4	MBRQT_5	MBRQT_6	QTMBR_1	QTMBR_2	QTMBR_3	QTMBR_4
k=10											
#Dist2Rect	481.0	1,055.8	1,052.9	1,089.7	1,767.2	1,703.4	1,767.5	743.9	631.9	1,031.0	1,768.5
#Dist2Sum	481.0	409.7	403.8	419.3	406.5	396.5	405.6	634.4	531.4	557.7	475.7
#PA total	7.6	9.0	9.2	9.3	9.0	8.7	9.0	8.4	7.9	8.4	9.6
#PA IN	4.03	5.35	5.34	5.50	5.39	5.28	5.35	4.21	4.18	4.78	6.20
#PA LN	3.62	3.66	3.86	3.83	3.63	3.46	3.63	4.20	3.76	3.66	3.38
k=1,000											
#Dist2Rect	772.2	2,210.9	2,240.7	2,276.9	,3946.9	3,734.6	3,776.7	1,127.3	935.3	1,741.5	3,873.1
#Dist2Sum	772.2	677.6	684.4	692.7	684.0	656.1	661.2	997.7	817.0	843.5	705.5
#PA total	83.4	85.3	86.2	86.0	85.4	84.6	84.9	86.0	84.1	85.1	87.4
#PA IN	6.43	8.31	8.45	8.55	8.44	8.18	8.19	6.22	6.55	7.84	11.52
#PA LN	77.01	76.97	77.78	77.43	77.01	76.43	76.73	79.75	77.57	77.31	75.90

All MBRQT^{c,a} parameterizations require at least 1.1 page accesses more. This is mainly caused by their deficiencies with regard to the internal node accesses. In the R_25 scenario, the summary-like MBR R-tree has a height

of 4 whereas those of the MBRQT^{c,a} parameterizations all have a height of 5—which comes on top of the latter's lower fanout and greater amount of internal nodes. But also for the leaf node accesses, only MBRQT_5 is better than the MBR approach.[333] Due to the regular data point distribution and the great leaf node capacity in the R_25 scenario, the reductions of the leaf nodes' indexed surface areas are not as pronounced anymore. See Figure 130 for a visualization of a set of leaf node summaries for MBRQT_6. Furthermore, the effects of the misassignment of data points to leaf nodes are obviously more present here, again (as opposed to the R_5 scenario's summary-like R-trees where the effects were almost immeasurable). Compare the leaf node access results of the MBRQT^{c,a} parameterizations with each other: For example, MBRQT_1 (3.66 leaf node accesses) is clearly better than MBRQT_3 (3.83). Also, Figure 113 on page 369 showed that for the R_25 scenario's summary-like R-trees, the differences with regard to the spatial scattering of the leaf nodes' data points are more pronounced than in the R_5 scenario. Hence, it is comprehensible why in the R_25 scenario, the MBR approach is most often superior to the MBRQT^{c,a} parameterizations with regard to leaf node access performance even though it has been the other way round in the R_5 scenario. Nevertheless, the *absolute* differences between the techniques are smaller than ever.

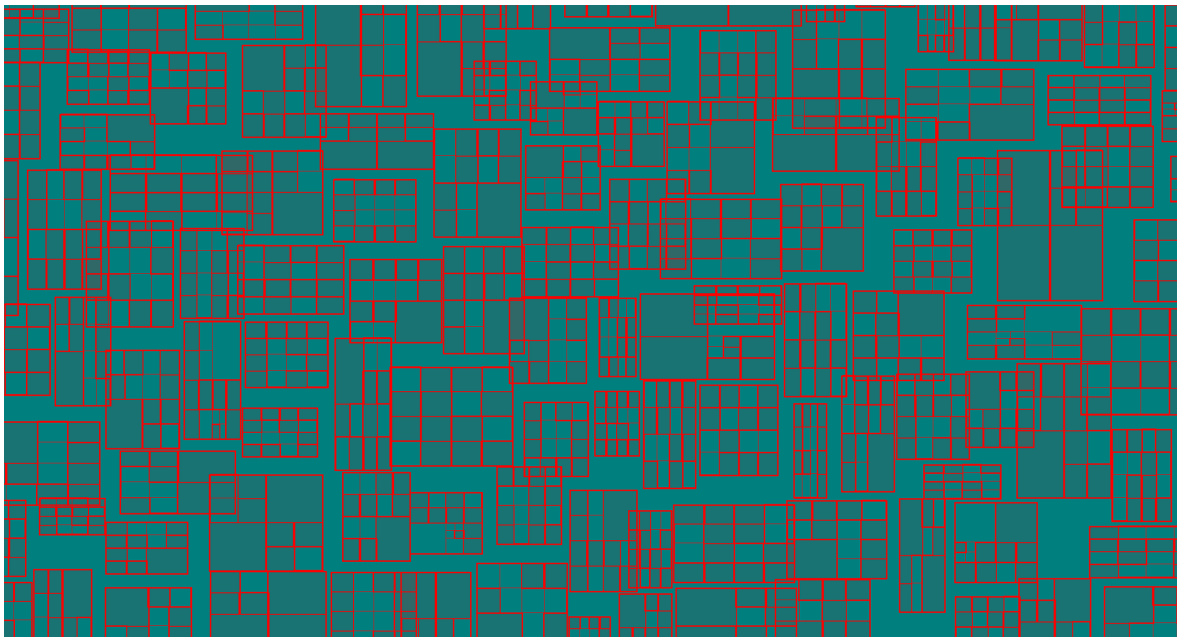


Fig. 130: Example visualization of leaf node summaries of the R_25 scenario's summary-like MBRQT_6 R-tree (visualization tool at the same zoom level as in the images of Figure 129).

For the QTMBR$^{b}_{c,a}$ parameterizations, QTMBR_4 is now the worst technique in its peer group because its R-tree has a height of 5 whereas those

[333] In the R_5 scenario's summary-like R-trees, all MBRQT^{c,a} parameterizations have been better for the amount of leaf node accesses than the MBR approach. See Table 86 on page 417.

of the other QTMBR$^{b}_{c,a}$ parameterizations all have a height of 4. Hence, QTMBR_4 is behind due to the consequential deficiencies in its internal node access performance. Notably, QTMBR_1 is now clearly better than QTMBR_3 with regard to internal node accesses (4.21 as opposed to 4.78). This is caused by the former's greater fanout (174.0 as opposed to 88.8) and its lower amount of internal nodes (8,339 as opposed to 16,339). It shows that at least for *intra*-approach comparisons, the fanout and the amount of internal nodes have very considerable effects on the results (as the general spatial accuracy should be almost identical for the internal node summaries of QTMBR_1 and QTMBR_3 because for both, $td = 2$ and $b = 8$). For *inter*-approach comparisons, this influence is hard to assess as lower fanouts might also be compensated by a greater spatial accuracy of the summaries. Interestingly, for the first time, QTMBR_1 to QTMBR_3 are better than the MBRQT^{c,a} parameterizations. This is also simply because the former's R-trees have a lower height (4 as opposed to 5). Due to the general convergence of the techniques' page access performances for the R collection, the R-trees' heights are of great importance for the R_25 scenario's overall result. Therefore, it is important to achieve large fanouts, here.

The importance of a large fanout is additionally underlined by the following, unique observation: For $k = 1,000$, QTMBR_1 (6.22 internal node accesses) is better than the MBR approach (6.43) for the internal node accesses. Hence, for the first time for summary-like R-trees, the MBR approach is outperformed for internal node accesses. For the $r = 1,000$ range queries, the results are similar: QTMBR_1 requires 6.19 internal node accesses on average while for the MBR approach, it is 6.41 accesses. The superiority of QTMBR_1 can only be caused by the greater fanout (174.0 as opposed to 118.4) and the lower amount of internal nodes (8,339 as opposed to 12,234) of the QTMBR_1 R-tree in comparison to the MBR R-tree. Again, it is an indication that for greater result sets, the general structural properties of the R-trees become more significant while the importance of the summaries' spatial accuracy decreases a little—which is intuitively comprehensible.

Apart from that, all results are as expected. Hence, we conclude the assessment of the R_25 scenario's query results. Since it is also the last scenario to consider, this simultaneously marks the end of the query result analyses. Therefore, the following section summarizes the key insights from the extensive evaluation of the centralized application scenario once again.

13.4. Summarization of the Results for the Centralized Application Scenario

In this section, we present an overview of the results of the centralized application scenario. In particular, we summarize the key insights from the extensive evaluation (section 13.4.1), assess the degree of achievement with regard to thesis objective ❷ (section 13.4.2), and outline several starting points for future work (section 13.4.3).

13.4.1. KEY RESULTS OF THE EVALUATION. In the following, the key insights from the extensive evaluation of the centralized application scenario are concisely summarized. In some cases, we already directly provide ideas for problem solutions or directions for further research.

We first start with some general observations. One aspect is that at least for erratic data, the cumulated indexed surface areas are unreliable indicators for assessing the techniques' indexing performance—even in *intra-approach* comparisons. This is because the indexed surface areas of the nodes exhibit typical long tail distributions, i.e. few nodes index very large surface areas while the majority of the nodes index only very small surface areas. Consequently, the cumulated result is dominated by the former. The long tail distribution of indexed surface areas applies to both internal as well as leaf nodes. A more suitable means for assessing the *universal* indexing performance is a comparison of the techniques' distributions of the indexed surface area per node—but only for *MBR-like* R-trees (where the structures of all techniques' R-trees are identical). For *summary-like* R-trees, the structures of the techniques' R-trees differ and thus, the node ranks are not really comparable to each other. Hence, a sole assessment of the distributions of the indexed surface area per node might conceal possible degenerations of specific summary-like R-trees. Therefore, for the summary-like R-trees, an investigation of the spatial scatterings of the leaf nodes' data points is essential in order to make well-founded statements. In general, differences between the techniques' results are greatly amplified by erratic data. For regularly distributed data points, the divergences diminish to fairly large extents, and also the indexing-related key figures are not as deceptive as before. Hence, here, the cumulated indexed surface areas are usually also a fairly appropriate representation of the distributions of the indexed surface area per node—even in inter-approach comparisons. It is therefore not absolutely necessary to examine the corresponding distributions of the indexed surface area per node. However, it has to be kept in mind that still, degenerations of summary-like R-trees might exist. Consequently, it is important to consider the spatial scatterings when investigating the reasons for the outcomes of query results.

Another insight is that the provision of the zipping option for the $\mathrm{MBRQT}^{c,a}$ and $\mathrm{QTMBR}^{b}_{c,a}$ summaries can be omitted completely. The reason is that both MBR data (such as for the full-precision basic MBR of $\mathrm{MBRQT}^{c,a}$ summaries or the quantized refinement MBRs of $\mathrm{QTMBR}^{b}_{c,a}$ summaries) as well

as linearly encoded quadtree data (for the refining quadtree of MBRQT^{c,a} summaries or the basic quadtree of QTMBR$^{b}_{c,a}$ summaries) is of high entropy and can only be compressed in *extremely* rare cases. In this context, it has been shown that often, the set of MBRQT^{c,a} respectively QTMBR$^{b}_{c,a}$ summaries of a specific R-tree is almost completely encoded in either the LQ scheme *or* the CBLQ scheme. In such cases, the dominant encoding scheme could be preselected—which has to be cautiously considered from case to case, though. Both measures (the omission of the zipping option and the preselection of the appropriate encoding scheme in suitable situations) would reduce computational costs for the summary calculation and also storage space requirements as the metadata information can be reduced or completely omitted because the amount of available summary description types is narrowed down. In this matter, a conceivable idea for *leaf node summaries* in R-trees with a low leaf node capacity would be to provide the direct representation of data points as an *additional* option. As shown in the evaluation, it is possible that the summaries of in particular higher QTMBR$^{b}_{c,a}$ parameterizations consume more storage space than the data points themselves. Furthermore, it is generally a suitable description in case only one or two data points are stored in a leaf node.
As a further finding, it has been frequently observed that the memory utilization rates of the MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summaries are fairly poor which means that a significant portion of the data stored on disk is actually filling data introduced by the byte- and word-alignment. This puts our MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ parameterizations at a disadvantage in comparison to the MBR approach (whose summaries do not feature any filling data at all). Since simultaneously, it has been shown that the differences in the amount of conducted distance calculations are generally non-critical for the overall runtime (due to the dominance of the time required for page accesses), there is no reason to not make the best possible use of the available storage space, i.e. creating more accurate summaries with eventually greater amounts of indexed areas.[334]
We now proceed to discuss the key insights from the results of the MBR-like R-trees. In general, with the MBR-like R-trees, the aim was to assess if there is any *potential* to improve the results of MBR summaries for the way we utilize and parameterize our MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summaries, and how large this eventual potential for improvement actually is. As this was the first assessment of an integration of our summarization approaches into R-trees, there was no previous knowledge in this re-

[334] In this context, it is nonetheless important to also consider the runtime expenses for the deserialization of the summaries (i.e. the reconstruction of the indexed areas from the summaries' bit vectors). As has already been outlined in the evaluation of the runtimes of the R-tree constructions (section 13.1), this is an issue which requires further work. Before conducting an all-encompassing evaluation with optimized memory utilization rates of MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summaries, it is therefore necessary to solve this problem.

gard. For the MBR-like R-trees, a basic R-tree is built by utilizing MBR summaries in the construction phase and the MBR summaries are then replaced by the respective techniques' summaries without considering any storage space limitations. The resulting MBR-like R-trees are mostly unrealistic as most of the techniques require more storage space than the MBR approach and therefore, the capacity of the R-tree's internal nodes (4,096 B from which 16 B are reserved for general metadata) is frequently exceeded for most techniques. However, as just mentioned, the goal of the MBR-like R-trees was 'only' to explore the general *potential* for improvements, and not the creation of 'realistic' R-trees. Table 90 outlines the relative results of the techniques for the 10NN queries in the diverse scenarios, once again (since for the 10NN queries, the greatest relative differences occurred).

Table 90: Relative results of the total page access performances for the 10NN queries which were conducted for the *MBR-like* R-trees in the diverse scenarios. The MBR approach is the benchmark, i.e. its listed absolute amounts of total page accesses correspond to 100% in each row.

	MBR	MBRQT_1	MBRQT_2	MBRQT_3	MBRQT_4	MBRQT_5	MBRQT_6	QTMBR_1	QTMBR_2	QTMBR_3	QTMBR_4
T_5	18.3	99.1%	98.6%	98.1%	96.9%	95.9%	94.6%	3,580%	237.4%	670.2%	314.1%
T_25	13.7	99.2%	98.5%	98.5%	97.0%	95.3%	94.8%	1,132%	145.2%	280.0%	164.8%
R_5	10.5	98.0%	97.9%	97.8%	96.6%	96.2%	96.0%	124.8%	101.1%	107.0%	97.9%
R_25	7.7	99.0%	99.0%	99.0%	97.1%	97.1%	97.1%	113.8%	100.8%	101.7%	92.2%

In general, it shows that the QTMBR$^b_{c,a}$ approach can get *massive* problems in regions of very high data point density (such as they occur in the T collection $\rightarrow$ T_5 scenario and T_25 scenario) which is due to its inherent property that the smallest indexable spatial unit is the cell of an occupied quadtree region's quantization grid—which in contrast to a full-precision MBR *usually* cannot be of 'infinite' accuracy (i.e. a point in the extreme case). Depending on the parameterization of QTMBR$^b_{c,a}$, the problems arise for both internal node summaries as well as leaf node summaries, and to varying degrees. The deficiencies at the *internal nodes*' levels are massively fostered by the specification of the summary-from-summaries calculation algorithm for QTMBR$^b_{c,a}$: Due to the restriction of the basic quadtree's maximum depth to target depth td, the QTMBR$^b_{c,a}$ summaries cannot make use of their usually extremely effective minimization of the indexed surface areas. Hence, in the end, they mostly only conform to coarser approximations of the corresponding full-precision MBRs. In particular for internal nodes in regions of very high data point density, this is a massive disadvantage. The more accurate the smallest indexable unit, the more the problems can be mitigated ($\rightarrow$ accurate quantization of QTMBR_2 with $b = 12$). At *leaf node* level, the general results for the QTMBR$^b_{c,a}$ parameterizations are

similar with the exception of QTMBR_4 which is parameterized in such a way that almost absurdly small surface areas are indexed for the leaf nodes. Furthermore, QTMBR_4 is also able to actually index points or at least very short lines for spatially extremely narrow data point sets. As a consequence, with QTMBR_4, the amount of leaf node accesses can be reduced by very significant extents in comparison to the MBR approach (see Table 91). Of course, the use of QTMBR_4 summaries results in partly massively increased storage space requirements (which in 'realistic' R-trees would increase the amount of internal nodes). Nevertheless, it shows that at least for the leaf nodes, there is indeed *potential* for improvements over the utilization of MBR summaries.

Table 91: Relative results of the *leaf node* access performances for the 10NN queries which were conducted for the *MBR-like* R-trees in the diverse scenarios. The MBR approach is the benchmark, i.e. its listed absolute amounts of leaf node accesses correspond to 100% in each row.

	MBR	MBRQT_1	MBRQT_2	MBRQT_3	MBRQT_4	MBRQT_5	MBRQT_6	QTMBR_1	QTMBR_2	QTMBR_3	QTMBR_4
T_5	5.83	99.0%	97.4%	95.6%	98.4%	95.6%	91.3%	9,308%	460.9%	320.0%	82.7%
T_25	4.95	97.9%	96.1%	95.9%	95.0%	90.4%	89.1%	2,469%	203.1%	155.5%	71.7%
R_5	5.33	96.1%	95.9%	95.6%	93.3%	92.6%	92.3%	130.7%	101.3%	96.9%	87.8%
R_25	3.64	97.9%	97.9%	97.9%	94.1%	94.1%	94.1%	117.7%	100.8%	92.7%	78.4%

The MBRQT^{c,a} approach does not exhibit problems comparable to those of the QTMBR$^{b}_{c,a}$ approach. In general, all MBRQT^{c,a} parameterizations slightly improve the MBR approach for both internal node accesses as well as leaf node accesses. This is because their summaries are guaranteed to be at least as accurate as an MBR summary. However, they also display some suboptimal properties: For the internal nodes, the data points administered in the respective subtrees are usually so numerous and so spread that there are only rather few cases in which regions of dead space can be excluded from indexation. Hence, only marginal improvements are achieved over the MBR approach for the internal nodes' levels, both with regard to the spatial accuracy of the summaries (measured on basis of the cumulated indexed surface areas and the distributions of the indexed surface area per node) as well as the internal node access performance. For the leaf nodes, in regions of very high data point density, the threshold surface area a (which prevents the refinement of the basic MBRs) is already frequently undercut by the basic MBRs. Hence, the corresponding MBRQT^{c,a} summaries are only equivalent to the matching MBR summaries. For the T collection, this in particular affects the MBRQT^{c,a} parameterizations exhibiting the larger values for parameter a such that for these, the share of leaf nodes for which the MBRQT^{c,a} summaries are actually more accu-

rate than the corresponding MBR summaries is fairly low. In contrast, the MBRQT^{c,a} parameterizations featuring the lower values for parameter a improve the indexed surface areas of very significant shares of leaf nodes. However, it is evident that the conversion of these improvements into leaf node access performance generally suffers from various aspects:

— The respective 'outward appearances' of the MBRQT^{c,a} summaries in comparison to the MBR summaries are most often not as favorable as the numerical reductions of the indexed surface areas suggest.
— The fairly well-pronounced freedom from overlap between the MBR summaries of the *leaf nodes* (despite partially huge amounts of overlap between the internal nodes' MBR summaries) diminishes the advantageousness of spatially more accurate summaries.
— The query radii of the kNN and range queries increase the access probabilities of the nodes irrespective of their actually indexed surface areas. This further diminishes the advantageousness of more accurate summaries.
— The MBR approach already exhibits very good results, i.e. the room for further improvements is generally relatively limited.

For MBRQT^{c,a}, the outcome is that the potential for improvements is in the range of about 5.5% in the best case. The findings so far were mostly outlined with regard to the results for the T collection's scenarios which feature very erratic data. In the R collection's scenarios, the *internal node* access performances generally converge: Due to regular spatial distribution of the data points, it basically never occurs that MBRQT^{c,a} summaries can exclude regions of dead space from indexation. The QTMBR$^{b}_{c,a}$ summaries profit from the absence of extreme data point agglomerations which mitigates their outlined deficiencies for the lower-level internal nodes. Furthermore, the amounts of internal node accesses are generally significantly lower because there is much less overlap between the internal nodes' summaries. Also, the leaf node access performances tend to converge as the SELECTBEST(.)-method and the R*-split are even a bit more successful in keeping the leaf nodes free from overlap which generally simplifies the retrieval task. Additionaly, the deficiencies in the maximum spatial accuracy of the QTMBR_1, QTMBR_2, and QTMBR_3 leaf node summaries are far less severe since there are no extremely densely populated data space regions anymore.
In general, our summarization approaches are in some sense caught in a conceptual dilemma with regard to improving the MBR approach:

— On the one hand, it would be beneficial if the leaf node capacity is low. This is because then, the query result is dispersed over more leaf nodes. Consequently, it is more likely that unnecessary leaf node accesses are conducted due to the spatial coarseness of the MBR summaries. This is from a viewpoint considering the *R-tree structure*.

— On the other hand, it would be beneficial if the leaf node capacity is high. A greater capacity leads to a greater spatial spread of the data points stored in the leaf nodes. The ability of the MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summaries to reduce the indexed surface areas relative to MBR summaries could therefore be exploited more profitably. This is from a viewpoint considering the *summaries of the leaf nodes*.

The outlined dilemma emphasizes the importance of the leaf node accesses for the total page access performance, obviously. The assumption underlying our experiments was that basically, it is decided at *leaf node level* whether the MBR approach can be improved or not. Generally, this is a reasonable assumption as the majority of nodes in the R-tree are leaf nodes. Furthermore, especially for larger result sets, the total amount of page accesses is usually strongly dominated by the amount of leaf node accesses. Consequently, the focus in our experiments was sharply on improving the leaf node access performance. However, after conducting the evaluation, we think that there are also situations in which a greater focus on the amount of internal node accesses can be of benefit, too. We come back to this during the discussion of the directions for future work (see section 13.4.3).
Overall, the results for the MBR-like R-trees show that the potential for improvements over the MBR approach is limited but existent. The MBRQT^{c,a} parameterizations offer slight but constant potential for improvement. Unfortunately, the QTMBR$^{b}_{c,a}$ parameterizations exhibit diverse inherent deficiencies which result in that the QTMBR$^{b}_{c,a}$ approach generally cannot improve the MBR approach for MBR-like R-trees (except in special cases in which, however, huge amounts of storage space are consumed $\rightarrow$ leaf node access performance of QTMBR_4). With these findings, it was clear that *in the current form of usage*, the chances of improving the MBR approach for realistic, summary-like R-trees were very low (MBRQT^{c,a}) respectively basically non-existent (QTMBR$^{b}_{c,a}$) because generally, deteriorations in the performance of our summarization approaches are to be expected. This is due to a) the applied algorithms (such as the SELECTBEST$(.)$-method or the R*-split) which generally fit best to the MBR approach while no specific adjustments to the properties of the MBRQT^{c,a} and in particular QTMBR$^{b}_{c,a}$ summaries are conducted, and b) the necessity of compliance with the storage space capacity of the internal nodes (which is frequently exceeded for the MBR-like MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ R-trees), obviously.
We now consider the results for the summary-like R-trees. The summary-like R-trees are built by utilizing the respective techniques' summaries in the construction phases and, in contrast to the MBR-like R-trees, comply with the storage space restrictions of the internal nodes. Therefore, they are 'realistic' R-trees. For reasons of comparability, the same general algorithms—in particular the SELECTBEST$(.)$-method and the R*-split—are utilized for all summary-like R-trees. Table 92 shows the summary-like R-trees' relative results in the diverse scenarios for the 10NN queries.

Table 92: Relative results of the total page access performances for the 10NN queries which were conducted for the *summary-like* R-trees in the diverse scenarios. The MBR approach is the benchmark, i.e. its listed absolute amounts of total page accesses correspond to 100% in each row.

	MBR	MBRQT_1	MBRQT_2	MBRQT_3	MBRQT_4	MBRQT_5	MBRQT_6	QTMBR_1	QTMBR_2	QTMBR_3	QTMBR_4
T_5	18.6	108.7%	112.3%	113.8%	113.7%	109.0%	119.8%	5,161%	282.9%	977.6%	634.0%
T_25	13.6	119.4%	144.3%	141.2%	119.7%	116.4%	132.0%	1,501%	163.2%	402.9%	387.3%
R_5	10.3	99.3%	101.1%	98.2%	98.0%	96.9%	99.1%	125.4%	103.0%	110.4%	103.8%
R_25	7.6	117.9%	120.2%	122.0%	118.0%	114.3%	117.5%	110.1%	103.8%	110.5%	125.4%

The structural analysis of the summary-like R-trees reveals that for erratic data (i.e. in the T_5 scenario and in the T_25 scenario), the nodes in the summary-like QTMBR$^{b}_{c,a}$ R-trees generally exhibit problems with their memory utilization rates which, depending on the parameterization, occur for the internal nodes (QTMBR_4), for the leaf nodes (QTMBR_2), or for both (QTMBR_1, QTMBR_3). The problems are caused by the interplay of very high data point densities in specific regions of the data space, too coarse smallest indexable spatial units (either for the internal node summaries or the leaf node summaries), and the not truly random implementation for resolving ties of equally well-suited candidate nodes in the SELECTBEST(.)-method. Especially for the QTMBR$^{b}_{c,a}$ parameterizations with low leaf node memory utilization rates (QTMBR_1, QTMBR_2, and QTMBR_3), this means that they are at a disadvantage before even considering other structural properties of their R-trees or spatial properties of their summaries: The less data points are stored in leaf nodes on average, the more leaf nodes usually have to be accessed to retrieve the query result. Furthermore, the problems that occurr for the *MBR-like* QTMBR$^{b}_{c,a}$ R-trees still exist. In addition, it has been shown that the *summary-like* QTMBR$^{b}_{c,a}$ R-trees additionally suffer from degenerations: The SELECTBEST(.)-method is prone to make inappropriate assignments of data points to nodes when utilizing QTMBR$^{b}_{c,a}$ summaries during the R-tree construction phase. The result is that in the summary-like QTMBR$^{b}_{c,a}$ R-trees, the SELECTBEST(.)-method fails to keep coherent clusters of data points closely together in the R-tree structure. The consequential degeneration of the summary-like QTMBR$^{b}_{c,a}$ R-trees is indicated by fairly many corresponding observations: The significantly greater spatial scatterings of the leaf nodes' data points in comparison to those of the corresponding summary-like MBR R-trees, the substantially increased amount of indexed areas of the summary-like QTMBR_4 R-trees as opposed to their corresponding MBR-like QTMBR_4 R-trees (despite comparable amounts of nodes in the respective R-trees), or the circumstantial evidence from the comparative visualizations of the spatial extents of the internal nodes' MBRs where e.g. Hawaii is clearly

discernible for the summary-like MBR R-tree but not for the summary-like QTMBR_4 R-tree. However, the ultimate proof for the degeneration is still the clearly deteriorated leaf node access performance of the summary-like $\text{QTMBR}^{b}_{c,a}$ R-trees in comparison to their corresponding MBR-like $\text{QTMBR}^{b}_{c,a}$ R-trees given comparable leaf node memory utilization rates. In total, for the T collection, the summary-like $\text{QTMBR}^{b}_{c,a}$ R-trees are *much* worse than the summary-like MBR R-trees.

For the $\text{MBRQT}^{c,a}$ parameterizations, the differences in the leaf node access performances between their MBR-like and summary-like R-trees as well as the spatial scatterings of their leaf nodes' data points in their summary-like R-trees indicate that they also suffer from degenerations of their summary-like R-trees—albeit to significantly lower extents than the $\text{QTMBR}^{b}_{c,a}$ parameterizations. However, this is already sufficient for them to fall behind the MBR approach in terms of the leaf node access performance in almost every case. In addition, in the summary-like $\text{MBRQT}^{c,a}$ R-trees, the fanout is a constant 25% smaller and there are always 33% more internal nodes as opposed to the summary-like MBR R-trees. This deteriorates their internal node access performances. Of course, also the suboptimal properties identified for the MBR-like $\text{MBRQT}^{c,a}$ R-trees continue to apply for the summary-like $\text{MBRQT}^{c,a}$ R-trees. As a consequence, the $\text{MBRQT}^{c,a}$ parameterizations cannot compete with the MBR approach for the T collection.

For the R collection's scenarios (R_5 and R_25), the differences between the techniques' summary-like R-trees generally diminish such that the inferiority of our summarization approaches as opposed to the MBR approach is substantially less pronounced. Also, the memory utilization rates of the summary-like $\text{QTMBR}^{b}_{c,a}$ R-trees are now on comparable levels with those of the other techniques' R-trees (due to the absence of extreme data point agglomerations). In the R_5 scenario, five out of the six $\text{MBRQT}^{c,a}$ parameterizations even improve the MBR approach for the total amount of page accesses by between 0.7% (MBRQT_1) and 3.1% (MBRQT_6). It is noteworthy that the improvements are exclusively due to the leaf node access performances. However, especially in the R_5 scenario, the summary-like R-trees of the higher $\text{MBRQT}^{c,a}$ and $\text{QTMBR}^{b}_{c,a}$ parameterizations show a tendency to index almost each data point with an area of its own[335]—which is only a moderately reasonable indexation as neither the spatial accuracy nor the computational efficiency benefit. In contrast to the leaf node access performances, there is *not a single occasion* in which our summarization approaches improve the internal node access performance of the MBR approach for 10NN queries (see Table 93).

In general, there are four aspects which influence the internal node access performance of summary-like R-trees:

a) The height of the R-tree.

[335] Note that this also applies to the corresponding MBR-like R-trees.

Table 93: Relative results of the *internal node* access performances for the 10NN queries which were conducted for the *summary-like* R-trees in the diverse scenarios. The MBR approach is the benchmark, i.e. its listed absolute amounts of internal node accesses correspond to 100% in each row.

	MBR	MBRQT_1	MBRQT_2	MBRQT_3	MBRQT_4	MBRQT_5	MBRQT_6	QTMBR_1	QTMBR_2	QTMBR_3	QTMBR_4
T_5	12.52	112.7%	117.7%	119.8%	118.2%	114.9%	129.2%	1,126%	148.9%	1,183.8%	873.0%
T_25	8.74	126.0%	154.8%	154.5%	127.2%	126.2%	142.7%	383.6%	129.7%	460.5%	512.0%
R_5	5.13	102.3%	104.3%	100.6%	102.8%	100.5%	102.5%	114.5%	102.8%	118.1%	115.9%
R_25	4.03	132.9%	132.5%	136.6%	133.8%	131.1%	132.9%	104.6%	103.9%	118.8%	154.0%

b) The spatial accuracy of the summaries (which includes the eventual degeneration of the R-tree).
c) The number of internal nodes in the R-tree.
d) The fanout of the R-tree.

Obviously, the overall composition of these aspects (almost) always turns out favorable for the MBR approach. However, we think our summarization approaches provide the tools necessary in order to induce changes in this regard. We come back to this in the discussion of the starting points for improvements and future work (see section 13.4.3).
Generally, it is also evident that the greater the result sets, the more important the general properties of the R-tree structures become for the internal node access performances (as e.g. demonstrated by the superiority of QTMBR_1 over the MBR approach in terms of the internal node access performance for the 1,000NN queries in the R_25 scenario). In particular, this applies to collections of regularly distributed data points. However, since in almost every other case, the structural properties of the R-trees are in favor of the MBR approach, the superiority of the MBR approach for the internal node accesses generally even increases due to the greater significance of the general R-tree structure.
In total, it has to be admitted that the MBR approach is unmatched for our data collection representing 'real-life' data: the T collection ($\rightarrow$ T_5 scenario and T_25 scenario). The marginal improvements achieved by the MBRQT^{c,a} parameterizations in the rather artificial R_5 scenario do not really seem to justify the additional efforts necessary for integrating our summarization approaches into an R-tree, especially since in the R_25 scenario, the MBR approach is prime, again. As mentioned before: the MBR approach already exhibits very good results in all scenarios. In contrast, our summarization approaches have to cope with several unfavorable circumstances:

— The SELECTBEST(.)-method is evidently not well-suited for use with MBRQT^{c,a} and in particular QTMBR$^{b}_{c,a}$ summaries.
— Although the R*-split generally achieves good results with regard to the freedom of overlap between leaf nodes, it is heavily optimized for the

use with MBR summaries. While the general aims of the R*-split (minimization of surface area covered by MBRs/overlap between MBRs/MBR perimeters) are reasonable from a universal point of view and do not seem to be unsuited in terms of utilizing $\text{QTMBR}^{b}_{c,a}$ and especially $\text{MBRQT}^{c,a}$ summaries, it is likely that with specifically adapted split algorithms, our summarization approaches might achieve better results than with the R*-split. In this context, note that Engl investigates diverse optimization strategies for R-trees utilizing MBR-, $\text{MBRQT}^{c,a}$-, and $\text{QTMBR}^{b}_{c,a}$-based summaries in [Engl 2018].

— In principle, a lot of what is the purpose of our sophisticated summaries is already covered by the split algorithm. Abstractly, both try to minimize overlap between 'resources' but attempt to achieve this in different ways: The summarization approaches seek to delineate the data point clouds so accurately that the mutual overlap diminishes due to the sheer detail of the description whereas the split algorithms try to separate the data point clouds appropriately. In the distributed application scenario, one important aspect why our summarization approaches are so successful is that there is no control over the assignment of data points to resources. As a consequence, the data point clouds of the resources are heavily 'intermixed'—which naturally leads to massive amounts of overlap between many resources' MBR summaries. Generally, the summaries' freedom from overlap is the most important asset in spatial indexing. Especially the $\text{QTMBR}^{b}_{c,a}$ summaries are able to minimize the mutual overlap between resources by basically indexing suchlike small areas that the probability of overlap diminishes dramatically. Furthermore, the adaptiveness of our summarization approaches—i.e. to spend a lot of storage space for spatially spread resources while investing only minimal amounts for spatially narrow resources—also has very beneficial effects. In the R-tree, however, the assignment of data points to nodes is controllable. A suitable split algorithm like the R*-split ensures a sufficient freedom from overlap between the 'resources' (i.e. the nodes) which heavily mitigates the advantages of our summarization approaches. Additionally, the benefits of our approaches' adaptiveness are not as useful anymore since the split algorithm prevents very large spatial spreads and greatly differing cardinalities of the data point sets which are administered by the 'resources' (i.e the R-tree's nodes).
— As mentioned before, it was the first time we conducted and evaluated the integration of our summarization approaches into an R-tree. Therefore, we did not even have broader empirical data on how to parameterize the $\text{MBRQT}^{c,a}$ and $\text{QTMBR}^{b}_{c,a}$ approaches apart from rather small test runs and general intuitions. In contrast, traditional R-trees utilizing MBR summaries have been the subject of intensive research and optimizations for decades.
— Diverse inherent problems of our methodological approaches (such as the consequences for the spatial accuracy of the internal nodes' $\text{QTMBR}^{b}_{c,a}$

summaries due to the specification of the summary-from-summaries calculation, or the partially low memory utilization rates of our MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summaries) only became apparent to us during the evaluation at hand.

At this point, we conclude the summarization of the key insights from the evaluation. In the following, the degree of achievement with regard to thesis objective ❷ is assessed.

13.4.2. DEGREE OF THESIS OBJECTIVE ACHIEVEMENT AND CONTRIBUTIONS. The aim of thesis objective ❷ was the integration of suitable summarization approaches into an R-tree and the evaluation of the resulting improvement potential in comparison to the use of MBR summaries. For the integration, the MBRQT^{c,a} approach as well as the QTMBR$^{b}_{c,a}$ approach were identified as suitable candidates because they are 'self-organizing' approaches. This means they only require the data point set to describe as input which makes them well-suited for application in an R-tree, and also easy to integrate. We successfully integrated these two approaches while applying only few adjustments compared to 'classical' variants of the R-tree: The most important ones are a differentiation between the summary calculation processes for leaf nodes ($\rightarrow$ summary-from-data-points calculation) and internal nodes ($\rightarrow$ summary-from-summaries calculation), as well as the adjustments required due to the potentially varying storage space sizes of the MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summaries.[336] An example for such an adjustment is the eventual need for a split of a parent node due to the exceeding of its available storage space even though none of its child nodes was split but only a summary of one of its child nodes was updated—which, however, results in a larger size of the child node's summary, leading to the necessity of the parent node's split. With the integration, the foundation to pursue the more important part of thesis objective ❷—the assessment of the improvement potential—has been laid. In the subsequent extensive evaluation, MBR-like R-trees served as a benchmark for assessing the existing improvement potential. Summary-like R-trees were used to gauge the achieved degree of realization for our straightforward approach of keeping the changes in comparison to 'traditional' R-trees to a minimum. In this context, it has been shown that indeed, the spatial coarseness of MBR summaries yields potential for further improvements. At least at leaf node level, our MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summaries are able to drastically reduce both the cumulated indexed surface areas as well as the indexed surface area per node (for both MBR-like *and* summary-like R-trees). Furthermore, the assessment of the MBR-like R-trees reveals that improvement potential is definitely existent in terms of the amount page accesses required while querying, too.

[336] The latter changes could also be omitted by conducting appropriate adjustments to the summary calculation processes, see section 13.4.3.

However, the evaluation of the summary-like R-trees shows that our very straightforward approach of conducting as few adjustments as possible to the R-tree while integrating—with regard to their calculation processes—*mainly unchanged* MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summaries does not immediately lead to improvements of 'traditional' R-trees utilizing MBR summaries. The shortcomings of the resulting summary-like MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ R-trees as well as the critical reasons for the worse query performances have already been discussed in section 13.4.1.
Overall, we nonetheless think that the integration of our summarization approaches into the R-tree was successful: Not necessarily with respect to the query performance, but in view of acquiring knowledge about the potential and the crucial problem areas of the integration. It is obvious that it is not possible to improve the R-tree by orders of magnitude anymore. However, our evaluation provides a very solid foundation for trying to achieve *notable* improvements on basis of our summarization approaches in the course of further research. Therefore, thesis objective ❷ has been fully achieved.
Within the context of the centralized application scenario, an important contribution of this thesis is the successful implementation of R-trees using MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summaries while simultaneously identifying necessary modifications to 'traditional' R-trees. Furthermore, the specification of the summary-from-summaries calculation processes for both the MBRQT^{c,a} approach as well as the QTMBR$^{b}_{c,a}$ approach is to be mentioned. From a more general point of view, this is equivalent to a specification of how to summarize region data by use of these approaches. However, the main contribution is the extensive evaluation and the knowledge acquired from it in terms of existing improvement potential over MBR summaries in R-trees as well as important shortcomings and problem areas of integration efforts. The latter refers to both the *general* integration of sophisticated summaries as well as *specifically* to the concrete integration of MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summaries.
In this context, in the following section, we present an overview of starting points for improvements and ideas to *profitably* utilize MBRQT^{c,a} as well as QTMBR$^{b}_{c,a}$ summaries in connection with 'realistic' R-trees.

13.4.3. STARTING POINTS FOR FUTURE WORK. As this was the first assessment of integrating our summarization approaches into an R-tree, many ideas for improvement and starting points for future work result from the evaluation. Some pretty straightforward aspects, part of which have already been mentioned, are the following:

— Omit the zipping option for the MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summaries, and adjust the respective bit vector structures accordingly.
— Preselect the suitable linear quadtree encoding scheme in appropriate situations. A concrete implementation requires some additional experiments with regard to which scheme should be preselected in which situa-

tions. In general, it is clear that the LQ code is preferable when few black cells are built in the respective quadtree structures while the CBLQ code is beneficial when many black cells are created.
— The resolution of ties in the SELECTBEST(.)-method has to be changed to a true random select.
— The deserialization process (i.e. the reconstruction of the indexed areas from the binary information stored in a summary's bit vector) has to be further investigated, both with regard to a more suitable implementation of the utilized data structure representing the bit vector as well as the resulting consequences on the runtimes.
— Conduct further parameter tuning for $\text{MBRQT}^{c,a}$ and $\text{QTMBR}^{b}_{c,a}$.

A more complex aspect is the increase of the $\text{MBRQT}^{c,a}$ and $\text{QTMBR}^{b}_{c,a}$ summaries' memory utilization rates which oftentimes were fairly poor. The most reasonable approach is to change the summary calculation process in such a way that they are not primarily dependent on preset parameters but instead primarily consider the available storage space, e.g. when it is specified that 24 B are available for $\text{MBRQT}^{c,a}$ summaries. An adapted calculation process for $\text{MBRQT}^{c,a}$ summaries could be as follows: After each decomposition of an occupied quadtree cell, it is calculated how much storage space the encoding of the quadtree structure would require. This is fairly straightforward as in the LQ code, each literal requires 3 bits and in the CBLQ code, each literal requires 2 bits. As just discussed, the encoding scheme might also have been preset. Independently of this, in case the available storage space is not exceeded for at least one of the available encoding schemes, the decomposition continues. In contrast, if the storage space limitations are exceeded, the last decomposition is undone and the correspondingly reversed quadtree structure is then taken as refinement. Such an $\text{MBRQT}^{c,a}$ summary calculation process should be very easy to implement and also very runtime-efficient. For $\text{QTMBR}^{b}_{c,a}$ summaries, it is almost as simple as for $\text{MBRQT}^{c,a}$: It solely needs to be taken into account that each black quadtree cell also contains a quantized MBR which requires $4 \cdot b$ bits in addition. The explicit calculation of the quantized MBRs only has to take place after the final quadtree structure has been determined and is not required for the ascertainment whether the available storage space is exceeded (as the $4 \cdot b$ bits for a quantized MBR are invariable). Hence, also the storage-space-driven calculation of the $\text{QTMBR}^{b}_{c,a}$ summaries should not introduce noteable computational overhead. A suchlike specification of the summary calculation process is also suitable to circumvent the problems introduced by potentially varying sizes of $\text{MBRQT}^{c,a}$ respectively $\text{QTMBR}^{b}_{c,a}$ summaries. For example, it could easily be specified that *all* $\text{QTMBR}^{b}_{c,a}$ summaries shall consume a constant 24 B. Independent from this, for both the adapted $\text{MBRQT}^{c,a}$ and $\text{QTMBR}^{b}_{c,a}$ summary calculation processes, it is advisable to include an additional checkup to prevent a further decomposition of black quadtree cells

in case these cells become so small that problems with the accuracy of the floating-point number format might occur.
As another point, it has been shown that the summary-like MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ R-trees suffered from degenerations due to the inappropriate specification of the SELECTBEST$(.)$-method. There are basically two conceivable ways to solve this issue:

— The summary-like MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ R-trees could be built by using MBR summaries during the construction phase. After the construction of the R-tree structure, the MBR summaries are then replaced by the corresponding MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summaries. Thereafter, eventual overfull nodes need to be split. Hereby, a bottom-up procedure would be necessary, i.e the splitting of any overfilled nodes has to begin at the level above the leaf node level. As shown in section 13.2.1, most nodes of the R-tree should usually be preservable (even with the current specification of the summary calculation processes). By means of the just outlined storage-space-driven summary calculation procedures, it additionally would be much more controllable what the resulting MBRQT^{c,a} respectively QTMBR$^{b}_{c,a}$ R-trees look like. Independently of this, for the nodes which have to be split *post*-construction, the memory utilization rates would have to be checked, though. We suspect that the memory utilization rates of the resulting, split nodes could be rather low. In case of too low rates, it would be conceivable to dissolve 'underfilled' nodes and then apply reinsertion mechanisms. However, detailed specifications with regard to the dissolving and the reinsertion have to be developed in the course of corresponding future work. A starting point for the development of appropriate reinsertion mechanisms is e.g. the reinsertion algorithm of the R*-tree [Beckmann et al. 1990, p. 326f].
— A more straightforward approach (which would be more expensive in terms of the construction phase's runtime) would be to adapt the SELECTBEST$(.)$-method to the properties of the MBRQT^{c,a} respectively QTMBR$^{b}_{c,a}$ summaries. It was shown that especially the QTMBR$^{b}_{c,a}$ R-trees suffered to significant extents. A simple first idea for a SELECTBEST$(.)$-method adapted to QTMBR$^{b}_{c,a}$ summaries is to select the most appropriate node on basis of the MINDIST between the newly inserted data point dp and the nodes' QTMBR$^{b}_{c,a}$ summaries. Ties could be resolved with the corresponding current specification, i.e. selecting the node for which the center point of its summary's MBR is closest. Still undecided situations (i.e. there is a set of candidate nodes which are equally well-suited according to the MINDIST and the center point) are to be resolved by a *truly random* assignment of dp to one of the candidate nodes. Due to the quantization applied by the QTMBR$^{b}_{c,a}$ summaries, eventual susceptibilities with regard to conducting unsuitable data point assignments to nodes in a *traditional* sense—which express themselves in thin, elongated MBRs describing the nodes in traditional R-trees—are, in our

opinion, not very pronounced. Adequate split algorithms can additionally counteract. For MBRQT^{c,a} summaries, a MINDIST-based SELECTBEST(.)-method might not be quite as appropriate as for QTMBR$^{b}_{c,a}$: We see a possibility of eventually occuring degenerations of their basic full-precision MBRs when applying a MINDIST-driven assignment of data points to nodes. On the other hand, in traditional R-trees using MBR summaries, this issue is generally handled by the utilization of appropriate split algorithms (like the R*-split) rather than by adaptions of the SELECTBEST(.)-method. Hence, also due to the ease of implementation, the outlined MINDIST-driven algorithm is still our recommendation as starting point for further assessments: From a general point of view, it is the most reasonable assignment (and is therefore presumably more suitable than the currently applied center-point-driven assignment). For both MBRQT^{c,a} and QTMBR$^{b}_{c,a}$, we would also initially adhere to the R*-split. Research in terms of more suitable split algorithms could take place afterwards.

The combination of an adapted SELECTBEST(.)-method (with a truly random resolution of ties) and adapted summary calculation processes in order to increase the summaries' memory utilization rates should definitely enhance the competitiveness of the summary-like MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ R-trees. Whether this is enough to outperform summary-like MBR R-trees is an open question, though. The results for the cumulated indexed surface areas and the distributions of the indexed surface area per node suggest that at leaf node level, it is easily possible to notably improve the MBR approach in case one succeeds to prevent the degeneration of the summary-like MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ R-trees. However, it is questionable if these improvements are sufficient with regard to compensating for the MBR approach's superior internal node access performance. In this context, we have divergent ideas to improve the internal node access performances of our summarization approaches in R-trees storing erratic and regularly distributed data point sets:

— For erratic data, we think it is worthwhile to test the utilization of very detailed MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summaries to describe internal nodes. As the evaluation of the T_5 scenario and the T_25 scenario showed, the amounts of internal node accesses in relation to the heights of the respective R-trees were very considerable. For example, for the T_5 scenario's MBR-like MBR R-tree, 12.51 internal node accesses were required on average for the 10NN queries even though the height of the R-tree was just 5 (i.e. there were only four levels of internal nodes). In Table 74 on page 374, we showed that e.g. more than two accesses were already conducted at level 1—even though level 1 had only five nodes. This is definitely improvable. As the reason for these results was the huge amount of overlap between the internal nodes' summaries at the same R-tree level, we suspect that very detailed MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summaries might be capable of achieving improvements. Of course, the more

detailed summaries would decrease the corresponding R-trees' fanouts while simultaneously increasing the numbers of internal nodes which increase the R-trees' heights in the long run. However, even if the height raises by one or two levels, there is still plenty of room for improvement over the MBR approach with its ratio of e.g. 4 internal node levels to 12.51 internal node accesses. A concern with regard to applying this approach might be the overall runtime of the R-tree construction when utilizing significantly more detailed MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summaries for the description of internal nodes. Hereby, optimized algorithms which accelerate the summary-from-summaries calculation have to be developed.[337] In case these runtimes are still too long, there would yet be the possibility of utilizing MBR summaries during the R-tree construction and replacing them afterwards (as outlined before), though.

— For regular data, the internal node access performance of the MBR approach was fairly close to the optimum when considering the corresponding R-trees' heights. For example, for the R_5 scenario's MBR-like MBR R-tree, 5.14 internal node accesses were conducted on average for an R-tree whose height was 5 (i.e. again, four levels of internal nodes existed). Due to the general convergence of the results in the R_5 scenario and the R_25 scenario, we identified that the fanouts, the numbers of internal nodes, and especially the R-trees' heights are of very great importance in suchlike scenarios ($\rightarrow$ regular data point distribution). Hence, we do not think it is suitable to work with detailed MBRQT^{c,a} and QTMBR$^{b}_{c,a}$ summaries for internal nodes here: the negative effects on the mentioned structural properties would be too great. This aspect and also the R collection's query results imply that for MBRQT^{c,a}, it would be best to simply utilize non-refined basic MBRs for the internal nodes' levels.[338] For QTMBR$^{b}_{c,a}$, it could be tested whether 8 B summaries with optimized memory utilization rates and an adapted SELECTBEST(.)-method (which prevents the degeneration of QTMBR$^{b}_{c,a}$ R-trees) are beneficial. At least, the maximum fanout of the nodes storing child node entries could be increased by 50% as only $8\,B + 8\,B$ (for the pointer) $= 16\,B$ are required for a corresponding child node entry as opposed to $16\,B + 8\,B = 24\,B$ like for child node entries with MBR summaries. Hence, for regular data point distributions, sacrificing spatial accuracy to increase the fanout might turn out favorably—which is a reversed approach to the one we pursued in the course of this thesis. Also, an investigation of QTMBR$^{b}_{c,a}$ R-trees which utilize 16 B summaries (i.e. the same amount of storage space as an MBR summary) would be interesting.

[337] Note that a conceivable algorithm is discussed in the last two paragraphs of section 13.4.3.

[338] Note that this already goes in the direction of creating hybrid R-trees, i.e. R-trees in which different summarization approaches are utilized. This idea is discussed separately below.

Related to this, it has to be admitted that the specification of the summary-from-summaries calculation process (i.e. the calculation process for summaries of internal nodes), at least for $\text{QTMBR}^{b}_{c,a}$, was based on not entirely realistic assumptions. Target depth td was specified to be calculated in such a way (as $\lfloor \log_4 c \rfloor$) that even in the absolute worst case (i.e. when each cell of the initial quadtree is occupied with data), no more than c cells result for the final quadtree. Since the basic quadtree of a $\text{QTMBR}^{b}_{c,a}$ summary decomposes the entire data space, this means that the data points which are associated with a node have to be spread across the entire data space to make the worst case happen. Even for the level directly below root level, this is extremely unlikely as the R-tree generally holds neighboring data points together, i.e. the data points associated with a node are usually all located in subspaces which are small compared to the entire data space. Hence, an adjusted determination of target depth td should be developed. A simple approach would be to set td depending on the level in the R-tree on which the node whose summary is recalculated is located. Future research could elaborate on a more sophisticated determination of td.
A further promising idea to improve the traditional R-tree could be the implementation of hybrid R-trees. With hybrid R-tree, we mean an R-tree in which summaries of *different* summarization approaches describe the nodes. There are various conceivable implementations of such R-trees:

— A very straightforward approach would be a combination of MBR and $\text{MBRQT}^{c,a}$ summaries. In such an R-tree, an MBR summary would be utilized when threshold surface area a is already undercut by the basic MBR. Otherwise, an $\text{MBRQT}^{c,a}$ summary is employed. In some sense, this is already implemented in the current usage of the $\text{MBRQT}^{c,a}$ summaries. However, for a *storage-space-efficient* distinction between MBR and $\text{MBRQT}^{c,a}$ summaries, we suggest a decoupled storage of the meta-information (i.e. the information about with which summarization approach a concrete summary instance was calculated) and the summary itself: Each parent node has an additional metadata block in which for each of its child node entries, the binary information on the sort of its associated summary is stored. If the summary of child node entry i ($0 \leq i < M$) is an MBR summary, the i-th bit of this metadata block is set to 1. Otherwise, it is 0—which means that the corresponding child node entry has an $\text{MBRQT}^{c,a}$ summary. Since only binary information is stored, the additional metadata block is very concise: Assuming a page size of 4,096 B and 16 B of general metadata per node (as specified in section 10.1), the *maximum* possible amount of child node entries is $4,080\,B/24\,B = 170$, i.e. $M = 170$.[339] Hence, the metadata block stor-

[339] 24 B is the minimal size of a child node entry in such a hybrid R-tree utilizing MBR and $\text{MBRQT}^{c,a}$ summaries: Additionally to the 8 B for the pointer, at least 16 B are required for an MBR *summary* (an $\text{MBRQT}^{c,a}$ *summary* has a size of *at least* 24 B after the byte- and word-alignment).

ing the information on which summarization approach was utilized for which child node's summary requires 170 bit or 21.25 B, i.e. the maximum fanout of a parent node decreases by only 1. In Figure 131, a conceptual depiction of a corresponding parent node's storage structure is depicted. Note that such a hybrid R-tree would require the MBRQT^{c,a} summaries to be of a fixed size.

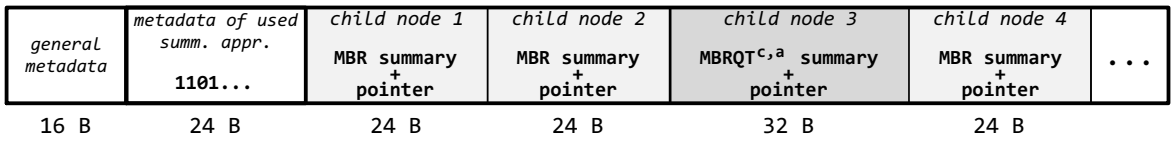


Fig. 131: Conceptual depiction of an internal node's storage structure in a hybrid R-tree which uses MBR and MBRQT^{c,a} summaries simultaneously.

— Another, more complex strategy for the implementation of hybrid R-trees would be to use specific summarization approaches at different levels of the R-tree. For instance, QTMBR$^{b}_{c,a}$ summaries could be used to describe nodes at the level below root node level, for the nodes at the level above leaf node level, MBR summaries might be utilized, and at the leaf node level, eventually MBRQT^{c,a} summaries are employed. The selection of the appropriate summarization approach to use at the respective level would depend on the different strengths and weaknesses of the various summarization approaches, obviously. For example, in the evaluation of the T_5 scenario's MBR-like R-trees, the QTMBR$^{b}_{c,a}$ summaries showed to be better than the full-precision-MBR-based approaches at level 1 and also partly at level 2 (see Table 74 on page 374) whereas at the level above leaf node level, the MBRQT^{c,a} summaries started to gain considerable advantages over the MBR summaries. Of course, these specific examples refer to MBR-like and therefore unrealistic R-trees. Nevertheless, it shows that different summarization approaches are differently well-suited at specific levels of an R-tree. Admittedly, a concrete implementation of such a complex hybrid R-tree is a distant prospect since a lot of additional research has to be conducted beforehand. The following is an excerpt of unresolved questions:
  — Which summarization approach should be utilized at which level, and how to parameterize these approaches?
  — How well do the previously discussed, very detailed MBRQT^{c,a} respectively QTMBR$^{b}_{c,a}$ summaries for internal nodes perform in R-trees for erratic data collections? What consequences does this have on the assessment of which summarization approach is most suitable at which level(s)?
  — How well does the just outlined simultaneous usage of MBR and MBRQT^{c,a} summaries *at the same level* of an hybrid R-tree perform? Should this idea also be integrated into the more complex strategy?

In general, a lot of combinations of any complexity are conceivable. However, in view of the extent to which the MBR approach can be improved at all, such a complex approach to hybrid R-trees is probably too much effort for too little benefit. Therefore, we suggest to focus on implementing and evaluating the other ideas first. Afterwards, it can be assessed whether the development of suchlike complex hybrid R-trees still appears to be promising.

Another aspect worth exploring is the utilization of $\text{QTMBR}^b_{c,a}$ summaries in application fields where *solitary* quantized MBRs are used. For instance and as already mentioned in the discussion of the related work in section 2.4.3, there are corresponding approaches such as those making use of the so-called QRMBR technique (these approaches are reviewed in [Hwang et al. 2003]). Another example of a multidimensionsal data structure employing solitary quantized MBRs is the A-tree [Sakurai et al. 2002]. There are certainly also fields aside from multidimensional data structures in which quantized MBRs are used. The linearly encoded basic quadtree of the $\text{QTMBR}^b_{c,a}$ summaries is a very storage-space-efficient means to reduce the data space onto which the quantization raster has to be imposed. We think that $\text{QTMBR}^b_{c,a}$ should be capable of outperforming the suggested approaches using quantized MBRs because these need to impose the quantization raster onto the *entire* data space—which has negative effects on the granularity of the quantization raster. A further field of application for $\text{QTMBR}^b_{c,a}$ summaries could be in higher-dimensional data spaces for which the fanout of an R-tree drastically decreases when utilizing MBR summaries. This is because two d-dimensional points need to be captured in full-precision to delineate an MBR in a d-dimensional space (which requires $2 \cdot d \cdot 4$ B assuming a single-precision floating-point number format). As in addition, usually, the distances between data points dramatically increase in higher-dimensional data spaces (due to the curse of dimensionality), the induced lower accuracy of an $\text{QTMBR}^b_{c,a}$ summary's quantized MBR in comparison to a full-precision MBR should also have significantly lower negative effects. A reasonable implementation would require a specification of the $\text{QTMBR}^b_{c,a}$ summary calculation process in such a way that only one quantized MBR results for an entire $\text{QTMBR}^b_{c,a}$ summary, though. Also, solely only-black-nodes-encodings (like the LQ code) are suitable.
Finally, the summary-from-summaries calculation processes lay a foundation for the use of $\text{MBRQT}^{c,a}$ and $\text{QTMBR}^b_{c,a}$ summaries to index region data. Consequently, their application in suitable application fields working with region data could be investigated. In this context, the specification of a target depth td still appears to be suitable to be the starting point of the summary calculation processes. However, at least one aspect requires additional work with regard to the $\text{QTMBR}^b_{c,a}$ summaries: As already mentioned before, another mechanism to determine td is necessary. In the evaluation of the centralized application scenario, it showed that

for data which is concentrated in very narrow regions of the data space, the specification of td being calculated as $\lfloor \log_4 c \rfloor$ led to too shallow initial quadtrees. In the end, it resulted in indexing only a single area which was equivalent to a coarser approximation of the corresponding full-precision MBR. We already discussed an approach which sets td dependent on the corresponding node's level in the R-tree. Of course, there are also conceivable possibilities which are not dependent on the existence of an R-tree structure. As a corresponding adaption to set td to more appropriate values, a very straightforward approach would be to replace parameters c and a with parameter td, i.e. to simply use manually set values for td. Of course, this would only work in supervised or well-known scenarios. Another idea would be to calculate td in dependence of a) the size of the MBR calculated from the region data to summarize (i.e. the input rectangles) and b) the size of the data space. The resulting consequences on the runtime of the $\text{QTMBR}^b_{c,a}$ summary calculation process have to be investigated carefully, though. In general, a greater target depth td increases the runtime exponentially as the number of cells in the initial quadtree is 4^{td}. In this context, feasible algorithms to accelerate especially the $\text{QTMBR}^b_{c,a}$ summary calculation have to be developed. A simple idea would be to calculate the MBR m of the input rectangles first. Afterwards, the cells of the initial quadtree which are overlapped or intersected by m are determined as set of overlapped or intersected quadtree cells oic. The subsequent overlap calculations between the input rectangles and the cells of the initial quadtree can then be restricted to the cells contained in oic (see section 10.4.2 again for the corresponding $\text{QTMBR}^b_{c,a}$ summary calculation process).
For these overlap calculations, a simple scanline algorithm can be applied for further acceleration. Hereby, the input rectangles are sorted in ascending order by their minimum x value in a list L_{input}. Afterwards, for each quadtree cell $qc_i \in oic$, the following is conducted: Iterate element-wise over L_{input}, with *elem* being the currently considered input rectangle. In case *elem*'s minimum x value is smaller than or equal to the maximum x value of qc_i, determine the overlap area of qc_i and *elem*, and cache it for the subsequent calculation of qc_i's quantized MBR. If *elem*'s minimum x value is greater than the maximum x value of qc_i ($elem.x_{min} > qc_i.x_{max}$), abort the iteration and start to calculate qc_i's quantized MBR from the cached overlap areas. See Figure 132 for an example visualization. There, we have three input rectangles $elem_1$, $elem_2$, and $elem_3$ (also sorted in this order in L_{input}). Both $elem_1$ and $elem_2$ need to be considered because their minimum x value is smaller than the maximum x value of the currently considered quadtree cell qc_i (light grey fill). When $elem_3$ is reached, the consideration of L_{input} can be aborted because $elem_3.x_{min} > qc_i.x_{max}$ (i.e. $elem_3$ and qc_i cannot overlap) and due to the sorting of L_{input}, eventual subsequent elements of L_{input} have an even greater minimum x value than $elem_3.x_{min}$. The problem with this scanline algorithm is that it becomes inefficient for the quadtree cells in the right half of the data space (cells with dark grey fill

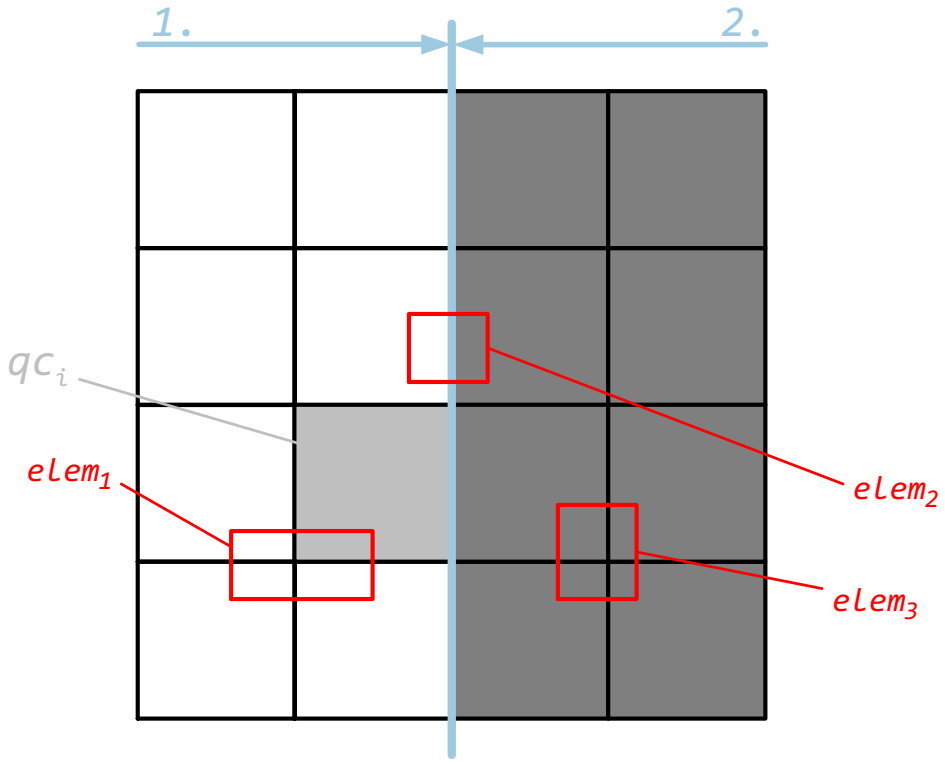


Fig. 132: Example visualization of the scanline-based, accelerated calculation of $\text{QTMBR}^{b}_{c,a}$ summaries from a set of input rectangles. In the first phase of the algorithm (1.), the list of input rectangles L_{input} is sorted in ascending order by its elements' minimum x value. In the second phase (2.), L_{input} is sorted in descending order by its elements' maximum x value.

in Figure 132). Therefore, also the $qc_i \in oic$ should be sorted in ascending order by their minimum x value and considered in this sequence. When the minimum x value of the current qc_i is greater than or equal to the mean value of the x-dimension, the first phase of the scanline algorithm ends and the elements in L_{input} need to be sorted in descending order by their maximum x value. The scanline is then effectively executed back-to-front, i.e. the condition for aborting the iteration is now $elem.x_{max} < qc_i.x_{min}$. Also see Figure 132 for a visualization of the scanline algorithm's second phase ($\rightarrow$ 2.). Note that the scanline algorithm does not necessarily have to run in the x-dimension but can also run in the y-dimension. In general, it could e.g. be specified that it runs in the dimension in which the MBR m has the greater extent.

Part IV:

Conclusion

14. OVERALL CONCLUSION

In this section, the thesis is briefly summarized as a whole. More detailed discussions of the results and the contributions can be found at the appropriate places in this work. See section 8.9 beginning on page 262 for the summarization for the distributed application scenario, and section 13.4 beginning on page 428 for the summarization for the centralized application scenario.

This thesis is concerned with the effective and efficient summarization of two-dimensional spatial data point sets. The term *effective* refers to a spatially very accurate delineation of the data point sets to summarize while the term *efficient* relates to a concise storage space footprint. The corresponding summaries can in particular be utilized in search systems working with spatial point data and for which the concept of resource description and selection is applicable. In general, there are many conceivable application fields for suchlike summaries, also aside from search-based scenarios. However, in this thesis, two similarity-search-based spatial application scenarios are assessed.

The first one is a distributed application scenario in which the summarization approaches are utilized for efficient searches in distributed systems such as peer-to-peer systems (in which the peers constitute the resources whose spatial footprints have to be described). The summaries' shall enable the targeted selection of the resources administering the relevant data points. It is the main part of the thesis for which the summarization approaches are specifically developed. Aside from the MBR approach which utilizes the well-known Minimum Bounding Rectangles (MBRs) to summarize the spatial contents of resources, 13 further summarization approaches are presented in this thesis. They can be classified into three categories: data partitioning approaches, space partitioning approaches, and hybrid approaches. The evaluation of the approaches for various data collections and environmental conditions shows that in each case, the query performance of the MBR approach can be improved by most of our summarization approaches when utilizing the same amount of storage space. In particular, specific hybrid approaches improve the MBR approach by several orders of magnitude in each assessment. Except for a single pathological case, also the former state-of-the-art approaches are significantly improved by the hybrid approaches which have been developed during the course of this dissertation project. In general, the employment of sophisticated summaries in a distributed search environment is a well explored and understood application scenario— which is reflected in the results.

As a second field of application, the utilization of sophisticated summaries in a centralized application scenario is investigated in this thesis, in the context of spatial searches supported by a centralized multidimensional data structure. More specifically, the multidimensional data structure is an R-tree which is a hierarchical, tree-based structure that has been the subject of intensive research for decades. Summaries are used to describe

the spatial footprints of the R-tree's nodes (which are the 'resources' in this scenario). The summaries enable the targeted selection of paths in the R-tree structure when querying. Traditionally, an R-tree uses MBR summaries to describe its nodes' spatial contents. The goal of the centralized application scenario's investigation is to verify whether the utilization of sophisticated summaries within such a well-researched multidimensional data structure can still yield potential for improvements. Consequently, the two summarization approaches which are most suitable for application in the given scenario are selected and integrated into an R-tree. Hereby, we pursue a very straightforward approach, i.e. the extents of the changes both in terms of the summary calculation processes as well as the R-tree structure are kept as minimal as possible. The evaluation shows that improvement potential is existent. Nonetheless, it cannot be realized due to various reasons which have been identified and discussed in the course of the evaluation. As this is the first time evaluation of such an endeavor, the complex interplay of sophisticated summaries and R-tree structure have not been understood well enough to improve the MBR summaries in this environment *yet*. On basis of the results and the evaluation, a profitable utilization of summarization approaches within an R-tree appears to be possible but definitely requires more tuning and further research. The thesis at hand provides a very solid foundation for future work in this area, and also supplies various ideas and starting points for future enhancements.

A. APPENDIX

Directly Transmitted Data Points for the Techniques Assessed in the Evaluation of the Query Radius Reduction (T1 collection)

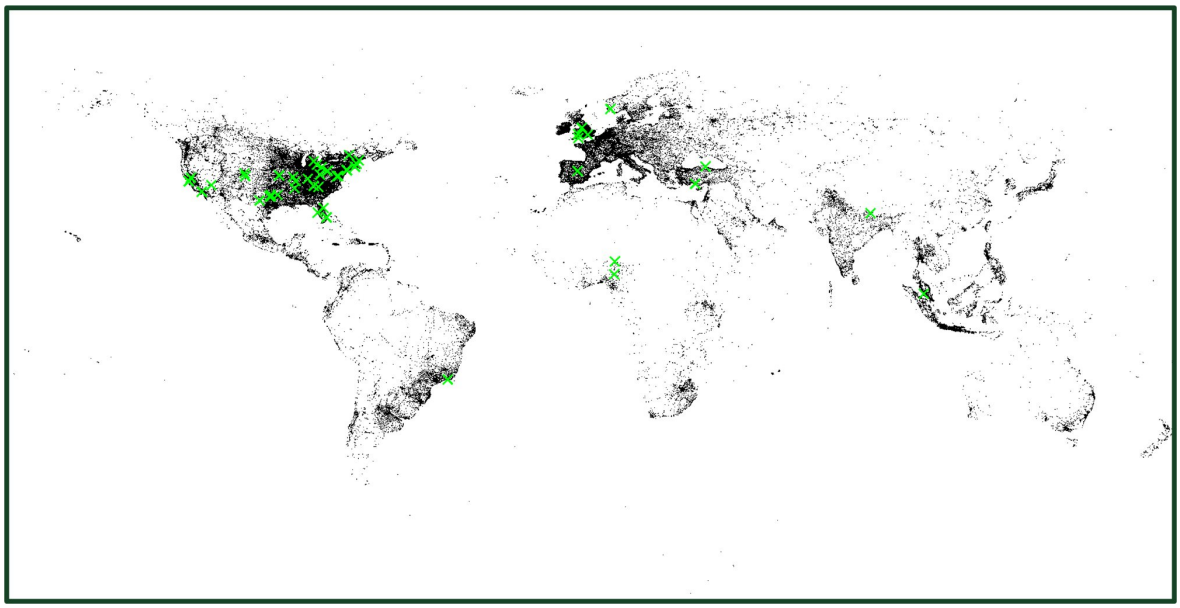


Fig. 133: Directly transmitted data points for the MBR approach and the T1 collection.

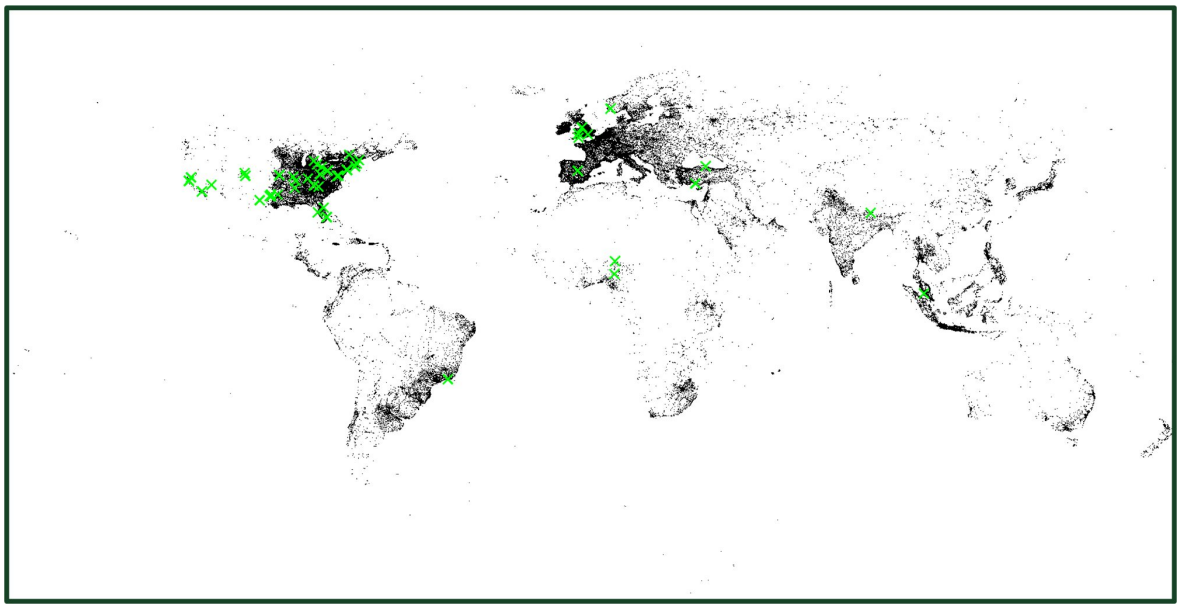


Fig. 134: Directly transmitted data points for $\mathrm{UFS}_{256,cc}$ and the T1 collection.

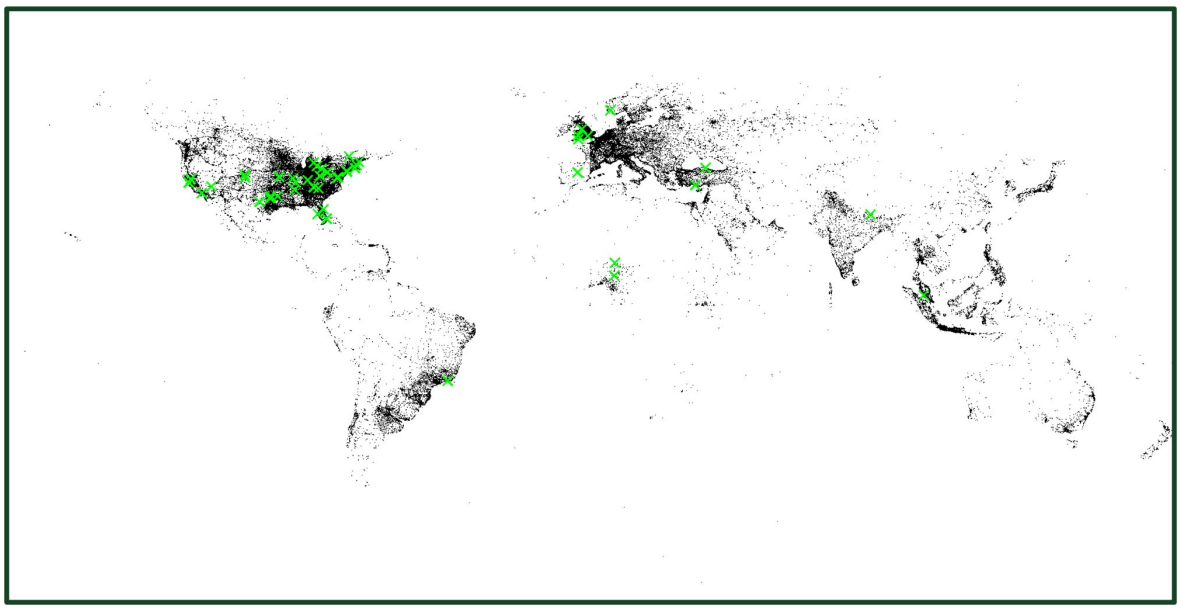


Fig. 135: Directly transmitted data points for KDMBR^{6}_{256} and the T1 collection.

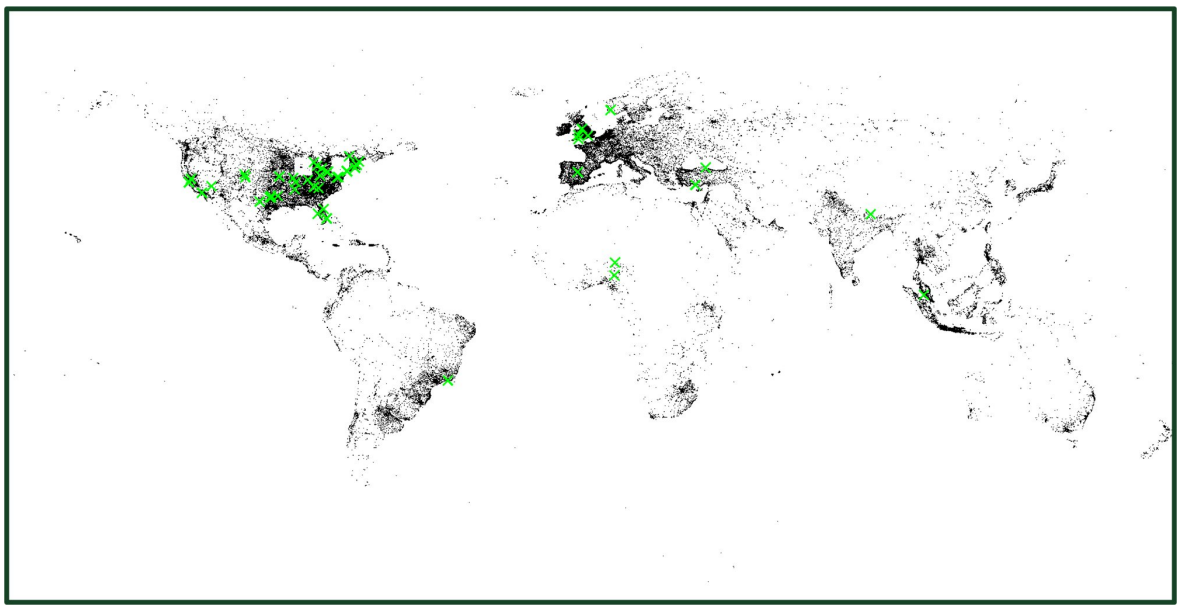


Fig. 136: Directly transmitted data points for $\text{KDQT}^{64,1.0E-5}_{32}$ and the T1 collection.

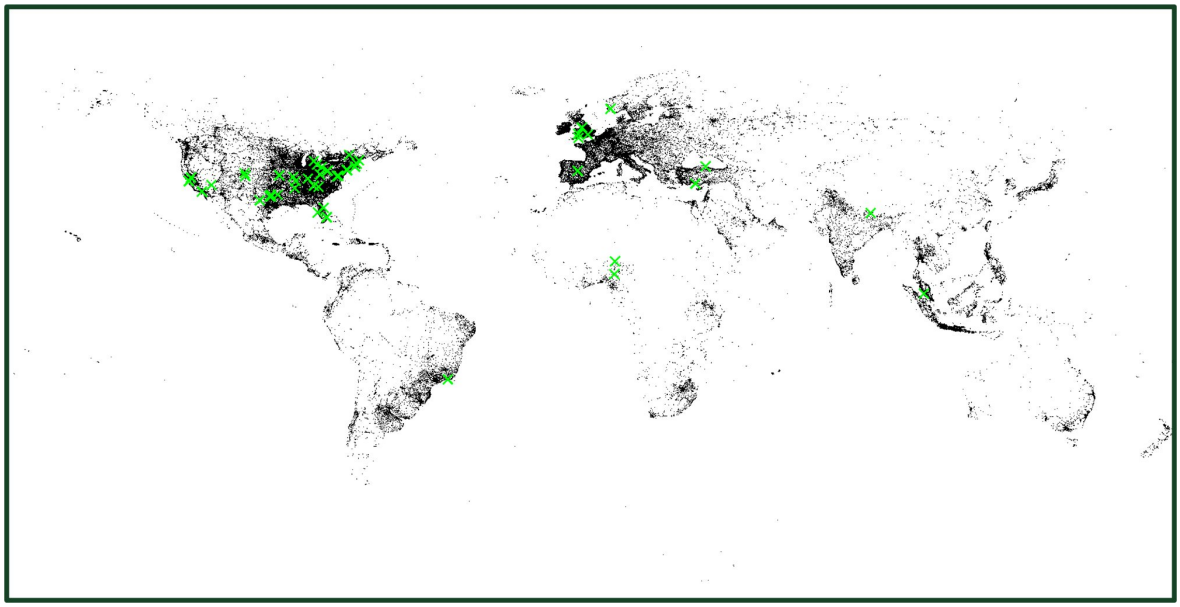


Fig. 137: Directly transmitted data points for $\text{QTMBR}^{6}_{512,0.05}$ and the T1 collection.

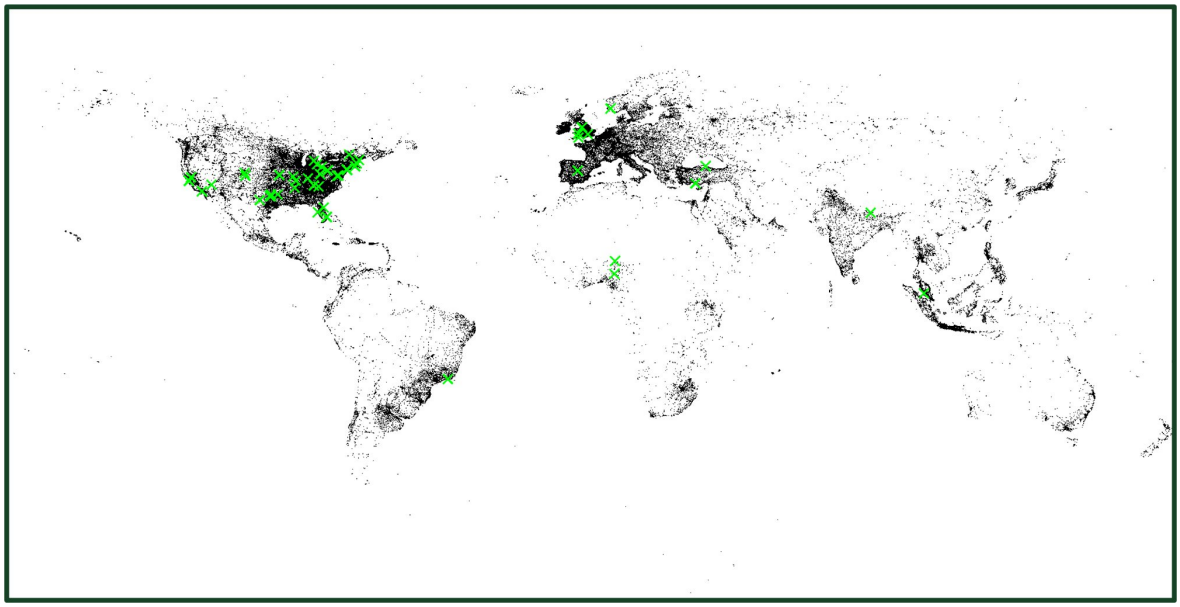


Fig. 138: Directly transmitted data points for $\text{MBRQT}^{16,1.0}$ and the T1 collection.

List of Figures

List of Tables

REFERENCES

D. J. Abel. 1984. A B^+-tree structure for large quadtrees. *Computer Vision, Graphics, and Image Processing* 27, 1 (1984), 19–31. DOI: http://dx.doi.org/10.1016/0734-189X(84)90079-3 (Cited on page 54.)

N. Abramson. 1963. *Information theory and coding*. McGraw-Hill. https://books.google.de/books?id=x4hQAAAAMAAJ (Cited on page 22.)

H. K. Ahn, N. Mamoulis, and H. M. Wong. 2001. *A Survey on Multidimensional Access Methods*. Technical Report UU-CS-2001-14. Department of Information and Computing Sciences, Utrecht University. (Cited on pages 36 and 50.)

Giuseppe Amato and Pasquale Savino. 2008. Approximate similarity search in metric spaces using inverted files. In *3rd International ICST Conference on Scalable Information Systems, INFOSCALE 2008, Vico Equense, Italy, June 4-6, 2008*. 28. DOI: http://dx.doi.org/10.4108/ICST.INFOSCALE2008.3486 (Cited on page 243.)

Amineh Amini, Ying Wah Teh, and Hadi Saboohi. 2014. On Density-Based Data Streams Clustering Algorithms: A Survey. *J. Comput. Sci. Technol.* 29, 1 (2014), 116–141. DOI: http://dx.doi.org/10.1007/s11390-014-1416-y (Cited on page 33.)

Luc Anselin. 1989. What is Special About Spatial Data? Alternative Perspectives on Spatial Data Analysis (89-4). (1989). https://escholarship.org/uc/item/3ph5k0d4 (Cited on page 5.)

G. Antoshenkov. 1995. Byte-aligned Bitmap Compression. In *Proceedings of the Conference on Data Compression (DCC '95)*. IEEE Computer Society, Washington, DC, USA, 476. http://dl.acm.org/citation.cfm?id=874051.874730 (Cited on page 22.)

David Arthur and Sergei Vassilvitskii. 2007. k-means++: the advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*. 1027–1035. http://dl.acm.org/citation.cfm?id=1283383.1283494 (Cited on page 33.)

Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. 1998. An Optimal Algorithm for Approximate Nearest Neighbor Searching Fixed Dimensions. *J. ACM* 45, 6 (Nov. 1998), 891–923. DOI: http://dx.doi.org/10.1145/293347.293348 (Cited on page 44.)

Lakshmi Balasubramanian and M. Sugumaran. 2012. A State-of-Art in R-Tree Variants for Spatial Indexing. In *International Journal of Computer Applications (0975 – 8887) Volume 42– No.20*. 35–41. (Cited on page 47.)

Gill Barequet, Bernard Chazelle, Leonidas J. Guibas, Joseph S.B. Mitchell, and Ayellet Tal. 1996. BOXTREE: A Hierarchical Rep-

resentation for Surfaces in 3D. *Computer Graphics Forum* (1996). DOI: http://dx.doi.org/10.1111/1467-8659.1530387 (Cited on page 25.)

Payam Barnaghi, Jeremy Tandy, and Linda van den Brink. 2017. *Spatial Data on the Web Best Practices (Work in Progress: 28. September 2017)*. W3C note. W3C. https://www.w3.org/TR/2017/NOTE-sdw-bp-20170928/. (Cited on pages 66, 67, and 68.)

Rudolf Bayer and Edward M. McCreight. 1972. Organization and Maintenance of Large Ordered Indices. *Acta Inf.* 1 (1972), 173–189. DOI: http://dx.doi.org/10.1007/BF00288683 (Cited on page 38.)

Bruno Becker, Paolo G Franciosa, Stephan Gschwind, Thomas Ohler, Gerald Thiemt, and Peter Widmayer. 1991. *An Optimal Algorithm for Approximating a Set of Rectangles by Two Minimum Area Rectangles*. Technical Report. 13–25 pages. (Cited on pages 46, 75, and 283.)

Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. 1990. The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data (SIGMOD '90)*. ACM, New York, NY, USA, 322–331. DOI: http://dx.doi.org/10.1145/93597.98741 (Cited on pages 46, 281, 283, 303, and 441.)

Norbert Beckmann and Bernhard Seeger. 2009. A Revised R*-tree in Comparison with Related Index Structures. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data (SIGMOD '09)*. ACM, New York, NY, USA, 799–812. DOI: http://dx.doi.org/10.1145/1559845.1559929 (Cited on pages 46 and 303.)

Timothy Bell, Ian H. Witten, and John G. Cleary. 1989. Modeling for text compression. *ACM Computing Surveys (CSUR)* 21, 4 (1989), 557–591. (Cited on page 22.)

Jon Louis Bentley. 1975. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM* 18, 9 (Sept. 1975), 509–517. http://doi.acm.org/10.1145/361002.361007 (Cited on page 42.)

Jon Louis Bentley and Jerome H. Friedman. 1979. Data Structures for Range Searching. *ACM Comput. Surv.* 11, 4 (Dec. 1979), 397–409. http://doi.acm.org/10.1145/356789.356797 (Cited on page 42.)

Stefan Berchtold, Christian Böhm, and Hans-Peter Kriegel. 1998. The Pyramid-Technique: Towards Breaking the Curse of Dimensionality. In *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*. 142–153. DOI: http://dx.doi.org/10.1145/276304.276318 (Cited on page 49.)

Stefan Berchtold, Daniel A. Keim, and Hans-Peter Kriegel. 1996. The X-tree: An Index Structure for High-Dimensional Data. In *Proceedings of the 22th International Conference on Very Large Data Bases (VLDB '96)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 28–39. http://dl.acm.org/citation.cfm?id=645922.673502 (Cited on page 46.)

Pavel Berkhin. 2006. A Survey of Clustering Data Mining Techniques. In *Grouping Multidimensional Data - Recent Advances in Clustering*. 25–71. DOI: http://dx.doi.org/10.1007/3-540-28349-8_2 (Cited on pages 33 and 34.)

Piotr Berman and Bhaskar DasGupta. 1997. Complexities of Efficient Solutions of Rectilinear Polygon Cover Problems. *Algorithmica* 17, 4 (1997), 331–356. DOI: http://dx.doi.org/10.1007/BF02523677 (Cited on pages 28 and 29.)

Marshall Bern, David Eppstein, and John Gilbert. 1994. Provably good mesh generation. *J. Comput. System Sci.* 48, 3 (1994), 384–409. DOI: http://dx.doi.org/10.1016/S0022-0000(05)80059-5 (Cited on page 35.)

Kai Beskers and Johannes Fischer. 2014. High-Order Entropy Compressed Bit Vectors with Rank/Select. *Algorithms* 7, 4 (2014), 608–620. DOI: http://dx.doi.org/10.3390/a7040608 (Cited on page 119.)

B. Bhatta. 2011. *Remote Sensing and GIS*. OUP India. https://books.google.de/books?id=D3sKaAEACAAJ (Cited on page 5.)

Daniel Blank. 2015. *Resource Description and Selection for Similarity Search in Metric Spaces*. Ph.D. Dissertation. University of Bamberg. https://opus4.kobv.de/opus4-bamberg/frontdoor/index/index/docId/26046 (Cited on pages 10, 12, 13, 14, 56, 57, 58, 78, 103, and 173.)

Daniel Blank and Andreas Henrich. 2012. Describing and Selecting Collections of Georeferenced Media Items in Peer-to-Peer Information Retrieval Systems. In *Discovery of Geospatial Resources: Methodologies, Technologies, and Emergent Applications*. Information Science Reference, 1–20. https://books.google.de/books?id=Ml_PygAACAAJ (Cited on pages 13 and 62.)

Daniel Blank, Andreas Henrich, and Stefan Kufer. 2016. Using Summaries to Search and Visualize Distributed Resources Addressing Spatial and Multimedia Features. *Datenbank-Spektrum* 16, 1 (2016), 67–76. DOI: http://dx.doi.org/10.1007/s13222-015-0210-5 (Cited on pages 71 and 97.)

Hans H. Bock. 1996. Probabilistic models in cluster analysis. *Computational Statistics & Data Analysis* 23 (1996), 5–28. (Cited on page 34.)

Pedja Bogdanovich and Hanan Samet. 1999. The ATREE: A Data Structure to Support Very Large Scientific Databases. In *Selected Papers from the International Workshop on Integrated Spatial Databases, Digital Images and GIS (ISD '99)*. Springer-Verlag, London, UK, UK, 235–248. http://dl.acm.org/citation.cfm?id=648004.743264 (Cited on page 45.)

Christian Böhm. 2008. *Pyramid Technique*. Springer US, Boston, MA, 929–930. DOI: http://dx.doi.org/10.1007/978-0-387-35973-1_1050 (Cited on page 49.)

Christian Böhm, Stefan Berchtold, and Daniel A. Keim. 2001. Searching in High-dimensional Spaces: Index Structures for Improving the Per-

formance of Multimedia Databases. *ACM Comput. Surv.* 33, 3 (Sept. 2001), 322–373. DOI: http://dx.doi.org/10.1145/502807.502809 (Cited on pages 36, 40, 43, 45, 46, 48, 49, 50, 51, 52, 100, 123, and 457.)

Ewout Bongers and Johan Pouwelse. 2015. A survey of P2P multidimensional indexing structures. *CoRR* abs/1507.05501 (2015). http://arxiv.org/abs/1507.05501 (Cited on page 54.)

Stephan Börzsönyi, Donald Kossmann, and Konrad Stocker. 2001. The Skyline Operator. In *Proc. of the 17th Int. Conf. on Data Engineering*. IEEE Computer Society, Washington, DC, USA, 421–430. (Cited on pages 110 and 111.)

Steven Brakman and Harry Garretsen. 2009. *Trade and Geography: Paul Krugman and the 2008 Nobel Prize in Economics*. CESifo Working Paper Series 2528. CESifo Group Munich. https://EconPapers.repec.org/RePEc:ces:ceswps:_2528 (Cited on page 4.)

M. Burrows and D. J. Wheeler. 1994. *A block-sorting lossless data compression algorithm*. Technical Report. (Cited on page 22.)

Benjamin Bustos, Gonzalo Navarro, and Edgar Chávez. 2003. Pivot Selection Techniques for Proximity Searching in Metric Spaces. *Pattern Recognition Letters* 24, 14 (2003), 2357–2366. (Cited on page 125.)

Min Cai, Martin R. Frank, Jinbo Chen, and Pedro A. Szekely. 2004. MAAN: A Multi-Attribute Addressable Network for Grid Information Services. *J. Grid Comput.* 2, 1 (2004), 3–14. DOI: http://dx.doi.org/10.1007/s10723-004-1184-y (Cited on page 54.)

D. R. Caldwell. 2005. Unlocking the Mysteries of the Bounding Box. A, 2 (Aug. 2005), 1–20. http://www.sunysb.edu/libmap/coordinates/seriesa/no2/a2.pdf (Cited on page 73.)

T. Caliński and J. Harabasz. 1974. A dendrite method for cluster analysis. *Communications in Statistics-Simulation and Computation* 3, 1 (1974), 1–27. (Cited on page 34.)

Jamie Callan. 2000. *Distributed Information Retrieval*. Kluwer Academic Publishers, Ir 5, 127–150. (Cited on pages 12 and 13.)

Kaushik Chakrabarti and Sharad Mehrotra. 1999. The Hybrid Tree: An Index Structure for High Dimensional Feature Spaces. In *Proceedings of the 15th International Conference on Data Engineering, Sydney, Austrialia, March 23-26, 1999*. 440–447. DOI: http://dx.doi.org/10.1109/ICDE.1999.754960 (Cited on pages 20 and 42.)

Chee Yong Chan and Yannis E. Ioannidis. 1999. An Efficient Bitmap Encoding Scheme for Selection Queries. In *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA*. 215–226. DOI: http://dx.doi.org/10.1145/304182.304201 (Cited on page 23.)

Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José Luis Marroquín. 2001. Searching in Metric Spaces. *ACM Comput. Surv.* 33,

3 (Sept. 2001), 273–321. DOI : http://dx.doi.org/10.1145/502807.502808 (Cited on page 58.)

Paolo Ciaccia and Marco Patella. 2000. PAC Nearest Neighbor Queries: Approximate and Controlled Search in High-Dimensional and Metric Spaces. In *Proceedings of the 16th International Conference on Data Engineering, San Diego, California, USA, February 28 - March 3, 2000*. 244–255. DOI : http://dx.doi.org/10.1109/ICDE.2000.839417 (Cited on page 243.)

Douglas Comer. 1979. The Ubiquitous B-Tree. *ACM Comput. Surv.* 11, 2 (1979), 121–137. DOI : http://dx.doi.org/10.1145/356770.356776 (Cited on page 279.)

Francisco Matias Cuenca-Acuna, Christopher Peery, Richard P. Martin, and Thu D. Nguyen. 2003. PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities. In *12th International Symposium on High-Performance Distributed Computing (HPDC-12 2003), 22-24 June 2003, Seattle, WA, USA*. 236–249. DOI : http://dx.doi.org/10.1109/HPDC.2003.1210033 (Cited on page 112.)

Peter H. Dana. 1995. Geodetic Datum Overview. (1995). https://www.colorado.edu/geography/gcraft/notes/datum/datum.html#GeoDat . Accessed: 04.07.2018. (Cited on page 62.)

Tran Khanh Dang, Josef Küng, and Roland Wagner. 2001. The SH-tree: A Super Hybrid Index Structure for Multidimensional Data. In *Database and Expert Systems Applications, 12th International Conference, DEXA 2001 Munich, Germany, September 3-5, 2001, Proceedings*. 340–349. DOI : http://dx.doi.org/10.1007/3-540-44759-8_34 (Cited on pages 48 and 85.)

Arup Dasgupta. 2013. Big data: The future is in analytics. *Geospatial World, April* (2013). https://www.geospatialworld.net/article/big-data-the-future-is-in-analytics/ (Cited on page 5.)

Renato Cordeiro de Amorim and Christian Hennig. 2016. Recovering the number of clusters in data sets with noise features using feature rescaling factors. *CoRR* abs/1602.06989 (2016). http://arxiv.org/abs/1602.06989 (Cited on page 34.)

Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. 2008. *Computational geometry: algorithms and applications, 3rd Edition*. Springer. http://www.worldcat.org/oclc/227584184 (Cited on pages 27, 34, and 35.)

Stefan de Konink and Radim Baca. 06.04.2010. R-tree.svg. (06.04.2010). https://commons.wikimedia.org/wiki/File:R-tree.svg (Cited on page 280.)

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the*

royal statistical society. Series B (methodological) (1977), 1–38. (Cited on page 34.)

P. Deutsch. 1996. DEFLATE Compressed Data Format Specification version 1.3. RFC 1951 (Informational). (May 1996). http://www.ietf.org/rfc/rfc1951.txt (Cited on page 120.)

Rodolphe Devillers, Alfred Stein, Yvan Bédard, Nicholas Chrisman, Peter Fisher, and Wenzhong Shi. 2010. Thirty Years of Research on Spatial Data Quality: Achievements, Failures, and Opportunities. *Trans. GIS* 14, 4 (2010), 387–400. DOI: http://dx.doi.org/10.1111/j.1467-9671.2010.01212.x (Cited on pages 66, 67, and 69.)

Simena Dinas and José M. Bañón. 2015. A literature review of bounding volumes hierarchy focused on collision detection. *Ingeniería y competitividad* 17, 1 (2015), 49–62. (Cited on page 25.)

Goran M. Djuknic and Robert E. Richton. 2001. Geolocation and Assisted GPS. *Computer* 34, 2 (Feb. 2001), 123–125. DOI: http://dx.doi.org/10.1109/2.901174 (Cited on page 116.)

J. C. Dunn. 1973. A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *Journal of Cybernetics* 3, 3 (1973), 32–57. DOI: http://dx.doi.org/10.1080/01969727308546046 (Cited on page 34.)

Herbert Edelsbrunner. 1998. Shape Reconstruction with Delaunay Complex. In *LATIN '98: Theoretical Informatics, Third Latin American Symposium, Campinas, Brazil, April, 20-24, 1998, Proceedings*. 119–132. DOI: http://dx.doi.org/10.1007/BFb0054315 (Cited on page 26.)

Herbert Edelsbrunner. 2010. Alpha shapes—a survey. *Tessellations in the Sciences; Virtues, Techniques and Applications of Geometric Tilings.* 27 (2010). (Cited on page 26.)

Herbert Edelsbrunner, David G. Kirkpatrick, and Raimund Seidel. 1983. On the shape of a set of points in the plane. *IEEE Trans. Information Theory* 29, 4 (1983), 551–558. DOI: http://dx.doi.org/10.1109/TIT.1983.1056714 (Cited on page 26.)

Herbert Edelsbrunner and Emo Welzl. 1986. Halfplanar Range Search in Linear Space and O(nˆ(0.695)) Query Time. *Inf. Process. Lett.* 23, 6 (1986), 289–293. DOI: http://dx.doi.org/10.1016/0020-0190(86)90088-8 (Cited on page 44.)

Martin Eisenhardt, Wolfgang Müller, Soufyane El Allali, Andreas Henrich, and Daniel Blank. 2006. Clustering-Based Source Selection for Efficient Image Retrieval in Peer-to-Peer Networks. *2006 8th IEEE International Symposium on Multimedia* (2006), 823–830. DOI: http://dx.doi.org/doi.ieeecomputersociety.org/10.1109/ISM.2006.47 (Cited on page 103.)

Ahmed Eldawy and Mohamed F. Mokbel. 2016. The Era of Big Spatial Data: A Survey. *Foundations and Trends in Databases* 6, 3-4 (2016), 163–273. DOI: http://dx.doi.org/10.1561/1900000054 (Cited on page 6.)

Felix Engl. 2018. *Optimierungsstrategien zu einem R-Baum mit zusammenfassungsbasierten Knotenbeschreibungen für Update Operationen und Anfragen*. Master's thesis. University of Bamberg, An der Weberei 5, 96052 Bamberg. (Cited on pages 277 and 437.)

David Eppstein. 2009. Graph-Theoretic Solutions to Computational Geometry Problems. In *Graph-Theoretic Concepts in Computer Science, 35th International Workshop, WG 2009, Montpellier, France, June 24-26, 2009. Revised Papers*. 1–16. DOI: http://dx.doi.org/10.1007/978-3-642-11409-0_1 (Cited on page 28.)

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*. 226–231. http://www.aaai.org/Library/KDD/1996/kdd96-037.php (Cited on page 34.)

Andrea Esuli. 2009. MiPai: Using the PP-Index to Build an Efficient and Scalable Similarity Search System. In *Second International Workshop on Similarity Search and Applications, SISAP 2009, 29-30 August 2009, Prague, Czech Republic*. 146–148. DOI: http://dx.doi.org/10.1109/SISAP.2009.14 (Cited on page 35.)

Ronald Fagin, Jürg Nievergelt, Nicholas Pippenger, and H. Raymond Strong. 1979. Extendible Hashing—A Fast Access Method for Dynamic Files. *ACM Trans. Database Syst.* 4, 3 (Sept. 1979), 315–344. DOI: http://dx.doi.org/10.1145/320083.320092 (Cited on page 38.)

Christos Faloutsos. 1986. Multiattribute Hashing Using Gray Codes. In *Proceedings of the 1986 ACM SIGMOD International Conference on Management of Data (SIGMOD '86)*. ACM, New York, NY, USA, 227–238. DOI: http://dx.doi.org/10.1145/16894.16877 (Cited on page 40.)

C. Faloutsos and S. Roseman. 1989. Fractals for Secondary Key Retrieval. In *Proceedings of the Eighth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '89)*. ACM, New York, NY, USA, 247–252. DOI: http://dx.doi.org/10.1145/73721.73746 (Cited on page 40.)

Leonard A. Ferrari, P. V. Sankar, and Jack Sklansky. 1984. Minimal rectangular partitions of digitized blobs. *Computer Vision, Graphics, and Image Processing* 28, 1 (1984), 58–71. DOI: http://dx.doi.org/10.1016/0734-189X(84)90139-7 (Cited on page 27.)

FGDC. 1998. Geospatial Positioning Accuracy Standards – Part 3: National Standard for Spatial Data Accuracy. Federal Geographic Data

Committee (FGDC). (1998). https://www.fgdc.gov/standards/projects/accuracy/part3/chapter3 (Cited on page 68.)

R. A. Finkel and J. L. Bentley. 1974. Quad trees A data structure for retrieval on composite keys. *Acta Informatica* 4, 1 (1974), 1–9. DOI: http://dx.doi.org/10.1007/BF00288933 (Cited on page 41.)

Jan Flusser. 2000. Refined moment calculation using image block representation. *IEEE Trans. Image Processing* 9, 11 (2000), 1977–1978. DOI: http://dx.doi.org/10.1109/83.877219 (Cited on page 29.)

A.S. Fotheringham, S. Fotheringham, C. Brunsdon, and M. Charlton. 2000. *Quantitative Geography: Perspectives on Spatial Data Analysis*. SAGE Publications. https://books.google.de/books?id=sqLtks3TjeEC (Cited on page 3.)

Deborah S. Franzblau. 1989. Performance Guarantees on a Sweep-Line Heuristic for Covering Rectilinear Polygons with Rectangles. *SIAM J. Discrete Math.* 2, 3 (1989), 307–321. DOI: http://dx.doi.org/10.1137/0402027 (Cited on page 28.)

H. Freeman and R. Shapira. 1975. Determining the Minimum-area Encasing Rectangle for an Arbitrary Closed Curve. *Commun. ACM* 18, 7 (July 1975), 409–413. DOI: http://dx.doi.org/10.1145/360881.360919 (Cited on page 25.)

Michael Freeston. 1987. The BANG File: A New Kind of Grid File. In *Proceedings of the Association for Computing Machinery Special Interest Group on Management of Data 1987 Annual Conference, San Francisco, California, May 27-29, 1987*. 260–269. DOI: http://dx.doi.org/10.1145/38713.38743 (Cited on page 44.)

Henry Fuchs, Zvi M. Kedem, and Bruce F. Naylor. 1980. On Visible Surface Generation by A Priori Tree Structures. *SIGGRAPH Comput. Graph.* 14, 3 (July 1980), 124–133. DOI: http://dx.doi.org/10.1145/965105.807481 (Cited on page 43.)

Volker Gaede and Oliver Günther. 1998. Multidimensional Access Methods. *ACM Comput. Surv.* 30, 2 (1998), 170–231. (Cited on pages 5, 36, 37, 38, 39, 40, 41, 42, 43, 44, 47, 50, 51, and 62.)

Prasanna Ganesan, Beverly Yang, and Hector Garcia-Molina. 2004. One Torus to Rule Them All: Multidimensional Queries in P2P Systems. In *Proceedings of the Seventh International Workshop on the Web and Databases, WebDB 2004, June 17-18, 2004, Maison de la Chimie, Paris, France, Colocated with ACM SIGMOD/PODS 2004*. 19–24. http://webdb2004.cs.columbia.edu/papers/2-1.pdf (Cited on pages 54 and 55.)

Irene Gargantini. 1982. An Effective Way to Represent Quadtrees. *Commun. ACM* 25, 12 (Dec. 1982), 905–910. (Cited on pages 54 and 65.)

Jacob E. Goodman, Joseph O'Rourke, and Csaba D. Tóth (Eds.). 2017. *Handbook of Discrete and Computational Geometry, Third Edition*. Chapman and Hall/CRC. http://www.csun.edu/~ctoth/Handbook/HDCG3.html (Cited on page 35.)

Joshua Grass and Shlomo Zilberstein. 1996. Anytime Algorithm Development Tools. *SIGART Bulletin* 7, 2 (1996), 20–27. DOI: http://dx.doi.org/10.1145/242587.242592 (Cited on page 243.)

Diane Greene. 1989. An Implementation and Performance Analysis of Spatial Data Access Methods. IEEE Computer Society, Los Angeles. (Cited on page 46.)

Oliver Günther. 1989. The Design of the Cell Tree: An Object-Oriented Index Structure for Geometric Databases. In *Proceedings of the Fifth International Conference on Data Engineering*. IEEE Computer Society, Washington, DC, USA, 598–605. http://dl.acm.org/citation.cfm?id=645474.653995 (Cited on page 47.)

Stephen C. Guptill, J.L.A. MORRISON, and International Cartographic Association. 1995. *Elements of Spatial Data Quality*. Elsevier Science Limited. https://books.google.de/books?id=i-1_AAAAMAAJ (Cited on page 67.)

Mordechai Guri, Boris Zadov, Eran Atias, and Yuval Elovici. 2017. LED-it-GO: Leaking (a lot of) Data from Air-Gapped Computers via the (small) Hard Drive LED. *CoRR* abs/1702.06715 (2017). http://arxiv.org/abs/1702.06715 (Cited on page 299.)

Antonin Guttman. 1984. R-trees: A Dynamic Index Structure for Spatial Searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data (SIGMOD '84)*. ACM, New York, NY, USA, 47–57. DOI: http://dx.doi.org/10.1145/602259.602266 (Cited on pages XIV, 8, 45, and 282.)

Gheorghi Guzun, Guadalupe Canahuate, David Chiu, and Jason Sawin. 2014. A tunable compression framework for bitmap indices. In *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*. 484–495. DOI: http://dx.doi.org/10.1109/ICDE.2014.6816675 (Cited on pages 22 and 23.)

Geospatial World Magazine GW. 2016. How big data saves lives. *Geospatial World, March 9* (2016). https://www.geospatialworld.net/article/how-big-data-saves-lives (Cited on page 4.)

Sariel Har-Peled. 2011. *Geometric Approximation Algorithms*. American Mathematical Society, Boston, MA, USA. (Cited on page 20.)

John A. Hartigan. 1975. *Clustering Algorithms* (99th ed.). John Wiley & Sons, Inc., New York, NY, USA. (Cited on page 34.)

Herman Johannes Haverkort. 2004. *Results on geometric networks and data structures*. Ph.D. Dissertation. (Cited on pages 25 and 26.)

Laura Heinrich-Litan and Marco E. Lübbecke. 2006. Rectangle covers revisited computationally. *ACM Journal of Experimental Algorithmics* 11 (2006). DOI: http://dx.doi.org/10.1145/1187436.1216583 (Cited on page 29.)

Andreas Henrich. 1990. *Der LSD-Baum - eine mehrdimensionale Zugriffsstruktur und ihre Einsatzmöglichkeiten in Datenbanksystemen*. Ph.D. Dissertation. University of Hagen, Germany. http://d-nb.info/910296634 (Cited on pages 43 and 348.)

Andreas Henrich. 1998. The LSDh-tree: An Access Structure for Feature Vectors. *Proc. of 14th Int. Conf. on Data Engineering : February 23 - 27, 1998 Orlando, Florida.* 0 (1998), 362–369. DOI : http://dx.doi.org/10.1109/ICDE.1998.655799 (Cited on page 43.)

Andreas Henrich and Daniel Blank. 2010. Description and Selection of Media Archives for Geographic Nearest Neighbor Queries in P2P Networks. *Proceedings of the ECIR2010 workshop on information access for personal media archives* (2010), 22–29. http://doras.dcu.ie/15373/ (Cited on pages 8, 9, 13, 71, 78, 88, 97, 115, 125, 233, 270, and 457.)

Andreas Henrich, Hans-Werner Six, and Peter Widmayer. 1989. The LSD Tree: Spatial Access to Multidimensional and Non-point Objects. In *Proceedings of the 15th International Conference on Very Large Data Bases (VLDB '89)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 45–53. http://dl.acm.org/citation.cfm?id=88830.88835 (Cited on page 43.)

John Hershberger and Subhash Suri. 2004. Adaptive Sampling for Geometric Problems over Data Streams. In *Proceedings of the Twenty-third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '04)*. ACM, New York, NY, USA, 252–262. DOI : http://dx.doi.org/10.1145/1055558.1055595 (Cited on page 26.)

Magnus Lie Hetland. 2009. *The Basic Principles of Metric Indexing*. Springer Berlin Heidelberg, 199–232. DOI : http://dx.doi.org/10.1007/978-3-642-03625-5_9 (Cited on pages 56, 57, and 58.)

Magnus Lie Hetland. 2015. Ptolemaic indexing. *JoCG* 6, 1 (2015), 165–184. http://jocg.org/index.php/jocg/article/view/42 (Cited on pages 56, 57, and 58.)

Magnus Lie Hetland, Tomás Skopal, Jakub Lokoc, and Christian Beecks. 2013. Ptolemaic access methods: Challenging the reign of the metric space model. *Inf. Syst.* 38, 7 (2013), 989–1006. http://dx.doi.org/10.1016/j.is.2012.05.011 (Cited on pages 56 and 58.)

Klaus Hinrichs. 1985. Implementation of the grid file: Design concepts and experience. *BIT Numerical Mathematics* 25, 4 (1985), 569–592. DOI : http://dx.doi.org/10.1007/BF01936137 (Cited on page 39.)

Gísli R. Hjaltason and Hanan Samet. 1995. Ranking in Spatial Databases. In *Advances in Spatial Databases, 4th International Symposium, SSD'95, Portland, Maine, USA, August 6-9, 1995, Proceedings*. 83–95. DOI : http://dx.doi.org/10.1007/3-540-60159-7_6 (Cited on page 51.)

Cyril Höschl IV and Jan Flusser. 2016. Decomposition of 3D Binary Objects into Rectangular Blocks. In *2016 International Conference*

on Digital Image Computing: Techniques and Applications, DICTA 2016, Gold Coast, Australia, November 30 - December 2, 2016. 1–8. DOI: http://dx.doi.org/10.1109/DICTA.2016.7797028 (Cited on pages 27 and 28.)

David A. Huffman. 1952. A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the Institute of Radio Engineers* 40, 9 (September 1952), 1098–1101. (Cited on page 21.)

Andreas Hutflesz, Hans-Werner Six, and Peter Widmayer. 1988a. Globally Order Preserving Multidimensional Linear Hashing. In *Proceedings of the Fourth International Conference on Data Engineering, February 1-5, 1988, Los Angeles, California, USA*. 572–579. DOI: http://dx.doi.org/10.1109/ICDE.1988.105505 (Cited on page 39.)

Andreas Hutflesz, Hans-Werner Six, and Peter Widmayer. 1988b. Twin Grid Files: Space Optimizing Access Schemes. In *Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data (SIGMOD '88)*. ACM, New York, NY, USA, 183–190. DOI: http://dx.doi.org/10.1145/50202.50222 (Cited on page 39.)

Sangyong Hwang, Keunjoo Kwon, Sang Kyun Cha, and Byung Suk Lee. 2003. Performance Evaluation of Main-Memory R-tree Variants. In *Advances in Spatial and Temporal Databases, 8th International Symposium, SSTD 2003, Santorini Island, Greece, July 24-27, 2003, Proceedings*. 10–27. DOI: http://dx.doi.org/10.1007/978-3-540-45072-6_2 (Cited on pages 46 and 446.)

Hiroshi Imai and Takao Asano. 1986. Efficient Algorithms for Geometric Graph Search Problems. *SIAM J. Comput.* 15, 2 (1986), 478–494. DOI: http://dx.doi.org/10.1137/0215033 (Cited on pages 28 and 86.)

Martin Isenburg, Peter Lindstrom, and Jack Snoeyink. 2004. Lossless compression of floating-point geometry. *Computer-Aided Design and Applications* 1, 1-4 (2004), 495–502. (Cited on page 23.)

H. V. Jagadish. 1990. Spatial Search with Polyhedra. In *Proceedings of the Sixth International Conference on Data Engineering*. IEEE Computer Society, Washington, DC, USA, 311–319. http://dl.acm.org/citation.cfm?id=645475.757681 (Cited on page 47.)

Ray A. Jarvis. 1973. On the Identification of the Convex Hull of a Finite Set of Points in the Plane. *Inf. Process. Lett.* 2, 1 (1973), 18–21. http://dblp.uni-trier.de/db/journals/ipl/ipl2.html#Jarvis73 (Cited on page 26.)

Myeong-Hun Jeong, Yaping Cai, Clair J. Sullivan, and Shaowen Wang. 2016. Data depth based clustering analysis. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS 2016, Burlingame, California, USA, October 31 - November 3, 2016*. 29:1–29:10. http://doi.acm.org/10.1145/2996913.2996984 (Cited on page 33.)

Ibrahim Kamel and Christos Faloutsos. 1993. On Packing R-trees. In *CIKM 93, Proceedings of the Second International Conference on In-*

formation and Knowledge Management, Washington, DC, USA, November 1-5, 1993. 490–499. DOI: http://dx.doi.org/10.1145/170088.170403 (Cited on pages 383, 384, and 467.)

Ibrahim Kamel and Christos Faloutsos. 1994. Hilbert R-tree: An Improved R-tree Using Fractals. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB '94)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 500–509. http://dl.acm.org/citation.cfm?id=645920.673001 (Cited on page 46.)

Norio Katayama and Shin'ichi Satoh. 1997. The SR-tree: An Index Structure for High-dimensional Nearest Neighbor Queries. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data (SIGMOD '97)*. ACM, New York, NY, USA, 369–380. DOI: http://dx.doi.org/10.1145/253260.253347 (Cited on page 48.)

Eiji Kawaguchi and Tsutomu Endo. 1980. On a Method of Binary-Picture Representation and Its Application to Data Compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence* vol. 2 (1980), 27–35. DOI: http://dx.doi.org/doi.ieeecomputersociety.org/10.1109/TPAMI.1980.4766967 (Cited on pages 54 and 65.)

Gershon Kedem. 1982. The quad-CIF Tree: A Data Structure for Hierarchical On-line Algorithms. In *Proceedings of the 19th Design Automation Conference (DAC '82)*. IEEE Press, Piscataway, NJ, USA, 352–357. http://dl.acm.org/citation.cfm?id=800263.809229 (Cited on page 41.)

J. Mark Keil. 2000. Polygon decomposition. *Handbook of Computational Geometry* 2 (2000), 491–518. (Cited on pages 27, 28, and 29.)

Allen Klinger. 1971. Patterns and Search Statistics. In *Optimizing Methods in Statistics*, Jagdish S. Rustagi (Ed.). Academic Press, 303–337. DOI: http://dx.doi.org/10.1016/B978-0-12-604550-5.50019-5 (Cited on page 41.)

K. Knowlton. 1980. Progressive transmission of grey scale and binary images by simple, efficient and lossless encoding schemes. In *Porceedings of the IEEE 68(7)*. 885–896. (Cited on page 42.)

Nick Koudas. 2000. Space Efficient Bitmap Indexing. In *Proceedings of the 2000 ACM CIKM International Conference on Information and Knowledge Management, McLean, VA, USA, November 6-11, 2000*. 194–201. DOI: http://dx.doi.org/10.1145/354756.354819 (Cited on page 23.)

Hans-Peter Kriegel, Peer Kröger, Peter Kunath, and Matthias Renz. 2007. Generalizing the Optimality of Multi-step *k*-Nearest Neighbor Query Processing. In *Advances in Spatial and Temporal Databases, 10th International Symposium, SSTD 2007, Boston, MA, USA, July 16-18, 2007, Proceedings*. 75–92. DOI: http://dx.doi.org/10.1007/978-3-540-73540-3_5 (Cited on pages 99 and 105.)

Hans-Peter Kriegel and Bernhard Seeger. 1987. Multidimensional Dynamic Quantile Hashing is Very Efficient for Non-Uniform Record Distributions. In *Proceedings of the Third International Conference on Data Engineering, February 3-5, 1987, Los Angeles, California, USA*. 10–17. DOI: http://dx.doi.org/10.1109/ICDE.1987.7272349 (Cited on page 39.)

Hans-Peter Kriegel and Bernhard Seeger. 1988. PLOP-Hashing: A Grid File without Directory. In *Proceedings of the Fourth International Conference on Data Engineering, February 1-5, 1988, Los Angeles, California, USA*. 369–376. DOI: http://dx.doi.org/10.1109/ICDE.1988.105439 (Cited on page 39.)

Hans-Peter Kriegel and Bernhard Seeger. 1986. *Multidimensional order preserving linear hashing with partial expansions*. Springer Berlin Heidelberg, Berlin, Heidelberg, 203–220. DOI: http://dx.doi.org/10.1007/3-540-17187-8_38 (Cited on page 39.)

Stefan Kufer. 2012. *Techniken der Ressourcenbeschreibung und Auswahl am Beispiel des verteilten geographischen Information Retrieval*. Master's thesis. University of Bamberg, An der Weberei 5, 96052 Bamberg. https://www.uni-bamberg.de/fileadmin/uni/fakultaeten/wiai_lehrstuehle/medieninformatik/Dateien/Publikationen/2012/kufer_2012_techniken.pdf (Cited on pages 71, 73, 87, 88, 97, 98, 115, and 125.)

Stefan Kufer, Daniel Blank, and Andreas Henrich. 2012. Techniken der Ressourcenbeschreibung und -auswahl für das geographische Information Retrieval. In *Proc. of the IR Workshop at LWA 2012*. 1–8. (Cited on pages 13, 71, 73, 97, 98, 115, and 125.)

Stefan Kufer, Daniel Blank, and Andreas Henrich. 2013. *Using Hybrid Techniques for Resource Description and Selection in the Context of Distributed Geographic Information Retrieval*. Springer Berlin Heidelberg, 330–347. DOI: http://dx.doi.org/10.1007/978-3-642-40235-7_19 (Cited on pages 71, 97, 110, and 125.)

Stefan Kufer and Andreas Henrich. 2014. Hybrid Quantized Resource Descriptions for Geospatial Source Selection. In *Proc. of the 4th Int. Workshop on Location and the Web (LocWeb '14)*. ACM, New York, NY, USA, 17–24. DOI: http://dx.doi.org/10.1145/2663713.2664428 (Cited on pages 71, 97, and 110.)

Stefan Kufer and Andreas Henrich. 2017. Quadtree-based Resource Description Techniques for Spatial Data in Distributed Databases. In *Datenbanksysteme für Business, Technologie und Web (BTW 2017), 17. Fachtagung des GI-Fachbereichs „Datenbanken und Informationssysteme" (DBIS), 6.-10. März 2017, Stuttgart, Germany, Proceedings*. 445–464. (Cited on pages 71 and 97.)

V. S. Anil Kumar and H. Ramesh. 2003. Covering Rectilinear Polygons with Axis-Parallel Rectangles. *SIAM J. Comput.* 32, 6 (2003), 1509–1541. DOI: http://dx.doi.org/10.1137/S0097539799358835 (Cited on page 28.)

Elmar Langetepe and Gabriel Zachmann. 2006. *Geometric data structures for computer graphics*. A K Peters. (Cited on pages 24 and 25.)

Eric Larsen, Stefan Gottschalk, Ming C. Lin, and Dinesh Manocha. 1999. *Fast Proximity Queries with Swept Sphere Volumes*. Technical Report. Department of Computer Science, UNC Chapel Hill. (Cited on page 25.)

Dik Lun Lee, Baihua Zheng, Wang-Chien Lee, and Jianliang Xu. 2004. The D-Tree: An Index Structure for Planar Point Queries in Location-Based Wireless Services. *IEEE Transactions on Knowledge & Data Engineering* 16 (2004), 1526–1542. DOI: http://dx.doi.org/doi.ieeecomputersociety.org/10.1109/TKDE.2004.97 (Cited on page 48.)

Jinwon Lee, Hyonik Lee, Seungwoo Kang, Su Myeon Kim, and Junehwa Song. 2007. CISS: An efficient object clustering framework for DHT-based peer-to-peer applications. *Computer Networks* 51, 4 (2007), 1072–1094. DOI: http://dx.doi.org/10.1016/j.comnet.2006.07.005 (Cited on page 54.)

Agnieszka Leszczynski and Jeremy Crampton. 2016. Introduction: Spatial Big Data and everyday life. *Big Data & Society* 3, 2 (2016), 2053951716661366. DOI: http://dx.doi.org/10.1177/2053951716661366 (Cited on page 6.)

King-Ip Lin, H. V. Jagadish, and Christos Faloutsos. 1994. The TV-tree: An index structure for high-dimensional data. *The VLDB Journal* 3, 4 (1994), 517–542. DOI: http://dx.doi.org/10.1007/BF01231606 (Cited on page 49.)

Tsong-Wuu Lin. 1997. Set Operations on Constant Bit-length Linear Quadtrees. *Pattern Recogn.* 30, 7 (July 1997), 1239–1249. (Cited on pages 54 and 65.)

Peter Lindstrom. 2014. Fixed-Rate Compressed Floating-Point Arrays. *IEEE Trans. Vis. Comput. Graph.* 20, 12 (2014), 2674–2683. DOI: http://dx.doi.org/10.1109/TVCG.2014.2346458 (Cited on pages 23 and 24.)

Peter Lindstrom and Martin Isenburg. 2006. Fast and Efficient Compression of Floating-Point Data. *IEEE Trans. Vis. Comput. Graph.* 12, 5 (2006), 1245–1250. DOI: http://dx.doi.org/10.1109/TVCG.2006.143 (Cited on page 23.)

W. T. Liou, Chuan Yi Tang, and Richard C. T. Lee. 1991. Covering convex rectilinear polygons in linear time. *Int. J. Comput. Geometry Appl.* 1, 2 (1991), 137–185. DOI: http://dx.doi.org/10.1142/S0218195991000128 (Cited on page 28.)

Witold Litwin. 1980. Linear Hashing: A New Tool for File and Table Addressing. In *Proceedings of the Sixth International Conference on Very Large Data Bases - Volume 6 (VLDB '80)*. VLDB Endowment, 212–223. http://dl.acm.org/citation.cfm?id=1286887.1286911 (Cited on page 38.)

Bin Liu, Wang-Chien Lee, and Dik Lun Lee. 2005. Supporting Complex Multi-Dimensional Queries in P2P Systems. In *25th International Conference on Distributed Computing Systems (ICDCS 2005), 6-10 June 2005, Columbus, OH, USA*. 155–164. DOI: http://dx.doi.org/10.1109/ICDCS.2005.75 (Cited on page 55.)

Stuart P. Lloyd. 1982. Least squares quantization in PCM. *IEEE Trans. Information Theory* 28, 2 (1982), 129–136. DOI: http://dx.doi.org/10.1109/TIT.1982.1056489 (Cited on page 33.)

D. B. Lomet and B. Salzberg. 1989. A robust multi-attribute search structure. In *[1989] Proceedings. Fifth International Conference on Data Engineering*. 296–304. DOI: http://dx.doi.org/10.1109/ICDE.1989.47229 (Cited on page 45.)

Jie Lu. 2007. *Full-text Federated Search in Peer-to-peer Networks*. Ph.D. Dissertation. Pittsburgh, PA, USA. Advisor(s) Callan, James P. AAI3285253. (Cited on pages 13 and 55.)

I.R. Management Association. 2016. *Geospatial Research: Concepts, Methodologies, Tools, and Applications: Concepts, Methodologies, Tools, and Applications*. Number Bd. 1 in Premier reference source. IGI Global. https://books.google.de/books?id=R_cQDAAAQBAJ (Cited on page 67.)

Songrit Maneewongvatana and David M. Mount. 1999. It's Okay to Be Skinny, If Your Friends Are Fat. In *Center for Geometric Computing 4th Annual Workshop on Computational Geometry*. (Cited on page 42.)

Songrit Maneewongvatana and David M. Mount. 2001. The Analysis of a Probabilistic Approach to Nearest Neighbor Searching. In *Algorithms and Data Structures, 7th International Workshop, WADS 2001, Providence, RI, USA, August 8-10, 2001, Proceedings*. 276–286. DOI: http://dx.doi.org/10.1007/3-540-44634-6_26 (Cited on page 48.)

Yannis Manolopoulos, Alexandros Nanopoulos, Apostolos N. Papadopoulos, and Yannis Theodoridis. 2006. *R-Trees: Theory and Applications*. Springer. DOI: http://dx.doi.org/10.1007/978-1-84628-293-5 (Cited on pages 47, 49, 279, 282, and 283.)

Maude Manouvrier, Marta Rukoz, and Geneviève Jomier. 2002. Quadtree representations for storage and manipulation of clusters of images. *Im. Vis. Comput.* 20, 7 (2002), 513–527. (Cited on pages 53, 54, 64, and 65.)

Takashi Matsuyama, Le Viet Hao, and Makoto Nagao. 1984. A file organization for geographic information systems based on spatial proximity. *Computer Vision, Graphics, and Image Processing* 26, 3 (1984), 303–318. DOI: http://dx.doi.org/10.1016/0734-189X(84)90215-9 (Cited on page 43.)

T.H. Merrett. 1978. Multidimensional paging for efficient database querying.. In *Proc. Int'l Conference on Management of Data, Milan (1978)*. 277–290. (Cited on page 39.)

B. Mirkin. 2012. *Clustering: A Data Recovery Approach, Second Edition*. Taylor & Francis. https://books.google.de/books?id=34-q8geA3yEC (Cited on page 34.)

Anirban Mondal, Yi Lifu, and Masaru Kitsuregawa. 2004. P2PR-Tree: An R-Tree-Based Spatial Index for Peer-to-Peer Environments. In *Current Trends in Database Technology - EDBT 2004 Workshops, EDBT 2004 Workshops PhD, DataX, PIM, P2P&DB, and ClustWeb, Heraklion, Crete, Greece, March 14-18, 2004, Revised Selected Papers*. 516–525. DOI : http://dx.doi.org/10.1007/978-3-540-30192-9_51 (Cited on page 55.)

D. Nebert. 2004. Developing Spatial Data Infrastructures: The SDI Cookbook. Technical report, Global Spatial Data Infrastructure. (2004). http://www.gsdi.org/docs2004/Cookbook/cookbookV2.0.pdf (Cited on pages 66 and 67.)

Randal C. Nelson and Hanan Samet. 1986. A Consistent Hierarchical Representation for Vector Data. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '86)*. ACM, New York, NY, USA, 197–206. DOI : http://dx.doi.org/10.1145/15922.15908 (Cited on page 42.)

Jürg Nievergelt, Hans Hinterberger, and Kenneth C. Sevcik. 1984. The Grid File: An Adaptable, Symmetric Multikey File Structure. *ACM Trans. Database Syst.* 9, 1 (March 1984), 38–71. DOI : http://dx.doi.org/10.1145/348.318586 (Cited on page 38.)

Arlind Nocaj and Ulrik Brandes. 2012. Computing Voronoi Treemaps: Faster, Simpler, and Resolution-independent. *Computer Graphics Forum* (2012). DOI : http://dx.doi.org/10.1111/j.1467-8659.2012.03078.x (Cited on page 101.)

David Novak and Pavel Zezula. 2014. Performance Study of Independent Anchor Spaces for Similarity Searching. *Comput. J.* 57, 11 (2014), 1741–1755. DOI : http://dx.doi.org/10.1093/comjnl/bxt114 (Cited on page 35.)

Yutaka Ohsawa and Masao Sakauchi. 1983. The BD-Tree - A New N-Dimensional Data Structure with Highly Efficient Dynamic Characteristics.. In *IFIP Congress*. 539–544. http://dblp.uni-trier.de/db/conf/ifip/ifip83.html#OhsawaS83 (Cited on page 44.)

Yutaka Ohsawa and Masao Sakauchi. 1990. A New Tree Type Data Structure with Homogeneous Nodes Suitable for a Very Large Spatial Database. In *Proceedings of the Sixth International Conference on Data Engineering, February 5-9, 1990, Los Angeles, California, USA*. 296–303. DOI : http://dx.doi.org/10.1109/ICDE.1990.113481 (Cited on page 45.)

T. Ohtsuki. 1982. Minimum dissection of rectilinear regions. In *Proc. IEEE Int. Symp. on Circuits and Systems ISCAS'82*. 1210–1213. (Cited on page 27.)

Molly A. O'Neil and Martin Burtscher. 2011. Floating-point data compression at 75 Gb/s on a GPU. In *Proceedings of 4th Workshop on General Purpose Processing on Graphics Processing Units, GPGPU 2011, Newport Beach, CA, USA, March 5, 2011*. 7. DOI : http://dx.doi.org/10.1145/1964179.1964189 (Cited on page 24.)

Patrick E. O'Neil and Dallan Quass. 1997. Improved Query Performance with Variant Indexes. In *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*. 38–49. DOI : http://dx.doi.org/10.1145/253260.253268 (Cited on page 23.)

B.C. Ooi, K. J. McDonell, and Sacks R. Davis. 1987. Spatial kd-tree: An Indexing Mechanism for Spatial Database. *COMPSAC conf.* (1987), 433–438. (Cited on page 43.)

P. van Oosterom. 1990a. A modified binary space partitioning tree for geographic information systems. *International Journal of Geographical Information Systems* 4, 2 (1990), 133–146. DOI : http://dx.doi.org/10.1080/02693799008941535 (Cited on page 44.)

P. van Oosterom. 1990b. Reactive data structures for geographic information systems. (1990). (Cited on page 47.)

P. van Oosterom. 1999. *Spatial Access Methods*. Geographical Information Systems, Vol. 1. Chapter 27, 385–400. (Cited on pages 36, 40, 44, 47, and 73.)

J. A. Orenstein. 1982. Multidimensional Tries Used for Associative Searching. *Inf. Process. Lett.* 14, 4 (1982), 150–157. http://dblp.uni-trier.de/db/journals/ipl/ipl14.html#Orenstein82 (Cited on pages 41 and 42.)

J. A. Orenstein and T. H. Merrett. 1984. A Class of Data Structures for Associative Searching. In *Proceedings of the 3rd ACM SIGACT-SIGMOD Symposium on Principles of Database Systems (PODS '84)*. ACM, New York, NY, USA, 181–190. DOI : http://dx.doi.org/10.1145/588011.588037 (Cited on page 40.)

Heather A. Phillips. 2010. The Great Library of Alexandria? Library Philosophy and Practice 2010. (2010). http://unllib.unl.edu/LPP/phillips.htm (Cited on page 6.)

Laurent Poirrier. 2009. *An efficient space partitioning technique based on linear kd-trees for collision culling*. Technical Report. University of Liége. http://www.montefiore.ulg.ac.be/~poirrier/download/particle/poirrier-kdtree-pp1.pdf (Cited on page 79.)

Raghu Ramakrishnan and Johannes Gehrke. 2003. *Database Management Systems* (3 ed.). McGraw-Hill, Inc., New York, NY, USA. (Cited on pages 36 and 50.)

Rajib Kumar Rana, Chun Tung Chou, Salil S. Kanhere, Nirupama Bulusu, and Wen Hu. 2010. Ear-phone: An End-to-end Participatory Urban Noise Mapping System. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in*

Sensor Networks (IPSN '10). ACM, New York, NY, USA, 105–116. DOI: http://dx.doi.org/10.1145/1791212.1791226 (Cited on page 116.)

Paruj Ratanaworabhan, Jian Ke, and Martin Burtscher. 2006. Fast Lossless Compression of Scientific Floating-Point Data. In *2006 Data Compression Conference (DCC 2006), 28-30 March 2006, Snowbird, UT, USA*. 133–142. DOI: http://dx.doi.org/10.1109/DCC.2006.35 (Cited on page 24.)

Jorma Rissanen. 1976. Generalized Kraft Inequality and Arithmetic Coding. *IBM Journal of Research and Development* 20, 3 (1976), 198–203. DOI: http://dx.doi.org/10.1147/rd.203.0198 (Cited on page 22.)

John T. Robinson. 1981. The K-D-B-tree: A Search Structure for Large Multidimensional Dynamic Indexes. In *Proceedings of the 1981 ACM SIGMOD International Conference on Management of Data (SIGMOD '81)*. ACM, New York, NY, USA, 10–18. DOI: http://dx.doi.org/10.1145/582318.582321 (Cited on page 42.)

Peter Rousseeuw. 1987. Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. *J. Comput. Appl. Math.* 20, 1 (Nov. 1987), 53–65. DOI: http://dx.doi.org/10.1016/0377-0427(87)90125-7 (Cited on page 34.)

Nick Roussopoulos, Stephen Kelley, and Frédéic Vincent. 1995. Nearest Neighbor Queries. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, California, May 22-25, 1995*. 71–79. DOI: http://dx.doi.org/10.1145/223784.223794 (Cited on pages 51, 100, 284, 296, and 298.)

Yasushi Sakurai, Masatoshi Yoshikawa, Shunsuke Uemura, and Haruhiko Kojima. 2002. Spatial indexing of high-dimensional data based on relative approximation. *VLDB J.* 11, 2 (2002), 93–108. DOI: http://dx.doi.org/10.1007/s00778-002-0066-9 (Cited on pages 47 and 446.)

Hanan Samet. 1984. The Quadtree and Related Hierarchical Data Structures. *ACM Comput. Surv.* 16, 2 (June 1984), 187–260. DOI: http://dx.doi.org/10.1145/356924.356930 (Cited on page 41.)

Hanan Samet. 1990. *Applications of spatial data structures - computer graphics, image processing, and GIS*. Addison-Wesley. (Cited on page 54.)

Hanan Samet. 2005. *Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. (Cited on pages 34, 35, 36, 37, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 58, 61, 62, 64, 73, and 76.)

Hanan Samet. 2008. K-Nearest Neighbor Finding Using MaxNearestDist. *IEEE Trans. Pattern Anal. Mach. Intell.* 30, 2 (2008), 243–252. DOI: http://dx.doi.org/10.1109/TPAMI.2007.1182 (Cited on page 51.)

Hanan Samet and Markku Tamminen. 1988. Efficient Component Labeling of Images of Arbitrary Dimension Represented by Linear Bintrees. *IEEE Trans. Pattern Anal. Mach. Intell.* 10, 4 (July 1988), 579–586. http://dx.doi.org/10.1109/34.3918 (Cited on page 79.)

Hanan Samet and Robert E. Webber. 1985. Storing a Collection of Polygons Using Quadtrees. *ACM Trans. Graph.* 4, 3 (July 1985), 182–222. DOI: http://dx.doi.org/10.1145/282957.282966 (Cited on page 41.)

Michael Schiwietz. 1993. *Speicherung und Anfragebearbeitung komplexer Geo-Objekte*. Ph.D. Dissertation. Ludwig Maximilian University of Munich, Germany. http://d-nb.info/931737397 (Cited on page 47.)

Cristina Schmidt and Manish Parashar. 2003. Flexible Information Discovery in Decentralized Distributed Systems. In *12th International Symposium on High-Performance Distributed Computing (HPDC-12 2003), 22-24 June 2003, Seattle, WA, USA*. 226–235. DOI: http://dx.doi.org/10.1109/HPDC.2003.1210032 (Cited on page 54.)

G. Schrack and Xian Liu. 1995. The spatial U-order and some of its mathematical characteristics. In *IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing. Proceedings*. 416–419. DOI: http://dx.doi.org/10.1109/PACRIM.1995.519557 (Cited on page 40.)

Bernhard Seeger and Hans-Peter Kriegel. 1990. The Buddy Tree: An Efficient and Robust Access Method for Spatial Data Base. In *Proc. of the Sixteenth Intl. Conf. on VLDB*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 590–601. http://dl.acm.org/citation.cfm?id=94362.94615 (Cited on pages 43 and 82.)

Timos K. Sellis, Nick Roussopoulos, and Christos Faloutsos. 1987. The R^+-Tree: A Dynamic Index for Multi-Dimensional Objects. In *Proceedings of the 13th International Conference on Very Large Data Bases (VLDB '87)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 507–518. http://dl.acm.org/citation.cfm?id=645914.671636 (Cited on page 46.)

Kenneth C. Sevcik and Nick Koudas. 1996. Filter Trees for Managing Spatial Data over a Range of Size Granularities. In *Proceedings of the 22th International Conference on Very Large Data Bases (VLDB '96)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 16–27. http://dl.acm.org/citation.cfm?id=645922.673466 (Cited on page 41.)

Senthil Shanmugasundaram and Robert Lourdusamy. 2011. A comparative study of text compression algorithms. *International Journal of Wisdom Based Computing* 1, 3 (2011), 68–76. (Cited on page 22.)

Mehdi Sharifzadeh and Cyrus Shahabi. 2006. The Spatial Skyline Queries. In *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006*. 751–762. http://dl.acm.org/citation.cfm?id=1164192 (Cited on page 51.)

Mehdi Sharifzadeh and Cyrus Shahabi. 2010. VoR-tree: R-trees with Voronoi Diagrams for Efficient Processing of Spatial Nearest Neigh-

bor Queries. *Proc. VLDB Endow.* 3, 1-2 (Sept. 2010), 1231–1242. DOI: http://dx.doi.org/10.14778/1920841.1920994 (Cited on pages 35 and 47.)

Milad Shokouhi and Luo Si. 2011. Federated Search. *Found. Trends Inf. Retr.* 5, 1 (Jan. 2011), 1–102. DOI: http://dx.doi.org/10.1561/1500000010 (Cited on pages 12 and 13.)

C.K. Singh. 2018. *Geospatial Applications for Natural Resources Management*. CRC Press. https://books.google.de/books?id=metTDwAAQBAJ (Cited on page 3.)

Roger W. Sinnott. 1984. Virtues of the Haversine. *Sky and Telescope* 68, 2 (1984), 158. (Cited on page 62.)

Hans-Werner Six and Peter Widmayer. 1988. Spatial Searching in Geometric Databases. In *Proceedings of the Fourth International Conference on Data Engineering, February 1-5, 1988, Los Angeles, California, USA*. 496–503. DOI: http://dx.doi.org/10.1109/ICDE.1988.105496 (Cited on page 39.)

Valeriu Soltan and Alexei Gorpinevich. 1993. Minimum Dissection of a Rectilinear Polygon with Arbitrary Holes into Rectangles. *Discrete & Computational Geometry* 9 (1993), 57–79. DOI: http://dx.doi.org/10.1007/BF02189307 (Cited on page 28.)

Juan Humberto Sossa-Azuela, Cornelio Yáñez-Márquez, and Juan Luis Díaz-de-León S. 2001. Computing geometric moments using morphological erosions. *Pattern Recognition* 34, 2 (2001), 271–276. DOI: http://dx.doi.org/10.1016/S0031-3203(99)00213-7 (Cited on page 29.)

Daniel Stutzbach and Reza Rejaie. 2006. Understanding churn in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM Internet Measurement Conference, IMC 2006, Rio de Janeriro, Brazil, October 25-27, 2006*. 189–202. DOI: http://dx.doi.org/10.1145/1177080.1177105 (Cited on page 62.)

Tomás Suk and Jan Flusser. 2010. Refined Morphological Methods of Moment Computation. In *20th International Conference on Pattern Recognition, ICPR 2010, Istanbul, Turkey, 23-26 August 2010*. 966–970. DOI: http://dx.doi.org/10.1109/ICPR.2010.242 (Cited on page 30.)

Tomás Suk, Cyril Höschl IV, and Jan Flusser. 2012. Decomposition of binary images - A survey and comparison. *Pattern Recognition* 45, 12 (2012), 4279–4291. DOI: http://dx.doi.org/10.1016/j.patcog.2012.05.012 (Cited on pages 27, 28, 29, and 30.)

Markku Tamminen. 1982. The extendible cell method for closest point problems. *BIT Numerical Mathematics* 22, 1 (1982), 27–41. DOI: http://dx.doi.org/10.1007/BF01934393 (Cited on page 39.)

Y.L. Theng. 2009. *Handbook of Research on Digital Libraries: Design, Development, and Impact: Design, Development, and Impact*. Information

Science Reference. https://books.google.de/books?id=M6CB7Uhx1zgC (Cited on page 115.)

Theodoros Tzouramanis, Michael Vassilakopoulos, and Yannis Manolopoulos. 1998. Overlapping Linear Quadtrees: A Spatio-temporal Access Method. In *Proceedings of the 6th ACM International Symposium on Advances in Geographic Information Systems (GIS '98)*. ACM, New York, NY, USA, 1–7. DOI: http://dx.doi.org/10.1145/288692.288695 (Cited on page 53.)

Erik van der Zee and Henk J. Scholten. 2014. Spatial Dimensions of Big Data: Application of Geographical Concepts and Spatial Technology to the Internet of Things. In *Big Data and Internet of Things: A Roadmap for Smart Environments*. 137–168. DOI: http://dx.doi.org/10.1007/978-3-319-05029-4_6 (Cited on page 3.)

Ranga Raju Vatsavai, Auroop R. Ganguly, Varun Chandola, Anthony Stefanidis, Scott Klasky, and Shashi Shekhar. 2012. Spatiotemporal data mining in the era of big spatial data: algorithms and applications. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data, BigSpatial@SIGSPATIAL 2012, Redondo Beach, CA, USA, November 6, 2012*. 1–10. http://doi.acm.org/10.1145/2447481.2447482 (Cited on page 5.)

Chris Veness. 2016. Calculate distance, bearing and more between Latitude/Longitude points. (2016). http://www.movable-type.co.uk/scripts/latlong.html. Accessed: 29.11.2018. (Cited on pages 116 and 191.)

T. Vincenty. 1975. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey Review* 23, 176 (1975), 88–93. (Cited on page 62.)

F. Luccio C. Mugnai W. Lipski Jr., E. Lodi and L. Pagli. 1979. On two dimensional data organization II. In *Fundamenta Informaticae, ser. Annales Societatis Mathematicae Polonae, Series IV, vol. II*. 245–260. (Cited on page 27.)

Emo Welzl. 1991. Smallest Enclosing Disks (Balls and Ellipsoids). In *Results and New Trends in Computer Science*. Springer-Verlag, 359–370. (Cited on page 25.)

J. Weng. 1995. SHOSLIF: A framework for sensor-based learning for high-dimensional complex systems. In *Proc. IEEE Workshop on Architectures for Semiotic Modeling and situation analysis in Large Complex Systems*. 303–313. (Cited on page 35.)

David A. White and Ramesh Jain. 1996a. Similarity Indexing: Algorithms and Performance.. In *Storage and Retrieval for Image and Video Databases (SPIE) (SPIE Proceedings)*, Ishwar K. Sethi and Ramesh C. Jain (Eds.), Vol. 2670. SPIE, 62–73. http://dblp.uni-trier.de/db/conf/spieSR/spieSR96.html#WhiteDJ96 (Cited on page 46.)

David A. White and Ramesh Jain. 1996b. Similarity Indexing with the SS-tree. In *Proceedings of the Twelfth International Conference on Data*

Engineering (ICDE '96). IEEE Computer Society, Washington, DC, USA, 516–523. http://dl.acm.org/citation.cfm?id=645481.655573 (Cited on page 47.)

J.P. Wilson and A.S. Fotheringham. 2008. *The Handbook of Geographic Information Science*. Wiley. https://books.google.de/books?id=4iqX4926x40C (Cited on pages 5 and 6.)

Frank Wissbrock. 2004. Information Need Assessment in Information Retrieval—Beyond Lists and Queries. In *Proceedings of the KI 2004 Workshop on Machine Learning and Interaction for Text-Based Information Retrieval (TIR-04)*, Benno Stein, Sven Meyer zu Eißen, and Andreas Nürnberger (Eds.). 77–86. http://www.uni-weimar.de/medien/webis/events/tir-04/tir04-papers-final/wissbrock04-information-need-assessment-in-IR.pdf (Cited on page 13.)

Kesheng Wu, Ekow J. Otoo, and Arie Shoshani. 2001. A Performance Comparison of bitmap indexes. In *Proceedings of the 2001 ACM CIKM International Conference on Information and Knowledge Management, Atlanta, Georgia, USA, November 5-10, 2001*. 559–561. DOI: http://dx.doi.org/10.1145/502585.502689 (Cited on page 22.)

Kesheng Wu, Ekow J. Otoo, and Arie Shoshani. 2006. Optimizing bitmap indices with efficient compression. *ACM Trans. Database Syst.* 31, 1 (2006), 1–38. DOI: http://dx.doi.org/10.1145/1132863.1132864 (Cited on pages 21, 22, and 23.)

Rui Xu and Donald C. Wunsch II. 2005. Survey of clustering algorithms. *IEEE Trans. Neural Networks* 16, 3 (2005), 645–678. DOI: http://dx.doi.org/10.1109/TNN.2005.845141 (Cited on page 34.)

Yuh-Horng Yang, Kuo-Liang Chung, and Yao-Hong Tsai. 2000. A compact improved quadtree representation with image manipulations. *Image Vision Comput.* 18, 3 (2000), 223–231. DOI: http://dx.doi.org/10.1016/S0262-8856(99)00014-1 (Cited on page 54.)

Xiang Yin, Ivo Düntsch, and Günther Gediga. 2011. Quadtree Representation and Compression of Spatial Data. *Trans. Rough Sets* 13 (2011), 207–239. DOI: http://dx.doi.org/10.1007/978-3-642-18302-7_12 (Cited on page 54.)

Hai Yu, Pankaj K. Agarwal, Raghunath Poreddy, and Kasturi R. Varadarajan. 2008. Practical Methods for Shape Fitting and Kinetic Data Structures using Coresets. *Algorithmica* 52, 3 (2008), 378–402. DOI: http://dx.doi.org/10.1007/s00453-007-9067-9 (Cited on page 20.)

M. F. Zakaria, Louis J. Vroomen, Paul J. Zsombor-Murray, and J. M. H. M. van Kessel. 1987. Fast algorithm for the computation of moment invariants. *Pattern Recognition* 20, 6 (1987), 639–643. DOI: http://dx.doi.org/10.1016/0031-3203(87)90033-1 (Cited on page 29.)

Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko. 2006. *Similarity Search: The Metric Space Approach* (1st ed.). Springer Publishing Company, Incorporated. (Cited on pages 56 and 58.)

Pavel Zezula, Pasquale Savino, Giuseppe Amato, and Fausto Rabitti. 1998. Approximate Similarity Retrieval with M-Trees. *VLDB J.* 7, 4 (1998), 275–293. DOI: http://dx.doi.org/10.1007/s007780050069 (Cited on page 243.)

Chong Zhang, Weidong Xiao, Daquan Tang, and Jiuyang Tang. 2011. P2P-based multidimensional indexing methods: A survey. *Journal of Systems and Software* 84, 12 (2011), 2348–2362. DOI: http://dx.doi.org/10.1016/j.jss.2011.07.027 (Cited on pages 54 and 55.)

Jacob Ziv and Abraham Lempel. 1977. A universal algorithm for sequential data compression. *IEEE Trans. Information Theory* 23, 3 (1977), 337–343. DOI: http://dx.doi.org/10.1109/TIT.1977.1055714 (Cited on page 22.)

Jacob Ziv and Abraham Lempel. 1978. Compression of individual sequences via variable-rate coding. *IEEE Trans. Information Theory* 24, 5 (1978), 530–536. DOI: http://dx.doi.org/10.1109/TIT.1978.1055934 (Cited on page 22.)

University
of Bamberg
Press

Spatial data is everywhere, and its amount is increasing every day. Spatial data is often two-dimensional point data that is associated with data objects (e.g. media objects such as images or text). Due to the large amounts of data, there is a need for efficient and effective search systems that can handle the spatial properties of these data objects. The search systems can be both distributed and centralized solutions. In both scenarios, the concept of resource description and selection is an applicable paradigm for similarity searches performed regarding the spatial properties of the data objects. To achieve highly performant solutions, it is necessary to summarize the geographical footprint of a resource by means of geometrically delimited areas. Hereby, both efficiency as well as effectiveness are important. The term „effectiveness“ refers to a spatially very accurate geometric delimitation of the data point sets to be described whereas the term „efficiency“ aims at their storage-space-saving representation. The work deals with the development of effective and at the same time efficient summarization approaches for sets of two-dimensional data points. For the distributed application scenario, a total of 14 summarization approaches are presented and evaluated for different data collections. For the centralized application scenario, the two most suitable summary approaches are integrated into a centralized, multidimensional access structure and evaluated against the state-of-the-art for several data collections.

eISBN: 978-3-86309-673-1
9 783863 096731
www.uni-bamberg.de/ubp