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

Resource Description and Selection for Similarity Search in Metric Spaces

Daniel Blank

University of Bamberg Press

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

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

Band 19

University of Bamberg Press
2015

Resource Description and Selection for Similarity Search in Metric Spaces

von Daniel Blank

University of Bamberg Press
2015

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

Diese Arbeit hat der Fakultät Wirtschaftsinformatik und Angewandte Informatik der Otto-Friedrich-Universität als Dissertation mit dem Titel „Resource Description and Selection for Similarity Search in Metric Spaces. Problems and Problem-Solving Approaches using the Example of Content-Based Image Retrieval in Distributed Information Retrieval Systems“ vorgelegen.
1. Gutachter: Prof. Dr. Andreas Henrich
2. Gutachter: Prof. Dr. Ute Schmid
Tag der mündlichen Prüfung: 20. Januar 2015

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

http://www.uni-bamberg.de/ubp/

ISSN: 1867-7401
ISBN: 978-3-86309-310-5 (Druckausgabe)
eISBN: 978-3-86309-311-2 (Online-Ausgabe)
URN: urn:nbn:de:bvb:473-opus4-260465

Acknowledgements

First of all, I would particularly like to thank my supervisor Prof. Dr. Andreas Henrich for his support, his patience, his understanding, and his extraordinary faith in me during the last years ever since the beginning of the journey. His always helpful and inspiring comments guided the path I have taken. I would also like to thank the second assessor of my thesis Prof. Dr. Ute Schmid and Prof. Dr. Guido Wirtz for joining the thesis committee.

I would like to include in my thank all my past and actual colleagues at the Media Informatics Group—namely Dr. Raiko Eckstein, Martin Eisenhardt, Soufyane El Allali, Tobias Fries, Dr. Stefanie Gooren-Sieber, Tobias Gradl, Adrian Hub, Dr. Karlheinz Morgenroth, Hans-Christian Sperker, and Dr. Nadine Weber—as well as the administrative staff Silvia Förtsch, Siegfried Hofmann, and Daniela Pielenhofer. All of the aforementioned contributed to the challenging and respectful atmosphere with plenty hours of discussion and joy.

Special thanks go to my colleagues Stefan Kufer, Dr. Volker Lüdecke and PD Dr. Wolfgang Müller for also proofreading the thesis.

I would also like to acknowledge all contributors of the publicly available programming libraries used in this thesis.

Needless to say, I thank my dearest friends and my family back home—endless sources of courage and support.

Zusammenfassung

Die ständig zunehmende Menge an Medienobjekten im World Wide Web, in Unternehmen und auf privaten Endgeräten, verbunden mit einer immer größer werdenden Vielfalt an Datentypen, erfordert effektive und effiziente Suchdienste. Hierbei stellt die Bildsuche eine wichtige Aufgabe dar. In bestimmten Szenarien liegen die Medienobjekte dabei verteilt vor und Suchdienste müssen an diese veränderte Anforderung gegenüber einer zentralisierten Speicherung angepasst werden.

Die vorliegende Arbeit befasst sich mit der Ressourcenbeschreibung und -auswahl für die Ähnlichkeitssuche in metrischen Räumen. Zugriffsstrukturen für beliebige metrische Räume stellen flexibel einsetzbare Hilfsmittel für die Suche dar, da sie lediglich fordern, dass die Distanzfunktion zur Bestimmung der Nähe bzw. Ähnlichkeit zweier Medienobjekte eine Metrik ist. Als Szenario der Arbeit dient die Ressourcenbeschreibung und -auswahl bei der inhaltsbasierten Bildsuche in verteilten Information-Retrieval-Systemen am Beispiel des Peer-to-Peer Information Retrievals. Bei der Ressourcenauswahl gilt es, anhand von geeigneten Ressourcenbeschreibungen die vielversprechendsten Ressourcen zu bestimmen, die für das Informationsbedürfnis des Benutzers relevante Medienobjekte verwalten.

In dieser Arbeit werden Zugriffsstrukturen entwickelt, die in beliebigen metrischen Räumen arbeiten. Die ersten drei Kapitel der Arbeit führen in die Thematik ein, beleuchten die Grundlagen und betrachten existierende Ansätze auf dem Gebiet. In den Kapiteln 4, 5 und 6 werden die Ziele adressiert, auf die die Arbeit ausgerichtet ist und die im Folgenden kurz beschrieben sind.

Den Ausgangspunkt der Arbeit bildet ein Verfahren zur Ressourcenbeschreibung und -auswahl aus Eisenhardt et al. [2006] auf Basis sogenannter Cluster-Histogramme. Es erweitert einen Ansatz, der in Müller et al. [2005a] beschrieben ist. Ein wesentliches Ziel der vorliegenden Arbeit ist es, Ressourcenbeschreibungen und Auswahltechniken zu entwickeln, die den Ansatz auf Basis der Cluster-Histogramme

hinsichtlich Speicherplatzbedarf der Ressourcenbeschreibungen sowie Leistungsfähigkeit bei der Ressourcenauswahl verbessern. Die Arbeit stellt hierzu hochfeine, komprimierte Cluster-Histogramme vor. Durch die Verwendung einer Vielzahl an Referenzobjekten entstehen dünn besetzte Histogramme, die sich durch den Einsatz von Kompressionsverfahren auf kompakte Weise repräsentieren lassen. Die damit verbundene feingranulare Partitionierung des Feature-Raums erlaubt eine effiziente Ressourcenauswahl unter Verwendung kompakter Ressourcenbeschreibungen.

Der Ansatz auf Basis der Cluster-Histogramme und die in der vorliegenden Arbeit vorgestellten Erweiterungen adressieren die approximative Suche, ohne dabei präzise Ergebnisse zu garantieren. Daher ist es ein zweites wesentliches Ziel der Arbeit, Algorithmen für die Ressourcenbeschreibung und -auswahl in beliebigen metrischen Räumen zu entwickeln, die eine präzise Ähnlichkeitssuche ermöglichen. Dies führt zu RS4MI – ein Ansatz zur Ressourcenbeschreibung und -auswahl in metrischen Räumen, der sowohl die präzise als auch die approximative Suche unterstützt und die gegenläufige Abhängigkeit zwischen Speicherplatzeffizienz der Ressourcenbeschreibungen und deren Selektivität bei der Ressourcenauswahl steuern kann.

Ferner beleuchtet die Arbeit die breite Anwendbarkeit der Mechanismen zur Ressourcenbeschreibung und -auswahl. Diese ist für verschiedene Anwendungsfelder auch abseits des Peer-to-Peer Information Retrievals und des traditionellen verteilten Information Retrievals gegeben. Als ein Beispiel wird in dieser Arbeit IF4MI vorgestellt – eine zentralisierte Zugriffsstruktur für beliebige metrische Räume, die auf dem Konzept der invertierten Liste basiert. IF4MI verbindet Ansätze von hierarchischen und mehrstufigen metrischen Zugriffsstrukturen für die präzise Suche mit approximativen Ansätzen auf Basis invertierter Listen. Ein zweites Anwendungsbeispiel zeigt, wie die Techniken zur Ressourcenbeschreibung und -auswahl auf dem Gebiet der visuellen Analyse zur Erschließung großer Mengen von Bildkollektionen eingesetzt werden können. Dabei ist die Anwendbarkeit keinesfalls auf den Medientyp Bild beschränkt.

Contents

1. Introduction **1**
- 1.1. Motivation . . . 2
- 1.2. Problem Description . . . 6
  - 1.2.1. Resource Description and Selection . . . 6
  - 1.2.2. Similarity Search in Metric Spaces . . . 8
- 1.3. Thesis Objectives . . . 10
- 1.4. Thesis Structure . . . 12

2. Foundations of Metric Space Indexing **15**
- 2.1. The Metric Postulates . . . 16
- 2.2. Distance Measures and their Application . . . 17
  - 2.2.1. Minkowski Distances . . . 18
  - 2.2.2. Quadratic Form Distances . . . 18
  - 2.2.3. Earth Mover's Distances . . . 19
  - 2.2.4. Edit Distances . . . 19
  - 2.2.5. Hamming Distance . . . 20
  - 2.2.6. Squared Chord, Hellinger, and Matusita Distance . 20
  - 2.2.7. Distance Functions based on Cosine Similarity . . . 21
  - 2.2.8. Jaccard Distance . . . 21
  - 2.2.9. Canberra Distance . . . 22
  - 2.2.10. Kullback-Leibler Divergence and its Variants . . . 22
  - 2.2.11. Semantic and Other Distance Metrics . . . 23
- 2.3. Metric Space Partitioning . . . 23
  - 2.3.1. Generalized Hyperplane Partitioning . . . 24
  - 2.3.2. Ball Partitioning . . . 29
- 2.4. Pruning Rules . . . 32
  - 2.4.1. Using the Double-Pivot Distance Constraint . . . 32
  - 2.4.2. Using the Range-Pivot Distance Constraint . . . 33
  - 2.4.3. Using the Object-Pivot Distance Constraint . . . 35
- 2.5. Distance Distributions and the Intrinsic Dimensionality . . 36
- 2.6. Pivot Selection Techniques . . . 38

3. Metric Access Methods **41**
3.1. Clustering in Arbitrary Metric Spaces 42
3.2. Centralized Metric Access Methods 44
3.2.1. Distance-based Metric Access Methods for Precise Search 45
3.2.2. Embedding Methods for Precise Search 50
3.2.3. Metric Access Methods for Approximate Search . . 53
3.3. Distributed Metric Access Methods 57
3.3.1. The Diversity of Distributed Metric Access Methods 57
3.3.2. Resource Selection in Brokered Architectures . . . 67
3.3.3. Resource Selection in Structured Architectures . . 68
3.3.4. Resource Selection in Completely Decentralized Architectures 69
3.3.5. Resource Selection in Hierarchical Architectures . . 73

4. IF4MI—Inverted File for Metric Indexing and Search **77**
4.1. Outline of IF4MI 77
4.2. Evaluating IF4MI 81
4.2.1. Experimental Setup 81
4.2.2. The Number of Distance Computations of IF4MI . 82
4.2.3. Comparing the Number of Distance Computations of IF4MI with the M-tree and the PM-tree 87
4.2.4. Memory Requirements of the Approaches 90
4.2.5. Using the Space Partitioning of the M-Index 93
4.2.6. Improvements to Pivot Filtering 95
4.2.7. Processing Filter Queries 97

5. RS4MI—Resource Description and Selection for Metric Indexing and Search **101**
5.1. RS4MI for Approximate Search 101
5.1.1. Extending Cluster Histograms to HFS and UFS . . 102
5.1.2. Excursus: HFS and UFS for Geographic IR 108
5.2. RS4MI for Precise Range Query Processing 116
5.2.1. Experimental Setup 116
5.2.2. M-tree-based Local Clustering for Comparison . . . 117
5.2.3. Local k-medoids Clustering for Comparison 121
5.2.4. RS4MI Variants and their Evaluation 125
5.2.5. Brief Comparison of the Approaches 130

5.3. RS4MI for Precise k-Nearest Neighbors Query Processing . 131
5.3.1. A Conceptual Algorithm for Precise RS4MI k-Nearest Neighbors Query Processing 132
5.3.2. Evaluating Precise RS4MI k-Nearest Neighbors Query Processing 134

6. Applicability of the Approaches 141

7. Conclusion and Outlook 151

A. Photo License Information 155

List of Abbreviations 157

List of Symbols 159

List of Figures 168

List of Tables 170

References 171

Chapter 1.

Introduction

The task of resource description and selection reaches back to the origins of both writing and "information retrieval". Economic transactions, texts of hymns, prayers, and incantations were written on clay tablets by the Sumerians around 3000 B.C. Multiple tablets were grouped into boxes, baskets, and receptacles. They were archived in storerooms. In order to determine which boxes to open, small tablets per box described its content in an adequate way. In most cases, incipits (the first few words of the texts) together with date information were used instead of titles and keywords[1] [Lerner, 2009, *ch.* 1].

The Sumerians chose lists of incipits as adequate resource descriptions summarizing the content of the tablets in the boxes. Early "librarians" with a given information need had to read the incipits and perform resource selection, that is, select promising boxes to be opened. Rüger [2010, *ch.* 1, *p.* 2] notes: "*... little has changed except technology. Today, rather than climbing down into storerooms and matching our information need with the incipits of the documents in boxes, we deploy computers to match our information need with the text of the documents itself.*"

With the use of automated information processing technology, it is nowadays possible to analyze and index huge amounts of individual documents instead of relying on summarized information aggregated over multiple documents. Nevertheless, there are many scenarios where resource description and selection techniques are still beneficial and inevitable, such as in distributed retrieval scenarios.

The focus of this thesis lies in the design and analysis of resource description and selection techniques for similarity search in general metric spaces. Similarity search is an important search paradigm where similar objects with respect to a given query object are to be retrieved. Here,

[1] Lerner [2009, *p.* 3] notes that keywords chosen from the first two lines of the texts were also used in one of the Sumerian catalogues, but this approach was not adopted by later catalogers.

it is common to model the dissimilarity between objects as a distance between them assuming that the smaller the distance, the higher the similarity. The similarity search paradigm is frequently applied in different application fields. Many similarity search problems are modeled in metric spaces where the distance measure is a metric. Thus, metric space indexing techniques (see chapter 3) which do not rely on any assumptions about the object representations can be used. Examples for the application of metric space indexing techniques are text retrieval, multimedia retrieval, 3D object retrieval, similarity search on business process models, data compression, pattern recognition, machine learning, biomedical databases, statistical data analysis, malware detection, and data mining [Bustos et al., 2007; Chávez et al., 2001b, *ch.* 2; Hu et al., 2009; Kunze and Weske, 2011; Zezula et al., 2006, *p.* 3f.].

In this thesis, content-based image retrieval (CBIR) in a particular peer-to-peer (P2P) information retrieval (IR) system provides the example scenario for the design and evaluation of resource description and selection techniques for similarity search in general metric spaces[2]. This scenario is introduced and motivated in section 1.1. The research problem is formulated in section 1.2. Major thesis objectives are described in section 1.3 whilst the structure of the thesis is outlined in section 1.4.

1.1. Motivation

The ever increasing amount of media items in the World Wide Web (WWW), within companies, and on private devices as well as the growing diversity in data types [Novák, 2008, *p.* 5] requires effective and efficient retrieval techniques. Besides text, audio, and video retrieval, searching for images has become an important retrieval task.

In the early days of web image retrieval, search engines started using textual information on web pages in the vicinity of the images and text retrieval techniques to allow for image retrieval. Later, photo sharing communities such as Flickr[3] began to use user-provided tags, that is,

[2] Most of the research assuming the general metric space model for distributed similarity search is presented in the field of P2P IR. Resource description and selection techniques can of course also be applied in other infrastructures with physically distributed nodes such as in cloud infrastructures, grid infrastructures, or sensor networks, to name only a few.

[3] see *http://www.flickr.com*, last visit: 7.10.2014

1.1.1 — Excerpt of an initial search result.

1.1.2 — Excerpt of the refined search result after a similarity search for the "University of Bamberg" logo was performed.

Figure 1.1. — Using the image similarity search of Google[6] on 16.5.2013 (*http://images.google.com/*). Images in an initial search result can be used as query images for the search of visually similar images.

textual annotations describing the image content. Moreover, geographic and temporal metadata have been applied as important filter and search criteria. More recently, commercial search engines such as Bing[4] and Google[5] (see figure 1.1) have adopted CBIR techniques together with the similarity search paradigm to search for visually similar images according to a given query image.

CBIR uses automatically extracted image features which describe for example color, texture, or shape properties of an image. The application of CBIR techniques is especially beneficial when tags or other forms of metadata are ambiguous or not available. In an analysis of more than 100 million Flickr images, approximately 30% of the images have no tags or comments at all [Bolettieri et al., 2009, *p.* 11f.]. Bolettieri et al. [2009, *p.* 11] also reveals that the average number of comments per image is 0.52 and on average an image offers 5.02 tags with the distributions of both the number of tags and the number of

4 see *http://www.bing.com*, last visit: 7.10.2014
5 see *http://www.google.com*, last visit: 7.10.2014
6 Google and the Google logo are registered trademarks of Google Inc., used with permission.

comments per image being highly skewed. Additional problems arise for example from tag spamming (see e.g. Heymann et al. [2007]), lexical relations such as tag homonymy and synonymy (see e.g. Begelman et al. [2006]), and insufficient tag quality (see e.g. Müller et al. [2006] and Sen et al. [2007]). When trying to overcome these problems, CBIR offers beneficial techniques. It is shown that combining text IR and CBIR improves image retrieval quality [Paramita et al., 2009; Popescu et al., 2010; Tsikrika et al., 2011].

However, retrieval by visual content information is a challenging task. Besides the difficulty of building effective CBIR systems (see e.g. Datta et al. [2008]), it is also challenging to design scalable and efficient solutions. Here, P2P IR systems are a potential alternative to client/server-based techniques currently offered by media sharing communities, web search engines, and social network sites.

Adapting the peer-to-peer definition in Milojicic et al. [2002, *p.* 1], P2P IR systems can be described as systems employing distributed resources (i.e. computing devices) to perform content-based search tasks in a decentralized manner. The devices can act as both clients and servers. By applying a scalable P2P IR protocol, a decentralized service for the administration of media items can be established in contrast to existing client/server-based solutions. There is no need for an expensive infrastructure [Han et al., 2004, *p.* 209] and idle computing power in times of inactivity can be used to perform CPU-intensive tasks [Clark, 2001, *p.* 19] such as analyzing and enriching media items. P2P IR systems offer the benefit that media items can remain on individual devices since there is no need for storing them on remote servers which are hosted by third-party service providers. Crawling, which consumes large amounts of web traffic, can thus be avoided [Bockting and Hiemstra, 2009, *p.* 1]. In addition, dependency from service providers which act as "*informational gatekeepers*" [Tene, 2008, *p.* 1490] can be reduced because they no longer determine which information can be retrieved or accessed and which cannot. So, in times of a strong market concentration in web search and social network applications as well as public debates addressing the privacy of data, P2P IR can offer benefits. In addition, it remains unclear if and how long existing client/server-based solutions can cope with the dramatic increase in data volumes.

Some P2P IR systems follow the idea of classic distributed IR systems, where resource selection techniques are applied to determine a ranking of promising resources based on descriptions of their content.

The resources are queried in ranked order to retrieve appropriate media items according to a user's information need[7].

In general, different criteria can be employed for the retrieval of media items such as text, timestamps, geographic footprints, and (low-level) audio or visual content information. Resource description and selection techniques for text data which are applicable in P2P IR settings are for example proposed in Cuenca-Acuna et al. [2003]. They are not addressed in this thesis, nor do we focus on techniques for time and date information. The focus of this thesis is the design and analysis of resource description and selection techniques for content information. These techniques are however also applicable for resource selection using geographic information as will be discussed in section 5.1.2[8].

As a proof-of-concept, our resource description and selection techniques are evaluated based on a certain type of P2P CBIR system (for an outline see section 3.3.4 on pages 72–73). Nevertheless, the techniques are by no means limited to this particular P2P IR setting. They can also be applied in the context of other variants of distributed IR systems (see section 3.3). Furthermore, there is a range of possible application fields apart from distributed IR (see chapter 6). The techniques

[7] The information need concept is central to many definitions of an IR system. According to Frakes [1992, *p.* 1]: "*An IR system matches user* queries*—formal statements of information needs—to documents stored in a database.*" Translating an information need into a representation an IR system can deal with is difficult. Queries can be vague, imprecise, and incomplete [Henrich, 2008, *p.* 24]. In the following, it is assumed that there is an adequate "formal statement" of an information need. Hence, the process of obtaining this representation is not part of the subject matter. The same applies for the process of obtaining document representations which are usually stored in the database instead of the documents themselves. In the example scenario of the thesis, images are used as documents. Document representations are in the following also called feature objects or database objects.

[8] Resource selection based on a single criterion, for example image content, is only a first step on the way to an effective CBIR system. When querying for multiple criteria, for example for an image with a particular content which was taken in a certain geographic region, criterion-specific resource rankings can be combined by applying a merging algorithm for ranked lists (see e.g. Belkin et al. [1995] and Ilyas et al. [2008]). Moreover, resource description and selection schemes can be designed which support content-based search and in addition preserve the geographic distribution of the images by integrating both content-based and geographic search criteria (see e.g. Hariharan et al. [2008] which combines text and geographic information). However, such aspects are out of the scope of this thesis.

are also not limited to CBIR and the algorithms for precise search[9] presented in chapter 4 and sections 5.2 and 5.3 can be applied whenever the retrieval task can be modeled as a metric space search problem. In addition, as usual, the resource selection schemes for approximate search from section 5.1 can be adapted and used for similarity search in non-metric spaces (for a definition see section 2.1) where the amount of triangle inequality violations affects the search effectiveness of metric access methods when applied to non-metric similarity search tasks (see e.g. Skopal [2007]).

1.2. Problem Description

This section introduces the two concepts which form the main thesis title—namely *resource description and selection* (see section 1.2.1) and *similarity search in metric spaces* (see section 1.2.2)—in order to further clarify the thematic focus of this thesis.

1.2.1. Resource Description and Selection

The resource description and selection techniques presented in this thesis can be used in traditional distributed IR scenarios. Callan [2000] defines distributed IR based on three basic problems and tasks:

Resource description. To identify resources which most likely contain documents that satisfy a given information need, resources have to be described in an adequate way[10]. Callan [2000, *p.* 128] talks of "*brief*" resource descriptions which are in the following also

[9] The term *precise* (*similarity*) *search* is used to denote that all database objects which fulfill the similarity search criterion must be present in the result set. This corresponds to the use in for example Zezula et al. [1998]. From an IR perspective, this means that 100 per cent recall are required. The term *exact search* is not used in this context since it also refers to search tasks where only the exact query object is to be retrieved. On the other hand, *approximate* (*similarity*) *search* refers to search scenarios where—usually for runtime performance reasons—not necessarily all of the database objects which fulfill the search criterion are retrieved (see e.g. Patella and Ciaccia [2009]).

[10] On the one hand, cooperative resources compute the resource descriptions themselves. In case of uncooperative resources, the resource descriptions might be obtained through query-based sampling, that is, computing the resource descriptions from past query results (for the general idea in case of text retrieval see Callan and Connell [2001]; see e.g. Berretti et al. [2004] for query-based sampling in case of CBIR).

called summaries. Most of distributed IR research is concerned with text documents. Here, resources are for example described by the set of terms, or a representative subset of them, which are contained in the documents of a resource, and some kind of frequency information per term plus sometimes additional statistics. However, for some scenarios these brief descriptions are still not space efficient enough. Hence, the use of Bloom filters[11] is for example proposed for text retrieval in Cuenca-Acuna et al. [2003].

Resource selection. Based on a known set of resource descriptions, the entity that performs resource selection[12] decides which resources to query during search. It is a goal of resource selection to contact only those resources which contain relevant documents according to a user's information need. Routing the query to resources without any relevant documents should thus be avoided.

Since it is difficult to analyze resource selection independent from resource description techniques and vice versa, resource selection often refers to both aspects—resource description and selection. This corresponds to the understanding in this thesis where both resource description and selection techniques are analyzed.

Result merging. If a resource is contacted, it queries its local document collection and possibly returns a ranking of documents to the inquiring entity. After having received ranked lists from different resources, the inquiring entity has to merge these lists. This might be a challenging task because the relevance scores representing query-document similarities computed locally by different resources might be based on local (i.e. resource-specific) statistics and not be comparable. Since throughout the thesis cooperating resources are assumed all using the same similarity

[11] A Bloom filter [Bloom, 1970] is a bit vector indicating if a certain item (for example a term) is present or not. Multiple hash functions are applied for mapping an item to a set of bit positions. False positives may arise when multiple items are mapped to the same bit positions.

[12] In literature, this task is sometimes for example also called "*source selection*" [Paltoglou et al., 2008], "*server selection*" [Thomas and Hawking, 2009], "*database selection*" [Bender et al., 2005b], "*collection selection*" [Bockting and Hiemstra, 2009], "*peer-selection*" [Mass et al., 2011], or "*query routing*" [Nottelmann and Fuhr, 2006].

measure which is independent from local statistics, result merging becomes trivial in this case. Such a scenario is for example also assumed in Eisenhardt et al. [2006] and Müller et al. [2005a].

1.2.2. Similarity Search in Metric Spaces

The notion of similarity between a query and a document is central to IR. A similarity space is a pair $(\mathbb{U}, sim)$ where the universe $\mathbb{U}$ corresponds to the domain of feature objects and *sim* represents a similarity measure [Skopal and Bustos, 2011, *p.* 34:4]. It is common in many application domains such as CBIR to model the dissimilarity between two feature objects as a distance $dist : \mathbb{U} \times \mathbb{U} \rightarrow \mathbb{R}$ between them. Thus, $(\mathbb{U}, dist)$ is a dissimilarity space with a distance function *dist*. In general, it is assumed that the smaller the distance between two objects, the higher the similarity[13]. Retrieving the most similar items (for example images) according to a given query from the database $O \subset \mathbb{U}$ hence results in finding the closest (in terms of the distance function *dist*) database objects to the query object $q \in \mathbb{U}$. Thus, similarity queries are also referred to as proximity queries [Chávez et al., 2001b]. Range and k-nearest neighbors queries (k-NN queries) are among the most popular types of similarity queries [Kriegel et al., 2007, *p.* 75]. Hence, these types of queries are addressed in the following. Other types of similarity queries are for example described in Zezula et al. [2006, *p.* 15ff.]. Both, range and k-NN queries are based on the "*query by image example*" paradigm[14] [Smeulders et al., 2000, *p.* 1367] in case of CBIR.

[13] In this thesis, it is assumed that the similarity between feature objects can be adequately modeled with the help of an appropriate distance or dissimilarity measure. This modeling process together with its various psychological aspects (references are for example mentioned in Skopal and Bustos [2011]) is however out of the scope of the thesis.

[14] The "*query by image example*" paradigm [Smeulders et al., 2000, *p.* 1367] has found its way to web search. Query images can for example be selected from an existing result set which is the outcome of an initial text query and visually similar images are then retrieved (see e.g. figure 1.1 on page 3).

Definition 1 (range query, query ball):
A range query $range(q, r)$ with the query object $q \in \mathbb{U}$ and the search radius $r \in \mathbb{R}^+$ retrieves all database objects from O which are within distance r from q, that is, $\{o \in O \mid dist(q, o) \leq r\}$. The subspace $\mathbb{Q} \subset \mathbb{U}$ for which $\forall o \in \mathbb{Q} : dist(q, o) \leq r$ and $\forall o' \in \mathbb{U} \backslash \mathbb{Q} : dist(q, o') > r$ is called the query ball [Skopal and Bustos, 2011, *p.* 34:4]. □

In some scenarios, it is difficult to explicitly specify the search radius r. Instead, a user might be interested in the k closest database objects to q.

Definition 2 (k-nearest neighbors query):
A k-nearest neighbors query $knn(q, k)$ retrieves the k closest database objects to q, that is, a set $K \subset O$ with $\forall o \in K, o' \in O \backslash K : dist(q, o) \leq dist(q, o')$ and $|K| = k$ [Skopal and Bustos, 2011, *p.* 34:4]. Hereby, $|O| \gg k$ is assumed. □

It is hard to design efficient indexing techniques for general dissimilarity spaces [Skopal and Bustos, 2011]. However, many similarity search problems can be modeled in metric spaces. Here, the underlying dissimilarity space is a metric space where the distance measure *dist* satisfies the metric postulates (see section 2.1).

In literature, there is a distinction between metric access methods (MAMs) and (multidimensional) spatial access methods (SAMs) [Skopal, 2004]. Skopal [2010, *p.* 13] defines a MAM as a "*set of algorithms and data structure(s) providing efficient (fast) similarity search under the metric space model*". The metric space model is introduced in section 2.1. Whereas MAMs can be applied for similarity search in general metric spaces where no assumption is made about the representation of the database objects, SAMs are designed for vector spaces[15]. Comprehensive surveys on SAMs are for example presented in Gaede and Günther [1998] and Samet [2006]. MAMs are introduced and described in chapter 3.

[15] Note that any norm $||\cdot||$ defined over a vector space $\mathcal{V}$ and thus any normed vector space $(\mathcal{V}, ||\cdot||)$ induces a metric space by defining $\forall \vec{x}, \vec{y} \in \mathcal{V} : dist(x, y) = ||\vec{y} - \vec{x}||$ (see e.g. Hansen [1999, *p.* 57f.]).

1.3. Thesis Objectives

In this thesis, resource description techniques allowing for efficient[16] resource selection for similarity search in general metric spaces are presented. The main objectives of the thesis are as follows:

¶ The starting point of the research is an approach on resource selection from Eisenhardt et al. [2006] outlined in section 3.3.4 on pages 72–73. It enhances earlier work from Müller et al. [2005a]. *A major objective of this thesis is to devise resource description and selection techniques which improve the earlier approach presented in Eisenhardt et al. [2006] both in terms of space efficiency of the resource descriptions as well as in terms of resource selection performance.* The latter is measured by the fraction of resources which have to be queried—the less the better—to retrieve a certain fraction of the precise search result. This cost measure is frequently applied in the literature on distributed query processing such as in Bender et al. [2005b], Eisenhardt et al. [2006], Müller et al. [2005b], and Vlachou et al. [2012b].

The contribution of this thesis in this regard is presented in section 5.1.

- The approach in Eisenhardt et al. [2006] as well as the extensions to it presented in section 5.1 are approximate techniques which cannot guarantee precise results. *Thus, as a second major objective, the thesis will provide resource selection algorithms for precise similarity search in general metric spaces.* This leads to Resource description and Selection for Metric Indexing and search (RS4MI)—a framework for resource selection in metric spaces where different pruning rules become applicable depending on the trade-off between space efficiency of the resource descriptions and resource selection performance.

  Using the techniques for approximate similarity search presented in section 5.1, RS4MI should allow for both precise and approximate search. Extending RS4MI with capabilities for precise search is addressed in sections 5.2 and 5.3.

[16] Of course, efficiency of an IR system is not the only important aspect. Ensuring search effectiveness is essential, too. Nevertheless, this thesis focuses only on efficiency issues.

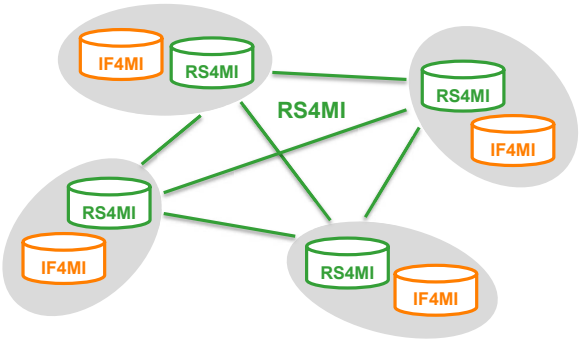


Figure 1.2. — Our scenario of a distributed RS4MI system also using IF4MI for the local query processing (resources shaded gray).

As a third major objective, the thesis should reveal the broad applicability of the resource description and selection techniques in different application fields apart from the P2P and distributed IR domain.

As one example, Inverted File for Metric Indexing and search (IF4MI) is proposed in chapter 4—a centralized MAM based on the inverted file concept. IF4MI bridges the gap between existing hierarchical and multi-step MAMs for precise search on the one hand and approximate techniques based on inverted files on the other. In addition, the use of IF4MI complements RS4MI when designing a practical distributed metric space search system. Resources can use IF4MI for local query processing and to support distributed query processing based on RS4MI (for an overview see figure 1.2, explained in more detail throughout this thesis).

As a second application example, in chapter 6, the resource description and selection techniques are applied in the field of visual analytics for the analysis of large sets of image collections. Here, the applicability is by no means restricted to the image media type.

1.4. Thesis Structure

The remainder of this thesis is structured as follows:

Chapter 2 outlines in more detail important concepts of metric space indexing since they provide the theoretical background of this thesis and the basis for the design and analysis of MAMs—both the already existing approaches discussed in chapter 3 as well as the newly proposed IF4MI and RS4MI in chapters 4 and 5, respectively. First, section 2.1 states the metric postulates. Afterwards, section 2.2 presents various distance measures with a focus on distance metrics. Different space partitioning schemes and pruning rules which are seminal for the design of MAMs are presented in section 2.3 and section 2.4, respectively. Section 2.5 introduces the analysis of distance distributions and the concept of the intrinsic dimensionality which is helpful to quantify the difficulty of a metric space indexing task. Selecting reference objects[17] is an inevitable step in the construction of MAMs. This task is discussed in section 2.6.

Chapter 3 gives an overview on existing MAMs and thus recapitulates already existing approaches related to IF4MI and RS4MI. Clustering in arbitrary metric spaces is briefly introduced in section 3.1 since it is closely related to the design of MAMs. Afterwards, centralized MAMs are addressed in section 3.2. Multiple centralized MAMs such as M-tree [Ciaccia et al., 1997] and PM-tree [Skopal et al., 2005] implementations as well as a variant of the M-Index [Novák and Batko, 2009; Novák et al., 2011] serve as comparison baselines when evaluating the performance of IF4MI in chapter 4. Distributed MAMs are finally outlined in section 3.3.

Chapter 4 presents IF4MI. It takes precise search to the MAMs based on inverted files. Since IF4MI is built on top of an inverted file, it inherently provides a multi-feature MAM with additional text retrieval capabilities. The main characteristics of IF4MI are outlined in section 4.1. Afterwards, its applicability is evaluated in section 4.2.

[17] In the literature, reference objects are for example also called "*anchors*" [Novák and Zezula, 2013], "*foci*" [Traina Jr. et al., 2007], "*pivots*" [Chávez et al., 2001b], "*sites*" [Skala, 2009], or "*vantage objects*" [Vleugels and Veltkamp, 1999].

Chapter 5 outlines RS4MI, a resource selection framework for similarity search in general metric spaces. The chapter consists of three major sections. Section 5.1 focuses on approximate search techniques which extend the approach from Eisenhardt et al. [2006] (outlined in section 3.3.4 on pages 72–73). Afterwards, algorithms for precise search are addressed. Section 5.2 focuses on range query processing whereas section 5.3 discusses the processing of k-NN queries.

Chapter 6 highlights some research fields where resource description and selection techniques for arbitrary metric spaces can provide a valuable contribution. Application domains are listed where the resource description and selection techniques developed in this thesis can be used. In general, two modes of application are distinguished—searching for similar feature objects according to a given query object and searching for similar resources given a particular resource description. Chapter 6 also shows how RS4MI can be used in the field of visual analytics for the analysis of for example large sets of image collections.

Chapter 7 concludes this thesis. The final chapter gives a brief summarization and points out how the thesis objectives have been addressed. Finally, the chapter outlines important aspects of future work.

Figure 1.3 summarizes the structure of this thesis and shows where the main thesis objectives are particularly addressed.

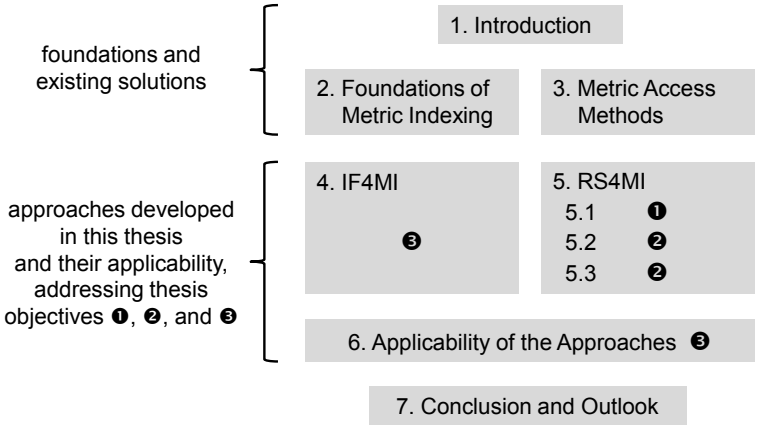


Figure 1.3. — Visualization of the thesis structure additionally indicating where the main thesis objectives are particularly addressed.

Chapter 2.

Foundations of Metric Space Indexing

This chapter outlines in more detail important concepts in the field of metric space indexing and search and thus provides the theoretical background of the thesis in this regard. The concepts which are introduced here have successfully been applied for the design and the analysis of MAMs—both for the design and the analysis of the already existing approaches discussed in chapter 3 as well as the newly proposed IF4MI and RS4MI in chapter 4 and chapter 5, respectively

MAMs (for an overview see chapter 3) require the distance measure to be a metric. Thus, section 2.1 states the assumptions of the metric space approach by introducing the metric postulates.

There is a wide applicability of MAMs in different application domains, as already mentioned in chapter 1. This is partially due to the fact that there is a huge amount of available distance metrics designed for the most diverse purposes. Section 2.2 introduces some distance measures with a focus on distance metrics. By introducing them, we also provide examples of their application in the field of CBIR and other domains. Some of the distance metrics are used by the MAMs presented in chapter 3 and thus important for the understanding of their application purpose. Furthermore, we use different metrics when evaluating our approaches IF4MI and RS4MI in chapters 4 and 5.

Space partitioning schemes and pruning rules which allow for the pruning of certain feature space regions or individual database objects from search provide the basis for the design of efficient MAMs. Various metric space partitioning schemes are presented in section 2.3. Corresponding pruning rules are outlined in section 2.4.

An important aspect when designing and applying MAMs is the analysis of distance distributions and the concept of the intrinsic dimensionality outlined in section 2.5. It is helpful for quantifying the difficulty of a metric space indexing task. A high intrinsic dimension-

ality can be perceived as an indicator for the presence of the curse of dimensionality known from vector space indexing [Pestov, 2007].

Selecting pivots which act as anchors in metric space indexing is an inevitable step in the construction of MAMs. Anchors are necessary for the space partitioning because coordinate information such as in case of SAMs cannot be exploited by MAMs. We focus on the task of pivot selection in section 2.6.

For more comprehensive introductions on metric space indexing see for example Chávez et al. [2001b], Clarkson [2006], Hetland [2009b], Hjaltason and Samet [2003a], and Zezula et al. [2006].

2.1. The Metric Postulates

The concept of a dissimilarity space $(\mathbb{U}, dist)$ is introduced in section 1.2.2. In a metric space $\mathcal{M} = (\mathbb{U}, dist)$, *dist* is a metric. Thus, *dist* is a real-valued distance (function) defined on $\mathbb{U} \times \mathbb{U}$ which satisfies the metric postulates 2.1 to 2.4 for all $x, y, z \in \mathbb{U}$ [Deza and Deza, 2009, *p.* 4; Hattori, 2003; Skopal and Bustos, 2011, *pp.* 34:6-34:7; Zezula et al., 2006, *p.* 8f.][18]:

$$dist(x, y) > 0 \iff x \neq y \qquad \textit{non-negativity} \qquad (2.1)$$

$$dist(x, y) = 0 \iff x = y \qquad \textit{identity of indiscernibles} \qquad (2.2)$$

$$dist(x, y) = dist(y, x) \qquad \textit{symmetry} \qquad (2.3)$$

$$dist(x, y) + dist(y, z) \geq dist(x, z) \qquad \textit{triangle inequality} \qquad (2.4)$$

Distance functions which do not satisfy all of the abovementioned postulates are in the following referred to as non-metric distance functions. A dissimilarity space $(\mathbb{U}, dist)$ where *dist* is non-metric is thus called a non-metric space.

If postulates 2.1 to 2.3 are satisfied and the triangle inequality does not hold, the distance is called a *semi-metric*. The distance *dist* is called a *quasi-metric* if postulate 2.3 and thus symmetry is the only postulate which is not satisfied; *dist* is a *pseudo-metric* if instead of postulate 2.2

[18] Note that non-negativity and thus postulate 2.1 in fact follows from postulates 2.2, 2.3, and 2.4 [Bryant, 1985, *p.* 13].

$dist(x, x) = 0$ for each $x \in \mathbb{U}$ holds together with the other postulates [Hattori, 2003; Skopal and Bustos, 2011, *p.* 34:6][19].

2.2. Distance Measures and their Application

In this section, some basic distance measures are introduced. Some of the presented distance formulas rely on multidimensional vector data with both input vectors having the same number of components δ.

In the context of CBIR, these distance measures can be used for comparing (feature) histograms (see Rubner et al. [2000]) which are used in the experimental evaluations in chapter 4 and chapter 5. In contrast to histograms, (feature) signatures (see Rubner et al. [2000]), that is, lists of cluster center and corresponding weight pairs, are also frequently used in CBIR. The interested reader is referred to Beecks [2013] and Beecks et al. [2013] and earlier work from those authors. Beecks et al. [2013] also contains an empirical analysis of search effectiveness comparing signatures and bag of visual words (BoVW) approaches (for the concept of visual words see Sivic and Zisserman [2003] and its origin in Julesz [1981]).

Some of the distance measures outlined in the following are mentioned or used in the remainder of this thesis. The purpose of this section is to give a brief overview on the variety of distance measures and metrics. If not stated otherwise, our introduction of distance measures with a focus on distance metrics is based on the works of Skopal and Bustos [2011, *ch.* 3.1] and Zezula et al. [2006, *p.* 9ff.].

There are many domain-specific solutions as well as a sheer mass of general metrics for different feature representations. A comprehensive overview on distance measures in different domains can for example be found in Deza and Deza [2009].

[19] Recently, Ptolemaic indexing [Hetland, 2009a] and Ptolemaic access methods (PtoAMs) [Hetland et al., 2013] were introduced. PtoAMs can be applied in case of Ptolemaic distances where postulates 2.1 to 2.3 hold and postulate 2.4 is substituted by Ptolemy's inequality: $dist(w, z) \cdot dist(x, y) \leq dist(w, x) \cdot dist(y, z) + dist(w, y) \cdot dist(x, z)$ for all $w, x, y, z \in \mathbb{U}$. If the triangle inequality additionally holds, the Ptolemaic distance is called a Ptolemaic metric. The validity of Ptolemy's inequality which is for example the case for the family of quadratic form distances and thus also for the well known Euclidean distance (see section 2.2) allows the application of certain pruning rules. If Ptolemy's inequality holds in addition to the metric postulates, additional pruning rules can be applied, possibly leading to increased search efficiency.

Among others, the following two properties are interesting when combining and modifying distance metrics. Multiple metrics can be combined by a weighted sum still preserving metric properties [Novák, 2008, *p.* 19]. Pele and Werman [2009, *sect.* 4] proves that any thresholded distance, $dist^{\psi}(q,o) = min(dist(q,o), \psi)$ with $\psi > 0$ is a metric if *dist* itself is a metric.

2.2.1. Minkowski Distances

The Minkowski metrics $dist_{\mathrm{L}_\alpha}$ ($\alpha \geq 1$) are applicable in δ-dimensional vector spaces:

$$dist_{\mathrm{L}_\alpha}(\vec{q},\vec{o}) = \left(\sum_{i=1}^{\delta} |\vec{q}[i] - \vec{o}[i]|^{\alpha}\right)^{1/\alpha} \tag{2.5}$$

Especially the metrics $dist_{\mathrm{L}_1}$ (Manhattan distance) and $dist_{\mathrm{L}_2}$ (Euclidean distance) are frequently used in CBIR (see e.g. Hu et al. [2008]) and various other domains; $dist_{\mathrm{L}_\infty}(\vec{q},\vec{o}) = \max_{i=1}^{\delta} |\vec{q}[i] - \vec{o}[i]|$ is referred to as the Maximum distance [Skopal and Bustos, 2011, *p.* 34:10].

2.2.2. Quadratic Form Distances

When Minkowski distances are applied, it is assumed that there is no correlation between the feature vector dimensions. In opposition, quadratic form distances model the dependencies of different feature vector components:

$$dist_{\mathrm{qf}}(\vec{q},\vec{o}) = \sqrt{(\vec{q}-\vec{o})^{\top} \cdot S \cdot (\vec{q}-\vec{o})} \tag{2.6}$$

Thus, the $\delta \times \delta$ matrix S captures the pairwise similarities $s_{i,j}$ of feature vector dimensions i and j ($1 \leq i,j \leq \delta$). In CBIR for example, it can be modeled that the similarity between dark green and light green is higher than the similarity between dark green and red. If S is positive semi-definite, $dist_{\mathrm{qf}}$ is a semi-metric; if the matrix is positive-definite, the quadratic form distance is a true metric[20] [Pele and Werman, 2010, *p.* 749f.]. When S corresponds to the identity matrix, the quadratic form distance $dist_{\mathrm{qf}}$ is equal to the Euclidean distance $dist_{\mathrm{L}_2}$. If S

[20] Note also that the signature quadratic form distance, a quadratic form distance defined on feature signatures, is a (Ptolemaic) metric [Lokoč et al., 2011].

corresponds to the inverse of the covariance matrix, the distance is called Mahalanobis distance.

2.2.3. Earth Mover's Distances

The earth mover's distance (EMD) [Rubner et al., 2000], an extension of the Mahalanobis distance [Deza and Deza, 2009, *p.* 350], is frequently applied in CBIR and many other domains. For computing the EMD, a transportation problem is solved. If the sums of signature weights are equal and the ground distance (i.e. the distance implied by the cost matrix) fulfills the metric postulates, the EMD itself is a distance metric [Rubner et al., 2000, *p.* 119f.]. Pele and Werman [2009] proposes the use of thresholded ground distances together with a fast algorithm for the computation of an EMD variant. In their study, both, the efficiency and the effectiveness of a CBIR task based on local image features is increased compared to other local feature matching techniques based on alternative EMD variants.

2.2.4. Edit Distances

Distance metrics are also applied to determine the similarity between sequences of symbols. The edit distance counts the minimum number of necessary edit operations to transform one sequence of symbols into another. If insert, delete, and replacement operations on strings are considered, this distance is called Levenshtein distance [Levenshtein, 1966]. There are many variants of the edit distance (see e.g. Zezula et al. [2006, *p.* 12f.]). In case of weighted edit distances, the non-negative weights of the insert and delete operation must be equal to preserve the symmetry property of the distance metric [Zezula et al., 2006, *p.* 13]. Popular applications of edit distances are for example spell checking [Croft et al., 2010, *sect.* 6.2.2] and the matching of amino-acid sequences in bioinformatics. For the latter, a distance metric based on the mPAM substitution matrix modeling substitution costs between proteins can be used [Xu and Miranker, 2004].

Edit distances can also be defined on tree structures (see Bille [2005]). Connor et al. [2011a] proposes a distance metric to measure the structural difference between trees which is based on information theory and in particular Shannon's entropy [Shannon, 1948]. In Chandrasekaran et al. [2008], for example user profiles are modeled as concept trees which

are matched against document profiles in order to recommend relevant research papers to interested authors. Tree edit distances are also applied to measure the structural difference between XML documents (for references see Zezula et al. [2006, *p.* 13]).

There are also metric variants of the edit distance for graph structures supporting for example search for similar images [Berretti et al., 2007], videos [Lee, 2006], business process models [Kunze and Weske, 2011], or function-call graphs to detect malware programs [Hu et al., 2009].

2.2.5. Hamming Distance

The Hamming distance [Hamming, 1950] is a distance metric frequently applied on bit strings. It measures the number of bit positions in which the bits of two bit strings differ from each other. This is equivalent to computing a bitwise XOR of both bit strings and then counting the number of bits which are set in the result [Muja and Lowe, 2013, *p.* 404]. It also corresponds to the Manhattan distance (see section 2.2.1) applied on bit strings [Deza and Deza, 2009, *p.* 45].

The Hamming distance can also be computed on text strings. Here, it counts the number of positions in which the characters of both strings differ. Thus, the Hamming distance can be perceived as a particular edit distance (see section 2.2.4) where only replacement operations are permitted.

2.2.6. Squared Chord, Hellinger, and Matusita Distance

Experimental results in Hu et al. [2008] and Liu et al. [2008] suggest, among others, the squared chord distance $dist_{\text{sc}}$ for improving the search effectiveness in CBIR. The squared chord distance is only applicable in case of non-negative feature vector components and the formula for computing $dist_{\text{sc}}$ is given as follows (see Hu et al. [2008] and Liu et al. [2008]):

$$dist_{\text{sc}}(\vec{q}, \vec{o}) = \sum_{i=1}^{\delta} \left(\sqrt{\vec{q}[i]} - \sqrt{\vec{o}[i]} \right)^2 \tag{2.7}$$

While $dist_{\text{sc}}$ is a non-metric distance function, there are distance metrics which are conceptually similar to the squared chord distance. According to Deza and Deza [2009, *p.* 249], the square root of the squared

chord distance $dist_{\mathrm{M}}(\vec{q},\vec{o}) = \sqrt{dist_{\mathrm{sc}}(\vec{q},\vec{o})}$ sometimes denotes the Matusita distance while $dist_{\mathrm{H}}(\vec{q},\vec{o}) = \sqrt{2 \cdot dist_{\mathrm{sc}}(\vec{q},\vec{o})}$ is called Hellinger distance[21]. If applied for the distance calculation between two probability distributions, the Hellinger distance $dist_{\mathrm{H}}$ is a metric [Deza and Deza, 2009, *p.* 249].

2.2.7. Distance Functions based on Cosine Similarity

The cosine similarity $sim_{\cos}$ is a popular measure in text retrieval applications to quantify the similarity between a document and a query when applying the vector-space model [Manning et al., 2008, *sect.* 6.3]. A distance measure $dist_{\cos}$ based on the cosine similarity $sim_{\cos}$ is sometimes used in CBIR (see e.g. Hu et al. [2008] and Liu et al. [2008]):

$$dist_{\cos}(\vec{q},\vec{o}) = 1 - sim_{\cos}(\vec{q},\vec{o}) = 1 - \frac{\sum_{i=1}^{\delta} \vec{q}[i] \cdot \vec{o}[i]}{\sqrt{\sum_{i=1}^{\delta} \vec{q}[i]^2 \cdot \sum_{i=1}^{\delta} \vec{o}[i]^2}} \quad (2.8)$$

A distance metric based on cosine similarity which is called the angle distance can be defined by applying *arccos* on $sim_{\cos}$ [Skopal and Bustos, 2011, *p.* 34:10]:

$$dist_{\mathrm{angle}}(\vec{q},\vec{o}) = arccos(sim_{\cos}(\vec{q},\vec{o})) \quad (2.9)$$

Since the arc cosine function *arccos* is strictly decreasing in the interval $[-1, 1]$, $dist_{\mathrm{angle}}$ preserves the ranking of $dist_{\cos}$. Thus, MAMs can be applied in the context of popular text retrieval models. Skopal and Moravec [2005] for example shows that MAMs can be beneficial for latent semantic indexing (LSI) where query vectors are no longer sparse and thus the use of inverted files during query processing becomes inefficient.

2.2.8. Jaccard Distance

The Jaccard coefficient which determines the similarity of two sets A and B by $sim_{\mathrm{J}}(A,B) = \frac{|A \cap B|}{|A \cup B|}$ is frequently used in IR such as for du-

[21] Note that the technique of applying a concave function to the distance values preserves the ranking but usually increases the intrinsic dimensionality of the dataset and thus makes the indexing task harder [Skopal, 2007]. For an explanation of the concept of the intrinsic dimensionality, see section 2.5.

plicate detection (see Naumann and Herschel [2010, *sect.* 3.1.1]). Since $sim_{\mathrm{J}}(A, B) \in [0, 1]$, $dist_{\mathrm{J}}(A, B) = 1 - sim_{\mathrm{J}}(A, B)$ provides a distance measure with $dist_{\mathrm{J}}(A, B) \in [0, 1]$. It can be proven that $dist_{\mathrm{J}}$ is a metric (see e.g. Xu and Agrafiotis [2003, appendix A]).

2.2.9. Canberra Distance

The Canberra distance $dist_{\mathrm{Canb}}$ is a true distance metric [Barioni et al., 2011, *p.* 330]. It is defined in Kokare et al. [2003] as:

$$dist_{\mathrm{Canb}}(\vec{q}, \vec{o}) = \sum_{i=1}^{\delta} \frac{|\vec{q}[i] - \vec{o}[i]|}{|\vec{q}[i]| + |\vec{o}[i]|} \tag{2.10}$$

Kokare et al. [2003] points out that $dist_{\mathrm{Canb}}$ can especially be useful in CBIR based on texture analysis where the Canberra distance can outperform the Euclidean and Mahalanobis distances.

2.2.10. Kullback-Leibler Divergence and its Variants

The Kullback-Leibler divergence [Kullback and Leibler, 1951] measures the difference between two probability distributions [Manning and Schütze, 1999, *p.* 72]. In case of two histograms, the distance can be calculated as follows (see e.g. Hu et al. [2008]):

$$dist_{\mathrm{KL}}(\vec{q}, \vec{o}) = \sum_{i=1}^{\delta} \vec{q}[i] \log \frac{\vec{q}[i]}{\vec{o}[i]} \tag{2.11}$$

The Kullback-Leibler divergence is for example applied in text retrieval for the comparison of language models (LMs) (see e.g. Zhai [2008, *sect.* 5.2] and Shokouhi and Si [2011, *p.* 30]) as well as in various CBIR applications (see e.g. Do and Vetterli [2002] and Liu et al. [2008]). A symmetric variant of the Kullback-Leibler divergence called Jenson-Shannon divergence or Jeffrey divergence is obtained through the following formula [Deselaers et al., 2008, *p.* 84]:

$$dist_{\mathrm{JS}}(\vec{q}, \vec{o}) = \sum_{i=1}^{\delta} \left(\vec{q}[i] \log \frac{2 \cdot \vec{q}[i]}{\vec{o}[i] + \vec{q}[i]} + \vec{o}[i] \log \frac{2 \cdot \vec{o}[i]}{\vec{o}[i] + \vec{q}[i]} \right) \tag{2.12}$$

To obtain a true distance metric fulfilling also the triangle inequality, the square root of $dist_{\mathrm{JS}}$ must be taken (for a proof see Endres and Schindelin [2003]). This property is for example exploited in Wu et al. [2012] to measure the distance between two concepts by comparing their latent topic visual LMs.

2.2.11. Semantic and Other Distance Metrics

Section 2.2.4 already mentions some distance measures for determining the similarity of tree or graph structures. Edit distances measure the work needed to transform one feature object into another.

Semantic distances usually also exploit concept hierarchies and relationships defined in trees and graphs. Two distance metrics employing concept hierarchies and thus information obtained from tree structures are for example used in Lodi et al. [2008]. WordNet [Miller, 1995], a lexical database for the English language, is applied as a knowledge base. The distance between two concepts is influenced by for example the height of the hierarchy, the depth of the least common ancestor of two concepts, and their individual depths within the concept hierarchy.

A conceptually similar approach based on the comparison of tree branch lengths is applied in bioinformatic applications where dissimilarities are for example measured by the UniFrac [Lozupone and Knight, 2005] distance metric.

As a very general approach, Connor et al. [2011b] proposes the ensemble distance, a distance metric applicable to ensembles. These can be interpreted as sets of event-probability pairs and are widely applicable. For example unordered data trees, texts, or images can be compared by the ensemble distance.

2.3. Metric Space Partitioning

After having presented various distance metrics—necessary for the applicability of MAMs—this section will focus on how to index a database under the metric space assumption.

Various space partitioning schemes can be applied and combined for the design of MAMs. In metric space indexing, feature objects and distance information are used for partitioning the feature space since it cannot be assumed that coordinate information is available which is particularly helpful when designing SAMs.

Metric space partitioning schemes are not only used for the design of centralized MAMs. In the context of distributed indexing, the partitioning provides the basis for the assignment of certain regions of the feature space to resources in order to determine their region(s) of responsibility. Resource description and selection schemes for arbitrary metric spaces are also based on these space partitioning techniques (see section 3.3).

Uhlmann [1991] introduces two basic decompositioning schemes which are denoted in Zezula et al. [2006, *p.* 20f.] as *generalized hyperplane partitioning* (see section 2.3.1) and *ball partitioning* (see section 2.3.2). They are introduced in the following together with several extensions which have been proposed.

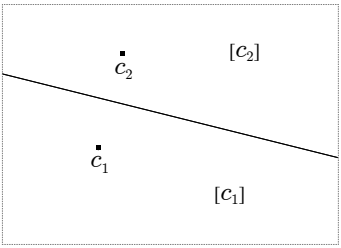
2.3.1. Generalized Hyperplane Partitioning

In order to partition a subset $\mathbb{S}$ of the feature space $\mathbb{U}$ with $\mathbb{S} \subseteq \mathbb{U}$ into two subsets $\mathbb{S}_1$ and $\mathbb{S}_2$ by generalized hyperplane partitioning (see Uhlmann [1991] and Zezula et al. [2006, *p.* 21] and figure 2.1.1), two corresponding reference objects c_1 and c_2 ($c_1, c_2 \in \mathbb{U}$ and $c_1 \neq c_2$) are applied:

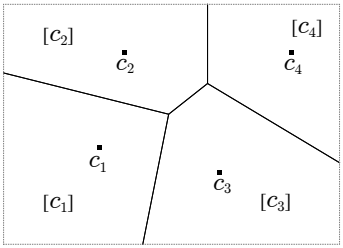
$$\begin{aligned} \mathbb{S}_1 &\leftarrow \{o \in \mathbb{S} \mid dist(c_1, o) \leq dist(c_2, o)\} \\ \mathbb{S}_2 &\leftarrow \{o \in \mathbb{S} \mid dist(c_1, o) \geq dist(c_2, o)\} \end{aligned} \tag{2.13}$$

Database objects with the same distance to both c_1 and c_2 can be assigned to either $\mathbb{S}_1$ or $\mathbb{S}_2$. Since generalized hyperplane partitioning does not guarantee a balanced split, the adequate selection of reference objects (here: c_1 and c_2) becomes important (see section 2.6).

Related to the generalized hyperplane partitioning is lemma 1 which lower-bounds a distance $dist(q, o)$. It is stated and proven in Hjaltason and Samet [2003a, *p.* 539f.]. This lemma is used in section 5.3 when analyzing the resource ranking of the RS4MI approach. Therefore, it is briefly introduced here.



2.1.1 — (Two-way) generalized hyperplane partitioning



2.1.2 — Voronoi-like or multiway generalized hyperplane partitioning

Figure 2.1. — Hyperplane partitioning schemes. For visualization purposes—here and in the remainder of this thesis—a two-dimensional Euclidean space is assumed and the cluster borders are drawn as solid black lines. Subspaces $\mathbb{S}_i$ ($i \in \mathbb{N}^+$) are in some cases of generalized hyperplane partitioning denoted $[c_i]$ emphasizing the use of reference objects c_i as cluster centers.

Lemma 1:
Let $q \in \mathbb{U}$ be a query object and let $o \in \mathbb{U}$ be an object that is closer to reference object c_1 than to reference object c_2, or equidistant from both (i.e. $dist(c_1, o) \leq dist(c_2, o)$). Given $dist(q, c_1)$ and $dist(q, c_2)$, we can establish a lower bound on $dist(q, o)$:

$$max\left(\frac{dist\,(q, c_1) - dist\,(q, c_2)}{2}, 0\right) \leq dist\,(q, o) \tag{2.14}$$

In figure 2.2, the lower bound distance is visualized as a colored line assuming q on a line between c_1 and c_2 in a two-dimensional Euclidean space. The lower bound distance holds for all query objects $q \in \mathbb{U}$ with $dist(q, c_2) < dist(q, c_1)$. It can be shown that the lower bound decreases as q is moved on the dotted line in figure 2.2 (see e.g. q'), that is, with constant distance between q and the partitioning line [Hjaltason and Samet, 2003a, *p.* 539f.].

Voronoi-like partitioning

Voronoi-like partitioning (see figure 2.1.2) is an extension of generalized hyperplane partitioning [Novák, 2008, *p.* 21f.]. Hetland [2009b, *p.* 212] denotes it as "*multiway generalized hyperplane partitioning*" in contrast

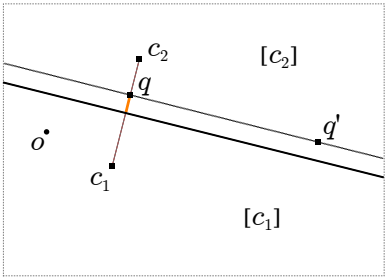


Figure 2.2. — Visualization of the lower-bound distance $(dist(q, c_1) - dist(q, c_2))/2$ on $dist(q, o)$ in case of generalized hyperplane partitioning (adapted from Hjaltason and Samet [2003a, *p.* 540]).

to the "*two-way generalized hyperplane partitioning*" visualized in figure 2.1.1 and outlined above. For a Voronoi-like partitioning, a set with more than two reference objects $C = \{c_i \mid c_i \in \mathbb{U} \wedge 1 \leq i \leq m\}$ and $|C| > 2$ is applied and every database object is assigned to its closest reference object:

$$\begin{aligned} &\forall i \in \{1, \ldots, m\} : \\ &\quad \mathbb{S}_i \leftarrow \{o \in \mathbb{S} \mid \forall j \in \{1, \ldots, m\} : dist(o, c_i) \leq dist(o, c_j)\} \end{aligned} \tag{2.15}$$

Brin [1995] denotes the cells of the Voronoi-like partitioning as Dirichlet domains.

Chávez et al. [2001b, *p.* 296] notes that the task of indexing in general metric spaces can be envisaged as the task of building an equivalence relation which allows to prune certain equivalence classes during search while those which cannot be pruned have to be searched. The partitioning schemes presented in this section directly relate to this problem because a partition of $\mathbb{S}$ induces an equivalence relation "$\sim$" on $\mathbb{S}$ and vice versa (see e.g. Green [1988, *p.* 20f., fundamental theorem on equivalence relations]).

The compact equivalence relation inducing generalized hyperplane partitioning (see Chávez et al. [2001b, *p.* 301]) is outlined in the following since it defines the notation which is used throughout this thesis and because it provides the starting point for the formulation of resource selection algorithms for precise search which extend the approximate

techniques presented in Eisenhardt et al. [2006] (see thesis objective • in section 1.3 which is addressed in section 5.2 and section 5.3):

DEFINITION 3 (compact equivalence relation based on C):
The compact equivalence relation is defined based on a non-empty set of reference objects $C = \{c_i \mid c_i \in \mathbb{U} \wedge 1 \leq i \leq m\}$ by

$$x \sim_C y \iff \mathit{closest}(x, C) = \mathit{closest}(y, C)$$

where $\mathit{closest}(z, C) = \{c_j \in C \mid \forall c_i \in C : \mathit{dist}(c_j, z) \leq \mathit{dist}(c_i, z)\}$. □

The partitioning arises by assigning every database object to the closest reference object. In this thesis, ties (i.e. *closest* returning a set with more than one pivot) are broken consistently and a database object in case of ties is always assigned to the reference object c_i with the smallest ID i ($1 \leq i \leq m$). An equivalence class $[c_i]$ resulting from the space partitioning is in the following denoted as a cluster and $[q]$ for example refers to the cluster where the query lies in (i.e. the query cluster). Reference objects c_i are also called cluster centers.

Permutation-based partitioning

According to Esuli [2009, *p.* 146], permutation-based indexes (PBI) are independently introduced in Chávez et al. [2008] and Amato and Savino [2008]. PBI rely on a permutation-based partitioning of the feature space, that is, a hierarchical form of hyperplane partitioning. Novák and Zezula [2013, *p.* 1] denotes permutation-based partitioning as "*recursive Voronoi-like partitioning*". Based on the non-empty set of reference objects C, a list L_x of cluster IDs can be computed for every feature object $x \in \mathbb{U}$ containing the IDs i of all $c_i \in C$ sorted by increasing $\mathit{dist}(c_i, x)$. L_x is used for determining the cluster to which a feature object x belongs and it is more formally defined by definition 4 according to Chávez et al. [2008, *p.* 1650][22,23].

[22] Note that one-based array and list data types are assumed throughout this thesis. Thus, the indices start with 1 and the last index is equal to the length/size of the data structure.

[23] Note that in the remainder of this thesis L_o and L_q are used to denote that the list is determined for a database object $o \in O$ or a query object $q \in \mathbb{U}$, respectively.

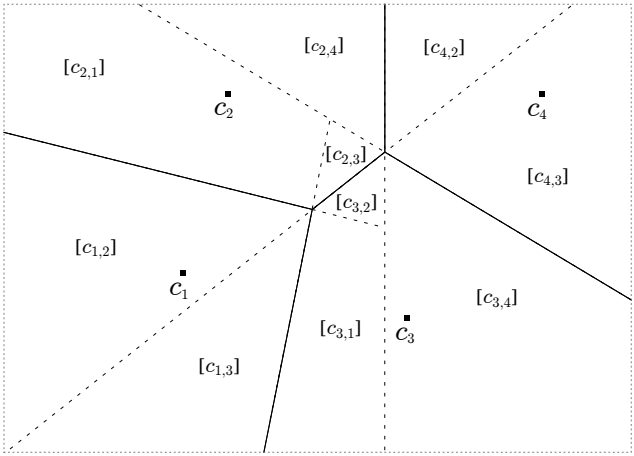


Figure 2.3. — Outline of the permutation-based space partitioning with $m = 4$ and $l = 2$ (adapted from Novák and Batko [2009]). Cluster $[c_{1,4}]$ and cluster $[c_{4,1}]$ do not exist.

Definition 4 (list L_x):
Let $C = \{c_i \mid c_i \in \mathbb{U} \wedge 1 \leq i \leq m\}$ and $x \in \mathbb{U}$. Then, L_x is defined as a permutation of $\{i \mid i \in \mathbb{N}^+ \wedge\ 1 \leq i \leq m\}$ so that, for all $1 \leq i < m$, it holds that either $dist(c_{L_x[i]}, x) < dist(c_{L_x[i+1]}, x)$ or $dist(c_{L_x[i]}, x) = dist(c_{L_x[i+1]}, x) \wedge L_x[i] < L_x[i+1]$. □

The list L_x is sometimes denoted as "*distance permutation*" [Skala, 2009, *p.* 49] or "*pivot permutation (PP)*" [Novák and Zezula, 2013, *p.* 1]. When m centers are used, theoretically $m!$ clusters can arise from the corresponding permutation-based partitioning. In order to restrict this maximum number of clusters to at most $m^{\underline{l}} = m \cdot (m-1) \cdot \ldots \cdot (m-l+1)$ clusters, l-permutations corresponding to the first l elements out of L_x can be applied. Thus, L_x^l in the following denotes the list L_x truncated at position l with $1 \leq l \leq m$.

To give a visual example, figure 2.3 outlines the space partitioning resulting from $l = 2$ with $m = 4$ reference objects being used. Cluster $[c_{3,1}]$ for example denotes that database objects in this cluster are closest to cluster center c_3 and second closest to center c_1. Theoretically, at most $4^{\underline{2}} = 12$ clusters are thus possible in this example. However, not all of them exist. Cluster $[c_{1,4}]$ and $[c_{4,1}]$ are missing in figure 2.3.

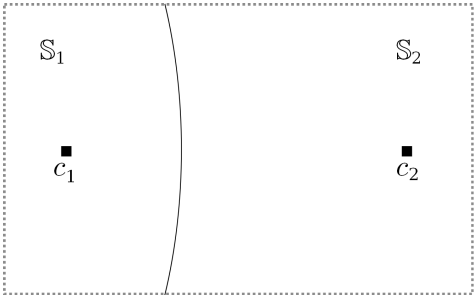


Figure 2.4. — Parameterized generalized hyperplane partitioning with $\xi = -dist(c_1, c_2)/3$; adapted from Lokoč and Skopal [2010, *p.* 131].

Generalized hyperplane partitioning based on the equivalence relation $\sim_C$ can be perceived as a special case of permutation-based partitioning where $L_x^l[1]$ returns the cluster ID of the cluster center which results from $closest(x, C)$ if ties are broken consistently (i.e. in the same way as in case of definition 4).

Parameterized generalized hyperplane partitioning

Lokoč and Skopal [2010] introduces parameterized generalized hyperplane partitioning as a means for obtaining more flexible partitions compared to the general hyperplane partitioning presented above. This is achieved by introducing the parameter ξ as a distance threshold (see figure 2.4).

$$\begin{aligned} \mathbb{S}_1 &\leftarrow \{o \in \mathbb{S} \mid dist(c_1, o) < dist(c_2, o) + \xi\} \\ \mathbb{S}_2 &\leftarrow \{o \in \mathbb{S} \mid dist(c_1, o) \geq dist(c_2, o) + \xi\} \end{aligned} \tag{2.16}$$

However, at the time of writing, Lokoč and Skopal [2010] is not cited by any approach proposing a MAM and it thus seems that no MAM uses the parameterized generalized hyperplane partitioning so far. It is mentioned here for reasons of completeness.

2.3.2. Ball Partitioning

Two basic concepts are related to the partitioning presented in this section: (metric) balls and (metric) (ball) shells. Both are defined in the

following, adapting the definitions of Hetland [2009b, *p.* 208 and *p.* 210, resp.]. Later in this section, ball partitioning and—as an extension of it—excluded-middle partitioning are introduced for partitioning a subset $\mathbb{S}$ of the feature space $\mathbb{U}$ with $\mathbb{S} \subseteq \mathbb{U}$.

Definition 5 (ball):
A ball (region) $[c]_{r^{\text{out}}}$ is defined by its center $c \in \mathbb{U}$ and a corresponding covering radius r^{out}, that is, an upper bound of the distance $dist(c, x)$ from the center c to any object $x \in \mathbb{U}$ in the ball region, i.e. $\forall x \in [c]_{r^{\text{out}}} : dist(c, x) \leq r^{\text{out}}$. □

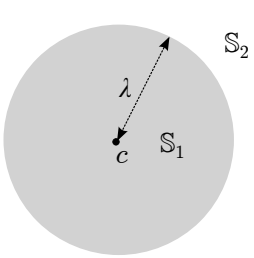
Applying this definition, a query ball $\mathbb{Q}$ (see definition 1 on page 9) in case of a range query $range(q, r)$ can also be denoted as $[q]_r$.

Definition 6 (shell, synonym: hyper-ring):
A shell (region) is the set difference $[c]_{r^{\text{out}}} \backslash [c]_{r^{\text{in}}}$ with $r^{\text{out}} \geq r^{\text{in}}$ of two metric balls. A shell can be represented with a single center c and two radii: the inner radius r^{in} and the outer radius r^{out}; r^{in} and r^{out} are the lower and upper bounds respectively on the distance $dist(c, x)$ for any object $x \in \mathbb{U}$ in the shell, i.e. $\forall x \in [c]_{r^{\text{out}}} \backslash [c]_{r^{\text{in}}} : dist(c, x) > r^{\text{in}} \wedge\ dist(c, x) \leq r^{\text{out}}$. □

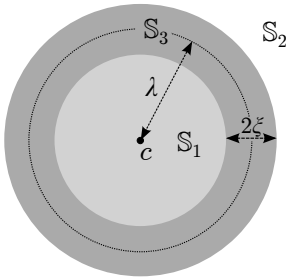
Within ball partitioning [Uhlmann, 1991, *p.* 20] (see figure 2.5.1) $\mathbb{S} \subseteq \mathbb{U}$ is partitioned by a ball $\mathbb{S}_1$ with radius λ around feature object $c \in \mathbb{U}$ (see also Zezula et al. [2006, *p.* 20]). The database objects located outside the ball $\mathbb{S}_1$ are assigned to $\mathbb{S}_2$. Ties, that is, objects on the border of $\mathbb{S}_1$ and $\mathbb{S}_2$ with $dist(c, o) = \lambda$, are arbitrarily but consistently assigned to either $\mathbb{S}_1$ or $\mathbb{S}_2$. Thus, the redundant conditions (i.e. $\leq$ and $\geq$) are included in the following formalization:

$$\begin{aligned} \mathbb{S}_1 &\leftarrow \{o \in \mathbb{S} \mid dist(c, o) \leq \lambda\} \\ \mathbb{S}_2 &\leftarrow \{o \in \mathbb{S} \mid dist(c, o) \geq \lambda\} \end{aligned} \tag{2.17}$$

In tree-based MAMs where a node entry is split when its capacity is exceeded, λ can for example be set to the median distance of the $dist(c, o)$ values of all $o \in \mathbb{S}$ to achieve a balanced split. If the median value is not unique, a balanced split can be achieved by arbitrarily assigning database objects with $dist(c, o) = \lambda$ to one of the two subsets $\mathbb{S}_1$ or $\mathbb{S}_2$ in a balanced fashion [Zezula et al., 2006, *p.* 20].



2.5.1 — Ball partitioning.



2.5.2 — Excluded-middle partitioning.

Figure 2.5. — Ball partitioning schemes; figures adapted from Zezula et al. [2006, *p.* 20]

Excluded-middle partitioning

Excluded-middle partitioning [Yianilos, 1998; Zezula et al., 2006, *p.* 21] can be envisaged as an extension of ball partitioning [Zezula et al., 2006, *p.* 21]. An additional distance threshold ξ is applied to obtain a partition with three elements $\mathbb{S}_1$, $\mathbb{S}_2$, and $\mathbb{S}_3$ (see also figure 2.5.2):

$$\begin{aligned} \mathbb{S}_1 &\leftarrow \{o \in \mathbb{S} \mid \mathit{dist}(c, o) \leq \lambda - \xi\} \\ \mathbb{S}_2 &\leftarrow \{o \in \mathbb{S} \mid \mathit{dist}(c, o) > \lambda + \xi\} \\ \mathbb{S}_3 &\leftarrow \text{otherwise} \end{aligned} \tag{2.18}$$

This partitioning is motivated by the fact that in case of general ball partitioning and range queries $\mathit{range}(q, r)$ with a search radius r where q lies near the border of $\mathbb{S}_1$ and $\mathbb{S}_2$, often both regions $\mathbb{S}_1$ and $\mathbb{S}_2$ must be accessed. In contrast, the existence of $\mathbb{S}_3$ (with a thickness of 2ξ) can potentially lead to a pruning of $\mathbb{S}_1$ or $\mathbb{S}_2$ or even both. When the search radius r is smaller than 2ξ, at least one region can be pruned from search.

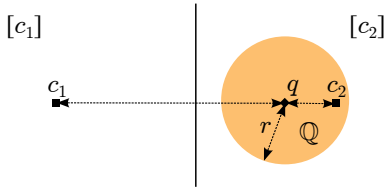


Figure 2.6. — Applying the pruning rule based on the double-pivot distance constraint in case of a range query $range(q, r)$ where cluster $[c_1]$ can be successfully pruned.

2.4. Pruning Rules

The pruning rules outlined in this section are applied by MAMs for cluster pruning and object pruning. In case of cluster pruning (see sections 2.4.1 and 2.4.2), certain regions of the feature space and thus all the objects within can be pruned from search. On the other hand, object pruning (see section 2.4.3) excludes individual database objects from search.

The pruning rules presented in the following are derived from different constraints and corresponding lemmas and proofs which are not in detail outlined here. They can be found in Samet [2006], Hetland [2009b], and Zezula et al. [2006, *p.* 26ff.]. The rules are directly described as applied by IF4MI and RS4MI—assuming a Voronoi-like generalized hyperplane partitioning with the presence of additional ball and/or sphere information. Pruning examples are provided to allow for an intuitive introduction.

2.4.1. Using the Double-Pivot Distance Constraint

If a query lies in the cell of center c^*, that is, reference object c^* is the closest center out of the set C of all available reference objects to a given query object q, any cluster $[c_i] \neq [c^*]$ and hence all objects within the very cluster can potentially be pruned by exploiting the triangle inequality. A cluster $[c_i] \neq [c^*]$ can be pruned if $(dist(q, c_i) - dist(q, c^*))/2 > r$, with r denoting the search radius. Figure 2.6 visualizes two clusters and a range query $range(q, r)$ with $q \in [c_2]$. Cluster $[c_1]$ can be pruned since $(dist(q, c_1) - dist(q, c_2))/2$

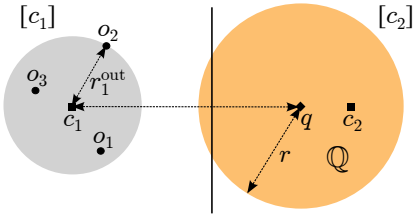


Figure 2.7. — Successful application of the pruning rule based on the range-pivot distance constraint in case of a range query *range*(q, r). Cluster $[c_1]$ can be pruned because *dist*$(q, c_1) - r > r_1^{\text{out}}$ and thus the query ball does not intersect the ball with radius r_1^{out} around c_1.

is bigger than the search radius r and thus the query ball $\mathbb{Q}$ does not intersect $[c_1]$.

2.4.2. Using the Range-Pivot Distance Constraint

If a covering radius r_i^{out} of a cluster $[c_i]$ is given, that is, the maximum distance of any object in the cluster from its center c_i, the very cluster can be pruned if *dist*$(q, c_i) - r > r_i^{\text{out}}$ (see figure 2.7). A similar condition can be applied according to r_i^{in}, that is, the minimum distance of any object within the cluster from its center c_i. We can prune cluster $[c_i]$ if *dist*$(q, c_i) + r < r_i^{\text{in}}$ (see figure 2.8).

The pruning rule based on the range-pivot distance constraint can be used in an inter-cluster way. Two matrices R^{out} and R^{in} are applied to store outer and inner cluster shell radii $r_{i,j}^{\text{out}}$ and $r_{i,j}^{\text{in}}$ respectively for $i, j \in \{1, \ldots, m\}$ where $r_{i,j}^{\text{out}}$ represents the maximum distance from any object out of cluster $[c_i]$ to cluster center c_j and $r_{i,j}^{\text{in}}$ represents the minimum distance from any object out of cluster $[c_i]$ to cluster center c_j. Elements $r_{i,i}^{\text{out}}$ and $r_{i,i}^{\text{in}}$ on the diagonal of the matrices R^{out} and R^{in} thus capture the outer cluster shell radius r_i^{out} and inner cluster shell radius r_i^{in} of cluster $[c_i]$ respectively as already introduced. Cluster $[c_i]$ can be pruned if there exists a cluster $[c_j]$ for which *dist*$(q, c_j) + r < r_{i,j}^{\text{in}}$ or *dist*$(q, c_j) - r > r_{i,j}^{\text{out}}$ [Wojna, 2002, *p.* 301].

Figure 2.9 visualizes a search situation performing a range query *range*(q, r) where cluster $[c_1]$ can be successfully pruned. By solely using the pruning rule based on the double-pivot distance constraint, cluster $[c_1]$ cannot be pruned since the query ball $\mathbb{Q}$ intersects cluster $[c_1]$. If

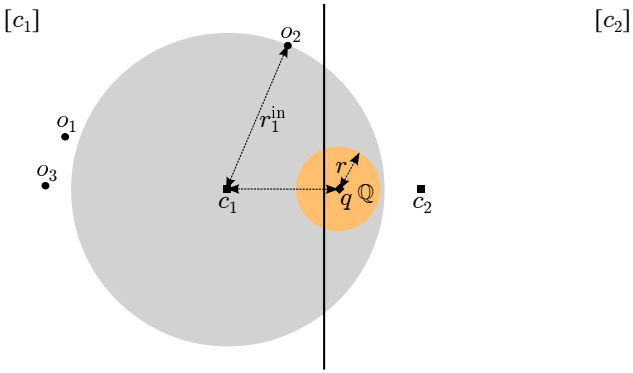


Figure 2.8. — Successful application of the pruning rule based on the range-pivot distance constraint in case of a range query *range*(q, r). Cluster $[c_1]$ can be pruned because *dist*$(q, c_1) + r < r_1^{\text{in}}$ and thus the query ball lies fully inside the ball with radius r_1^{in} around c_1 where—in cluster $[c_1]$—no database objects are lying.

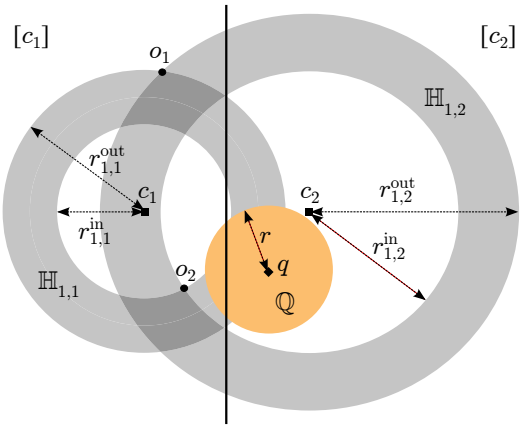


Figure 2.9. — A pruning example showing the usefulness of information stored in the matrices R^{in} and R^{out}; $r_{1,2}^{\text{in}}$ and $r_{1,2}^{\text{out}}$ are used for restricting the region of possible database objects in cluster $[c_1]$ to the two dark gray shaded intersection areas of $\mathbb{H}_{1,1}$ and $\mathbb{H}_{1,2}$. Since the query ball $\mathbb{Q}$ does not intersect any of these regions, cluster $[c_1]$ does not contain any database objects relevant to the query.

for every cluster we administer only the inner and outer shell radius of objects in the cluster (shown by the hyper-ring $\mathbb{H}_{1,1}$ around cluster center c_1 in figure 2.9), cluster $[c_1]$ can still not be pruned. Information from the matrices R^{in} and R^{out} is thus needed for successfully pruning cluster $[c_1]$. If we also apply the radii $r^{\text{in}}_{1,2}$ and $r^{\text{out}}_{1,2}$, that is, the minimum and maximum distance of database objects in cluster $[c_1]$ from c_2, it can be determined that there are in fact no relevant[24] database objects in the intersection area of the query ball $\mathbb{Q}$ and the hyper-ring $\mathbb{H}_{1,1}$. The region of possible database objects is limited to the two dark gray shaded intersection areas of $\mathbb{H}_{1,1}$ and $\mathbb{H}_{1,2}$ and since the query ball $\mathbb{Q}$ does not intersect any of these regions, cluster $[c_1]$ does not contain any database objects relevant to the query.

2.4.3. Using the Object-Pivot Distance Constraint

A further pruning rule can be applied on an object level rather than a cluster level. If $|dist(q, c_i) - dist(c_i, o)| > r$, object $o \in O$ can be pruned without computing $dist(q, o)$. Potentially expensive distance computations between the query object q and a database object o can be avoided at the price of storing distance values $dist(c_i, o)$ which are often anyway computed during the insertion of o into the index structure. The application of this pruning rule is visualized in figure 2.10.

Usually, $dist(c_i, o)$ values are stored for multiple cluster centers c_i. Hence, the computation of $dist(q, o)$ can be skipped if the condition in formula 2.19 is fulfilled. This so called pivot filtering is a direct application of the pruning rule based on the object-pivot distance constraint to multiple pivots.

$$\max_{c_i \in C} |dist(q, c_i) - dist(c_i, o)| > r \qquad \textit{pivot filtering} \tag{2.19}$$

[24] Note that we may use the term (non-)relevant in a different sense than it is used in IR where it is strongly related with the information need concept. We speak of (non-)relevant database objects to indicate that they are (not) part of the final query result. In a similar way, (non-)relevant feature space regions or resources do (not) contain database objects from the final result. This interpretation corresponds with for example Skopal et al. [2012].

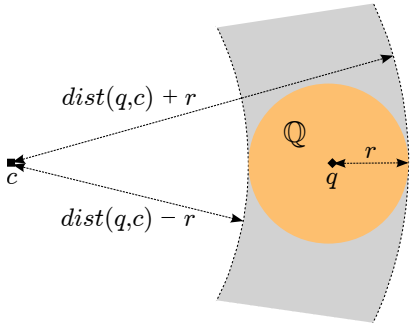


Figure 2.10. — Applying the object-pivot distance constraint (visualization adapted from Hetland [2009b, *p.* 207]): On the one hand, $dist(q,c) - dist(c,o) > r$ holds for database objects inside the inner white ball around center c which has radius $dist(q,c) - r$. On the other hand, $dist(c,o)$ - $dist(q,c) > r$ holds for database objects outside the outer ball around c with radius $dist(q,c) + r$. Thus, $|dist(q,c) - dist(c,o)| \leq r$ holds for database objects which lie inside the gray shaded shell (including the borders) containing the query ball $\mathbb{Q}$. These objects cannot be pruned from search based on c. By using additional reference objects, the region of possible database objects within the search radius can be further restricted through intersections of multiple shells.

2.5. Distance Distributions and the Intrinsic Dimensionality

An important concept for the design and analysis of MAMs is the concept of the intrinsic dimension/dimensionality. In opposition to the representational dimension δ of a δ-dimensional vector space, the intrinsic dimensionality is however an "*elusive concept*" [Chávez et al., 2001b, *p.* 273] and multiple definitions can be found in literature (see e.g. Chávez et al. [2001b, *sect.* 7.1] and Pestov [2007]). Chávez et al. [2001b, *p.* 281] describes it as "*the real number of dimensions in which the points can be embedded while keeping the distances among them*". The intrinsic dimensionality of a dataset can be a helpful concept when quantifying the difficulty of a metric space indexing task. In addition, the concept of the intrinsic dimensionality is also applied for determining the number of pivots to use within a MAM (see section 2.6), for selectivity and performance estimation of MAMs (see Traina Jr. et al.

[2000a]), for the selection of the most important feature dimensions in vector datasets (see Traina Jr. et al. [2000b]), and for estimating the search radius of k-NN queries (see Arantes et al. [2003]).

A widely used definition of the intrinsic dimensionality which can be efficiently computed is given in the following formula 2.20 according to Chávez et al. [2001b, *p.* 303]:

$$\rho = \frac{\mu^2}{2\sigma^2} \tag{2.20}$$

The computation of ρ relies on statistics obtained from a distance distribution; ρ increases if the mean μ of the distance distribution increases and/or the variance σ^2 shrinks. Figure 2.11 visualizes the rationale behind this definition. According to ρ, the histogram on the right reflects a higher intrinsic dimensionality than the histogram on the left. When performing a range query $\mathit{range}(q, r)$ and applying pivot filtering, database objects $o \in O$ with $\mathit{dist}(c, o) \in [\mathit{dist}(q, c) - r, \mathit{dist}(q, c) + r]$ cannot be discarded. The amount of database objects which must be exhaustively searched is proportional to the colored area in figure 2.11 which is in fact bigger in case of pivot c_2 than pivot c_1. Thus, if these histograms capture the distance distributions of two pivots c_1 and c_2, selecting c_1 might be the better choice. More objects can be discarded (being proportional to the white area under the curve) because of a larger variance with the histogram being less concentrated around its mean. In addition, there are search scenarios where an increase of the mean distance μ requires a larger search radius r, for example when retrieving a fixed number of database objects. With other things being equal, an increase of r leads to fewer potential for object pruning since the colored area in figure 2.11 increases. This supports the rationale why μ is contained in the numerator of formula 2.20 [Chávez and Navarro, 2005b, *p.* 370].

More thorough introductions of the concept and references to alternative definitions of the intrinsic dimensionality can for example be found in Chávez et al. [2001b, *sect.* 7.1], Hetland [2009b, *p.* 216f.], Pestov [2007], and Mao et al. [2010, *pp.* 26 and 30].

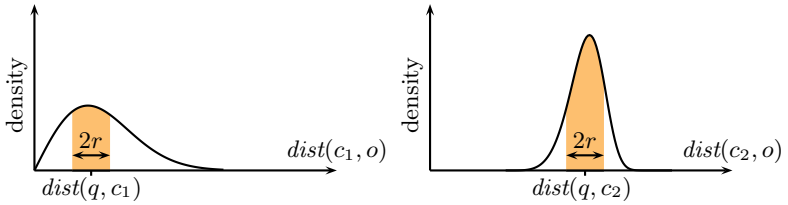


Figure 2.11. — Two exemplary distance histograms (figure adapted from Chávez et al. [2001b, *p.* 302]). According to the definition given in formula 2.20 on the previous page, the histogram on the right reflects a higher intrinsic dimensionality than the histogram on the left.

2.6. Pivot Selection Techniques

An adequate selection of reference objects can boost the pruning power of MAMs and thus improve their efficiency. Both approaches IF4MI and RS4MI rely on the use of pivots. Thus, available pivot selection schemes can be applied. The application of specialized pivot selection strategies to IF4MI and RS4MI is however not addressed in this thesis and an interesting field for future work. IF4MI and RS4MI are not optimized in this regard. In the experiments in chapters 4 and 5, if not mentioned otherwise, pivots are chosen at random.

In the remainder of this section, a brief overview on different pivot selection techniques is given and references are provided for the interested reader. Of course, the fundamental works on metric space indexing address the task of pivot selection (see e.g. Chávez et al. [2001b, *sect.* 7.4] and Zezula et al. [2006, *p.* 63ff.]). There is more recent work on choosing the pivots based on the distance relationships among them [Celik, 2008, *p.* 113]. A brief overview of some techniques in this regard can for example be found in Pedreira and Brisaboa [2007, *sect.* 2] and Socorro et al. [2011, *sect.* 3].

One of those techniques is the Sparse Spatial Selection (SSS) [Pedreira and Brisaboa, 2007]. Since SSS lies at the heart of the SSSTree, both concepts are introduced together in section 3.2.1 on page 49 of this thesis at hand. Mao et al. [2010] presents another conceptually interesting approach based on principal component analysis (PCA). Due to runtime performance reasons, PCA cannot be executed on the full matrix of object-to-object distances. Hence, a predetermined set of

outliers is used as candidate pivots and PCA is applied to the matrix of object-to-pivot distances in order to determine the final set of pivots. A further technique analyzing the distance relationship among pivots and choosing pivots far apart from each other and with similar distances to each other is proposed in the context of the Omni-family[25] of MAMs [Traina Jr. et al., 2007, *sect.* 4.1 and *sect.* 4.2]. An approach for finding good pivots in the field of PBI is for example presented in Figueroa Mora and Paredes [2010].

So far in this section, we have mostly addressed the question which objects to choose as reference objects. Another important question is how many pivots should be selected in order to trade-off storage space and runtime performance of a particular MAM. It might be argued that more pivots lead to more efficient search space pruning. However, the use of more pivots means on the one hand an increase in storage space and on the other more distance computations from the query to the database objects which are necessary upfront, that is, before being able to prune the search space.

The general question how many pivots should be used, which depends on the type of MAM and the applied space partitioning technique, is for example addressed in Amato and Savino [2008, *sect.* 4.1] where the rough heuristic formula is given that the number of reference objects should not be below $2\sqrt{|O|}$ in the context of PBI. Ruiz et al. [2013, *p.* 115] mentions the rule of thumb which "*is to use as many pivots as the available memory allows*" for MAMs solely applying pivot filtering. In contrast, Novák and Zezula [2013] shows that for a permutation-based space partitioning too many pivots can have negative influence on search efficiency. In Traina Jr. et al. [2007, *sect.* 4.1], the number of pivots depends on the intrinsic dimensionality of the dataset.

The task of pivot selection is not only important at index time but also at query time. Pivot selection in Bratsberg and Hetland [2012] determines for each query at runtime which and how many pivots should be applied based on the overlap between the query ball and particular data regions maintained by the utilized access method. Celik [2006, *ch.* 4] empirically shows that good pivots lie either close to or far from the query object. A more detailed description of this approach is part

[25] These kinds of MAMs are outlined in more detail in the Linear Approximating and Eliminating Search Algorithm (LAESA) section 3.2.1 on pages 46–47 of this thesis.

of section 3.2.1 on pages 46–47, where the Linear Approximating and Eliminating Search Algorithm (LAESA) is introduced. Celik [2008, *p.* 114] argues that the technique proposed in Celik [2006, *ch.* 4] is tailored for symmetrical distance distributions. Thus, Celik [2008, *p.* 114] gives a formula for estimating the quality of a potential pivot based on the cumulative distance distribution. The outcome of this formula then guides pivot selection and it is shown that this approach is especially beneficial in case of clustered distance distributions.

Chapter 3.

Metric Access Methods

After having introduced the foundations of metric space indexing, this chapter presents various MAMs. Before doing so, however, the first section 3.1 of this chapter introduces clustering in arbitrary metric spaces since the field of clustering and the design of access methods are closely interrelated. Clustering techniques can be used for building centralized access methods. Data items which are accessed together, for example because they are similar to each other and thus collectively contribute to the search result, can be stored physically close together to reduce the number of disk accesses [Salzberg and Tsotras, 1999, *p.* 167f.]. In addition, partitioning approaches to clustering can be beneficial for search space pruning to reduce the number of distance computations. However, as Chávez et al. [2001b, *p.* 318] notes: "*... it is not clear that good clustering algorithms directly translate into good algorithms for proximity searching.*"

There are centralized access methods where clustering algorithms are directly used. Nistér and Stewénius [2006] presents the vocabulary tree, an access method arising from recursive k-means clustering. Muja and Lowe [2013] uses multiple hierarchical clustering trees with a decomposition inspired by k-medoids clustering. For further references and a description of k-means and k-medoids clustering see section 3.1.

Not only clustering algorithms affect the design of MAMs, also the opposite is the case. This can be observed by the following two examples. MAMs—in its entirety—are used as building blocks of clustering algorithms to increase the efficiency of for example k-medoids clustering (see e.g. Barioni et al. [2008]). In addition, aggregations of data regions (i.e. summaries) which are integral parts of many MAMs are applied by clustering algorithms instead of clustering the feature objects themselves to speed-up the clustering process. The clustering result is then extrapolated from the clustering of the summaries (see e.g. Ganti et al. [1999] and Zhou and Sander [2003]).

From a conceptual point of view, summaries such as cluster balls in the inner nodes of an M-tree [Ciaccia et al., 1997] (see section 3.2.1 on pages 47–48) or computed by the List of Clusters (LC) [Chávez and Navarro, 2005a] (see section 3.2.1 on pages 49–50) can be used as resource descriptions in the field of distributed IR to allow for similarity search in arbitrary metric spaces. An approach from Berretti et al. [2002a,b, 2004] in this regard—based on the M-tree—is for example described in section 3.3.2 on pages 67–68.

Besides introducing clustering in arbitrary metric spaces, this chapter gives an overview on related work in the field of centralized MAMs in section 3.2—both precise and approximate techniques. IF4MI which is presented in chapter 4 is a centralized MAM bridging the gap between precise MAMs on the one hand and approximate MAMs based on inverted files on the other.

MAMs for efficient similarity search in distributed scenarios, in the following denoted as distributed MAMs, are outlined in section 3.3. There, distributed MAMs are described which are related to RS4MI—the resource description and selection scheme for metric space indexing and search which is introduced in chapter 5.

3.1. Clustering in Arbitrary Metric Spaces

Resource descriptions in distributed IR may arise from the application of clustering algorithms. Doulkeridis et al. [2007] and El Allali et al. [2008] for example use k-means clustering.

Conceptually, k-means clustering can be considered as a partitioning approach to clustering [Han and Kamber, 2006, *ch.* 7]. A cluster center in case of k-means is defined as the cluster mean, that is, the mean value of all database objects in the cluster. It is often referred to as the cluster centroid. The general k-means procedure is summarized in figure 3.1 (see Han and Kamber [2006, *p.* 403]).

As many other clustering algorithms, k-means clustering cannot be applied in arbitrary metric spaces. In case of k-means, the calculation of the cluster centers (i.e. the calculation of the cluster mean, see step 3 in the algorithm sketched in figure 3.1) may not be defined. On the other hand, k-medoids clustering which is a variant of k-means [Manning et al., 2008, *ch.* 16.4] can be applied in arbitrary metric spaces. Instead of computing the centroid, the medoid is determined. The medoid of a cluster can for example be defined as the feature object with the

1. Select an initial set $C = \{c_i | 1 \leq i \leq k\}$ of k cluster centers (e.g. randomly).
2. (Re-)Assign each remaining database object $o \in O \backslash C$ to the cluster with center $c^* = \arg\min_{c_i \in C} dist(c_i, o)$.
3. Recalculate the cluster centers.
4. If the cluster assignments have changed, continue with step 2.

Figure 3.1. — Sketch of a basic k-means clustering algorithm adapted from Han and Kamber [2006, *p.* 403].

minimum sum of squared object-to-medoid distances (see e.g. Qiu et al. [2010, *p.* 208]).

There is no unique k-medoids clustering algorithm; k-medoids clustering refers to multiple algorithms. Prominent examples are Partitioning Around Medoids (PAM) [Kaufman and Rousseeuw, 1990, *p.* 68ff.], Clustering LARge Applications (CLARA) [Kaufman and Rousseeuw, 1990, *p.* 126ff.], and Clustering Large Applications based upon RANdomized Search (CLARANS) [Ng and Han, 1994] (see Han and Kamber [2006, *ch.* 7.4.2] for an overview).

The FAst MEdoid Selection (FAMES) extension to k-medoids clustering [Paterlini et al., 2011] is applicable to different k-medoids algorithms. FAMES assures an efficient determination of the cluster medoids (see step 3 in figure 3.1) by applying a pivot-based algorithm which avoids the calculation of all pair-wise distances in each cluster. It is shown in Paterlini et al. [2011] that FAMES can also increase the effectiveness of the clustering compared to previously proposed k-medoids algorithms, since the efficiency gain is not due to the consideration of a random sample of database objects as medoid candidates—which is the approach of some other k-medoids algorithms.

An important issue in k-medoids clustering is how to best determine the number of desirable clusters k which result from the clustering process. There are several ways to do this. A brief overview of different approaches is given in section 5.2.3 of this thesis.

Another configuration option is the selection of the initial medoids (see step 1 in figure 3.1). Celebi et al. [2013] and De Amorim [2013]

analyze different techniques for the initialization of k-means clustering, some of which are also applicable in case of k-medoids clustering.

In the remainder of this section, only clustering algorithms which can be used in arbitrary metric spaces are addressed. Han and Kamber [2006, *ch.* 7.3] distinguishes several types of clustering algorithms, among them: *partitioning*, *hierarchical*, and *density-based methods*. The k-medoids algorithms are partitioning approaches (see Han and Kamber [2006, *ch.* 7.3]). As already mentioned, improving the efficiency of k-medoids clustering is for example addressed by the FAMES algorithm (see Paterlini et al. [2011]). A prominent hierarchical clustering algorithm applicable for clustering in arbitrary metric spaces is for example single-link clustering (see Sibson [1973]). Speeding-up the clustering process in case of hierarchical clustering in arbitrary metric spaces is addressed in Zhou and Sander [2003]. Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [Ester et al., 1996] is a density-based clustering algorithm for clustering in arbitrary metric spaces [Ester et al., 1998]. An efficient incremental version of DBSCAN is presented in Ester et al. [1998].

With respect to the work presented in this thesis, clustering algorithms for arbitrary metric spaces are used for computing resource descriptions. The approach in Berretti et al. [2002a,b, 2004] (for a description see section 3.3.2 on pages 67–68) for example uses a clustering algorithm based on the M-tree (for a description of the M-tree see section 3.2.1 on pages 47–48). Alternatively, k-medoids clustering which is used as an additional comparison baseline in section 5.2 and many other applicable algorithms are also possible such as for example the LC clustering presented in section 3.2.1 on pages 49–50.

3.2. Centralized Metric Access Methods

Hjaltason and Samet [2003a, *p.* 519f.] distinguishes two types of MAMs: *distance-based indexing* and *embedding methods*. MAMs for precise search which rely on distance-based indexing are presented in section 3.2.1 whilst embedding methods are outlined in section 3.2.2.

For example in CBIR, there are search situations where it is sufficient to retrieve only some and not necessarily all of the database objects within a given search radius or just a fraction of the k-NNs. Therefore, MAMs allowing for approximate search have been designed which trade-off result quality and runtime performance (for an overview

see e.g. Patella and Ciaccia [2009]). Some approaches for approximate search related to IF4MI and RS4MI are outlined in section 3.2.3.

3.2.1. Distance-based Metric Access Methods for Precise Search

This section gives a brief overview on distance-based indexing and thus MAMs for precise similarity search which directly use distance information between feature objects for building an index structure. We introduce MAMs which represent important concepts for the remainder of this thesis, either because they are related to IF4MI which also belongs to the group of MAMs relying on distance-based indexing, or because the approaches are used as building blocks of distributed MAMs presented in section 3.3. For more comprehensive surveys, the interested reader is referred to Chávez et al. [2001b], Hetland [2009b], Samet [2006], and Zezula et al. [2006, *p.* 67ff.].

In this section on pages 45–47, MAMs based only on *pivoting* [Hetland, 2009b, *sect.* 9.4], that is, storing distances from database objects to reference objects and pruning database objects during search through pivot filtering based on the precomputed object-to-pivot distances, are introduced. Afterwards on pages 47–50, MAMs are presented which use *aggregation* [Hetland, 2009b, *p.* 203f.], occasionally in addition to pivoting, to structure the feature space into multiple regions in order to prune non-relevant regions during search.

MAMs based only on pivoting

Pivoting is beneficial to reduce the number of distance computations during search. The basic ideas of approaches solely based on pivoting such as for example the Approximating and Eliminating Search Algorithm (AESA) and the LAESA are presented in the following.

MAMs based only on pivoting—if not used as a standalone MAM—also provide important building blocks for MAMs which additionally apply aggregation techniques [Hetland, 2009b, *p.* 208]. Our approaches IF4MI and RS4MI for example use aggregation techniques and can also apply pivoting.

AESA. The AESA [Vidal, 1994; Vidal-Ruiz, 1986] is an indexing technique storing all $\mathcal{O}(|O|^2)$ object-to-object distances. These precomputed distances are used for the pruning of non-relevant database

objects by applying pivot filtering (see formula 2.19 on page 35). During query processing, database objects which have not been pruned so far are iteratively selected as pivots—one pivot per round. A newly selected pivot should lie as close to the query as possible to improve the filtering power. For those database objects $o \in O$ which cannot be pruned, the distance $dist(q, o)$ has to be computed.

Besides its $\mathcal{O}(|O|^2)$ space complexity, the construction time complexity of AESA is also $\mathcal{O}(|O|^2)$ [Chávez et al., 2001b, *p.* 284]. Thus, AESA is applicable when indexing rather small datasets. It can for example also be applied by tree-based MAMs at the leaf level to further reduce the number of necessary distance computations (see e.g. Fredriksson [2007, *sect.* 3.3]). For a long time, AESA had been considered to require the least number of distance computations among all MAMs [Figueroa et al., 2009, *p.* 3.6:3; Socorro et al., 2011, *p.* 1511]. Thus, it is perceived as a best case comparison baseline for various MAMs. Recently, two approaches emerged which claim to be able to outperform AESA in this regard (see Figueroa et al. [2009] and Socorro et al. [2011]).

LAESA. The LAESA [Micó et al., 1994] is proposed to overcome the quadratic space complexity of the AESA. In contrast to the AESA, the LAESA applies a set of m pivots with $m \ll |O|$. Only object-to-pivot distances are stored and used for pruning database objects from search. It is thus important to choose good pivots. Whereas pivots in the LAESA at index time are chosen randomly, multiple alternatives are possible in this regard and the Omni-family of MAMs [Traina Jr. et al., 2007] for example offers one solution suitable for the LAESA (see section 2.6). Another seminal contribution which comes with the Omni-family of MAMs is the extension of the LAESA and its general concept to become applicable together with B-tree variants [Bayer and McCreight, 1972; Comer, 1979] and R-tree variants [Guttman, 1984; Beckmann et al., 1990]—access methods present in many database management systems which can thus make use of metric space indexes.

There are multiple improvements to the LAESA. On the one hand, there are explicit extensions to it. On the other hand, although not explicitly designed for improving the LAESA, techniques proposed in the context of alternative MAMs based on pivoting can be applied (for references and an overview see e.g. Chávez et al. [2001b, esp. *sect.* 5.1.3.2 and *sect.* 8.1] and Hetland [2009b, *sect.* 9.4]).

Memory usage can be reduced by storing the $m \cdot |O|$ object-to-pivot distances with less precision (see e.g. Chávez et al. [2001a] and Figueroa

and Fredriksson [2007]). Alternatively, or in addition, storing only some and not necessarily all of the m object-to-pivot distances per database object can be considered. For references according to these and other coarsening techniques in the context of metric space indexing see for example Figueroa and Fredriksson [2007].

Celik [2006, *ch.* 4.1 and *ch.* 4.2] shows empirically that it is beneficial to prioritize the pivots at query time and to use those pivots first which are "*either close to or far from the query object*" [Celik, 2006, *p.* 29]. As argued in Blank and Henrich [2013a, *p.* 25], this observation analytically follows from the pivot filtering formula 2.19 on page 35: $\max_{c_i \in C} |dist(q, c_i) - dist(c_i, o)| > r$. An analysis of the lower bound $\check{d}_i = |dist(q, c_i) - dist(c_i, o)|$ shows that the resulting absolute value can be high either because $dist(q, c_i)$ is high and $dist(c_i, o)$ is low or because $dist(q, c_i)$ is low and $dist(c_i, o)$ is high. As a consequence, especially medium values for $dist(q, c_i)$ are candidates with a limited potential for selective lower bounds. Thus, it might be sufficient to store only those object-to-pivot distances which lead to selective lower bounds or to consider those precomputed distances first to improve runtime performance.

Note that the query processing of the LAESA is similar to object pruning in the posting lists of the IF4MI approach outlined in chapter 4 of this thesis.

MAMs using aggregation

MAMs using aggregation apply the space partitioning techniques outlined in section 2.3 to structure the feature space. During query processing, pruning rules are applied to exclude clusters from search—and thus all the database objects within (see section 2.4).

MAMs based on the M-tree. The M-tree [Ciaccia et al., 1997] is introduced here because of several reasons. It is conceptually similar to the B-tree [Bayer and McCreight, 1972] and the R-tree [Guttman, 1984] and thus represents a basic MAM sharing many properties with familiar tree-based access methods. In addition, plenty of MAMs have been proposed which are conceptually similar to the M-tree. Hetland [2009b, appendix B] includes those approaches in his analysis such as for example the unbalanced DBM-tree [Vieira et al., 2010]. Furthermore, the M-tree is frequently applied in various domains (see e.g. Berretti et al. [2002a,b, 2004] and Kunze and Weske [2011]).

The M-tree is a balanced tree structure which is built in a bottom-up fashion [Ciaccia et al., 1997, *p.* 430]. Nodes are split if they overflow. Feature objects are administered at the leaf nodes. Inner node entries consist of a routing object, a pointer to a subtree, a maximum distance of objects in the subtree from the routing object, and a distance from the routing object to the routing object of the parent node entry. While traversing the balanced tree structure during query processing, subtrees are pruned if the space they cover does not intersect with the query ball. Also, algorithms for multi-way insertion (see e.g. Skopal et al. [2003]) and bulk loading (see e.g. Ciaccia and Patella [1998]) exist.

The Slim-tree [Traina Jr. et al., 2000] is an extension of the M-tree with modifications to the insertion and node splitting algorithms so that the overlap of the regions covered by node entries is reduced and thus query efficiency can be improved.

The PM-tree [Skopal et al., 2005] can be envisaged as another extension of the M-tree. Additional pivots are applied to support more restrictive pruning. Therefore, subregions of the feature space are represented more precisely by intersections of a ball and multiple shells in contrast to the M-tree where they are represented by only a ball. Pivot filtering is additionally applied at the leaf level where the database objects reside.

Both, the PM-tree and the M-tree with the splitting algorithm of the Slim-tree are evaluated against IF4MI in chapter 4. Furthermore, M-tree-based clustering lies at the heart of the resource selection approach of Berretti et al. [2002a,b, 2004] which provides a comparison baseline for the RS4MI approach in chapter 5.

GNAT. The Geometric Near-neighbor Access Tree (GNAT) [Brin, 1995] applies the Voronoi-like partitioning (outlined in section 2.3.1 on pages 25–26) in every tree node to recursively partition the feature space. Thus, the number of used pivots determines the degree of a node. The hyperplane information of the Voronoi-like space partitioning is however only used for building the structure and not for search space pruning. Here, the pruning rule based on the range-pivot distance constraint is applied. The information for search space pruning maintained for a GNAT node consists of minimum and maximum distances from database objects of a certain Dirichlet domain (i.e. cluster) to the pivots used in the current node. Conceptually, the information maintained for every node can be perceived as information from the R^{in} and R^{out} matrices discussed in section 2.4.2. Similar statistics for

search space pruning as in case of the GNAT can be applied in case of IF4MI (providing the basis for the pruning of posting lists) and RS4MI (applied as resource descriptions and thus allowing for the pruning of resources).

SSSTree. The SSSTree [Brisaboa et al., 2008] uses the SSS dynamic pivot selection technique already mentioned in section 2.6 which adaptively selects inserted database objects as new pivots. The first object is selected by default. Afterwards, a newly inserted database object is chosen as pivot if its distance to any earlier selected pivot lies beyond a certain threshold. This threshold relies on the maximum distance of any two objects in a cluster which is approximated in Brisaboa et al. [2008] as twice the covering radius of the particular cluster.

The GNAT and the SSSTree share some properties. Cluster assignments are in both cases based on a Voronoi-like space partitioning and both MAMs do not apply pivot filtering [Hetland, 2009b, appendix B]. In opposition to GNAT, the SSSTree is however no longer balanced as a consequence of the adaptive pivot selection technique. Furthermore, only the covering radius of the clusters is used by the SSSTree for search space pruning.

In experiments in Brisaboa et al. [2008], the SSSTree outperforms other related techniques such as different variants of the GNAT according to the number of necessary distance computations.

LC. Also the LC [Chávez and Navarro, 2005a] contrasts many balanced tree-based approaches for metric space indexing. The LC can be perceived as a degenerated tree [Tepper et al., 2011, *p.* 9] and it is argued in Chávez and Navarro [2005a] that such an unbalanced approach can be promising for dealing with high-dimensional spaces.

Besides its task as a MAM, the LC provides a ready-to-use clustering technique for arbitrary metric spaces. As its name suggests, the outcome of the LC approach can conceptually be considered as a list of cluster balls $\langle [c_i]_{r_i^{\text{out}}} \mid 1 \leq i \leq |C| \rangle$ with $c_i \in C$. Every cluster in case of the LC is defined by a metric ball $[c_i]_{r_i^{\text{out}}}$, that is, a center c_i and a corresponding covering radius r_i^{out}. There are two general alternatives for building the clusters. They may be of fixed size in terms of the number of database objects contained or of fixed radius where r_i^{out} remains constant for all clusters. During insertion, the list is analyzed from the beginning ($i = 1$) to the end ($i = |C|$). Database objects with $dist(c_i, o) \leq r_i^{\text{out}}$ are inserted into the cluster $[c_i]_{r_i^{\text{out}}}$. Objects for which

$dist(c_i, o) > r_i^{\text{out}}$ remain outside and the next cluster from the list is analyzed. In case of fixed size clusters, overflows must be handled. The resulting list of obtained cluster balls can directly be used as a resource description (see Marin et al. [2009] briefly outlined in section 3.3.5 on pages 74–75).

Initially, the first center is chosen at random. Succeeding centers are chosen during the insertion process maximizing the sum of distances to all previously selected centers [Chávez and Navarro, 2005a]. It is proposed in Téllez and Chávez [2012] to randomly select the centers $c_i \in C$ upfront and assign every database object to the cluster of the closest center to speed-up the construction process of the LC.

During search, the intersection of the query ball and the cluster balls is determined in the order of the clusters contained in the list. When there is no overlap between the current cluster and the query ball, the search continues with the next cluster from the list. If there is partial overlap, the current cluster has additionally to be examined and the search of course also continues with the next cluster. For the original LC, the search can stop as soon as the query ball is fully contained in the current cluster ball.

3.2.2. Embedding Methods for Precise Search

Embedding methods apply a function $map(x)$ which maps the feature object x to a δ-dimensional vector space. Distance computations of $dist'(\vec{x}, \vec{y})$ in the resulting vector space are usually less expensive than distance computations of $dist(x, y)$ in the original feature space. If the mapping is non-expansive, the inequality $dist'(map(x), map(y)) \leq dist(x, y)$ holds for all $x, y \in \mathbb{U}$. Filter-and-refine techniques can then be applied for example in the case of range queries $range(q, r)$ ensuring that all database objects $o \in O$ with $dist(q, o) \leq r$ are found. In the filter step, traditional SAMs can be used and all database objects o with $dist'(map(q), map(o)) > r$ can be safely pruned. The filtered result set may however contain false positives. These can be eliminated in the refine step by checking the actual distances $dist(q, o)$ [Hetland, 2009b; Hjaltason and Samet, 2000, 2003b].

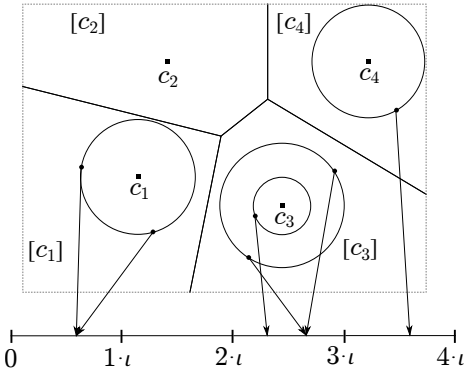


Figure 3.2. — Mapping technique of the Metric iDistance with $m = 4$, under the assumption of a normalized metric space and thus $\iota = 1$; adapted from Novák [2008, *p.* 60].

In the following, the Metric iDistance [Novák, 2008, *p.* 59f.] is presented as one example of a basic embedding technique[26]. The Metric iDistance is an extension of the basic iDistance [Jagadish et al., 2005]. Such extensions lie at the heart of many MAMs. The Metric iDistance is for example used by the M-Index [Novák and Batko, 2009; Novák et al., 2011] which is also presented in this section.

For a more comprehensive overview of embedding methods, the interested reader is referred to Athitsos et al. [2008], Hjaltason and Samet [2003b], and Zezula et al. [2006, *p.* 35f.].

Metric iDistance. The Metric iDistance [Novák et al., 2011, *sect.* 3.1; Novák, 2008, *p.* 59f.] is a generalization of the iDistance [Jagadish et al., 2005] to arbitrary metric spaces. Figure 3.2 visualizes the general mapping technique of the Metric iDistance. First, a set $C = \{c_i \mid 1 \leq i \leq m\}$ of m reference objects is selected. Every database object is then assigned to its closest reference object c_i^* thus obtaining a Voronoi-like space partitioning. A database object $o \in O$ is mapped to a one-dimensional key based on the distance $dist(c_i^*, o)$ from o to its closest reference object c_i^*. The key is obtained by the formula $dist(c_i^*, o) + (i-1) \cdot \iota$.

[26] Approaches using pivot filtering and thus for example the LAESA presented earlier on pages 46–47 can also be perceived as an embedding method. If the mapping is defined as $map(o) = \langle dist(c_i, o) | 1 \leq i \leq |C| \rangle$, the lower bound representing the left side of inequality 2.19 on page 35 can be obtained by $dist_{\mathrm{L}_\infty}(map(q), map(o))$ [Zezula et al., 2006, *p.* 34f.].

In case of a normalized metric space with $dist : \mathbb{U} \times \mathbb{U} \rightarrow [0, 1)$, the separation constant ι is set to 1 and thus the database objects are mapped to one-dimensional keys in the interval $[0, m)$. In other cases, a large enough value of ι ensures that the clusters are well separated. Based on the one-dimensional keys obtained through the mapping technique of the Metric iDistance, insertion and search algorithms of familiar access methods such as the B$^+$-tree [Comer, 1979] can be used for the design of MAMs.

M-Index. Similar to IF4MI and also RS4MI, the M-Index [Novák and Batko, 2009; Novák et al., 2011] combines ball partitioning and generalized hyperplane partitioning and all of the pruning rules outlined in section 2.4 are applicable. The M-Index adopts the idea of permutation-based partitioning (see section 2.3.1 on pages 27–29) where permutations of cluster IDs provide the basis for representing the clusters. A small number of m' reference objects together with an additional parameter l ($1 \leq l \leq m'$) denoting that the l closest centers are used for computing the PP-based representations is sufficient to achieve a very fine-grained cluster structure with a high number of cluster cells. Based on the PP representations L_o^l and thus assignments to clusters, database objects $o \in O$ are mapped to one-dimensional keys which are then indexed in a B$^+$-tree. Here, the Metric iDistance is applied. Thus, the M-Index represents an embedding method[27].

If a cluster cannot be pruned during query processing, the underlying B$^+$-tree is queried with the query ball being mapped to a key interval of the B$^+$-tree. Experiments in Novák and Batko [2009] and Novák et al. [2011] show better performance of the M-Index compared to the PM-tree according to the number of necessary distance computations.

Compared to IF4MI in its basic form which uses a Voronoi-like partitioning with m reference objects, the M-Index applies a permutation-based partitioning with a smaller number of m' reference objects ($m' \ll m$). Thus, the cluster to which a database object belongs can be determined with only m' distance computations in case of the M-Index

[27] Novák et al. [2010] outlines that the M-Index can also be considered as a locality-sensitive hashing (LSH) approach for general metric spaces where database objects close to each other are likely to be assigned to the same bucket. Two other examples of hash-based MAMs for precise search are for example Similarity Hashing [Gennaro et al., 2001] and the D-Index [Dohnal et al., 2003].

whereas IF4MI requires without further optimization m distance computations.

The permutation-based cluster structure of the M-Index is applied on top of the inverted file-based IF4MI approach in section 4.2.5. It appears from the evaluation that IF4MI without the permutation-based partitioning is competitive with the M-Index and that in addition the permutation-based partitioning can also be used with IF4MI.

3.2.3. Metric Access Methods for Approximate Search

There is a wide variety of MAMs for approximate search. Patella and Ciaccia [2009] for example provides a comprehensive survey and many of the techniques already outlined in section 3.2.1 and section 3.2.2 offer approximate extensions which relax search accuracy for an improvement in runtime performance.

Many MAMs for approximate search follow the idea of PBI assuming that a database object $o \in O$ and a query object $q \in \mathbb{U}$ are similar, if their permutations of cluster IDs ordered according to ascending $dist(o, c_i)$ and $dist(q, c_i)$ and possibly truncated at position l are similar, that is, the lists L_o^l and L_q^l (see section 2.3.1 on pages 27–29). In this context, (dis)similarity can be measured by Kendall's tau, the Spearman Footrule Distance, the Spearman Rho Distance, or any other measure for the comparison of two ordered lists (see e.g. Fagin et al. [2003]).

In Esuli [2009], pivot permutation prefixes are indexed in a prefix tree. Query processing based on this so called Permutation Prefix Index relies on matching the pivot permutation prefix of the query to subtrees.

Téllez et al. [2009] transforms the pivot permutations into more space efficient bit vector representations and compares them with the Hamming distance (see section 2.2.5) to speed-up query processing without noticeably losing recall, measured as the fraction of k-NNs found. This technique is used in Téllez and Chávez [2010] where locality-sensitive hashing (LSH) for metric space indexing is proposed.

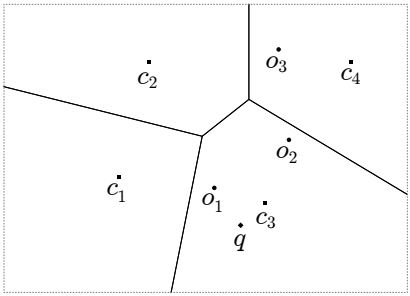
Other approaches for approximate search—some of them also applying permutation-based partitioning—are based on inverted files. In the following, such approaches will be briefly outlined. IF4MI can be perceived as an approach trying to bridge the gap between the approximate approaches based on inverted files on the one hand and MAMs allowing for precise search on the other.

Amato and Savino [2008]. Amato and Savino [2008] proposes an approach which maintains $|C|$ posting lists in total—one list per reference object $c_i \in C$. For a database object $o \in O$ to be inserted, the l_{idx} closest cluster centers are computed, that is, the list $L_o^{l_{\text{idx}}}$ is determined (see section 2.3.1 on pages 27–29). A posting for the very database object is added to all those l_{idx} posting lists. The posting stores the object ID and a weight $t = L_o^{l_{\text{idx}}}(c_i)$ corresponding to the position/rank t of the cluster index i in the list $L_o^{l_{\text{idx}}}$, which indicates that pivot c_i is the t-th closest pivot to the very database object. If a reference object does not belong to the l_{idx} closest reference objects of a database object to be inserted, no posting is generated. In total, $l_{\text{idx}} \cdot |O|$ postings are stored. An example is given in figures 3.3.1 to 3.3.3.

Query processing relies on the computation of $L_q^{l_{\text{q}}}$ and the matching of this query representation against the document representations $L_o^{l_{\text{idx}}}$. The posting lists of clusters with cluster IDs contained in $L_q^{l_{\text{q}}}$ are iteratively scanned in a term-at-a-time fashion (for term-at-a-time query processing in general see e.g. Croft et al. [2010, *sect.* 5.7.2]). Initially, as soon as a document ID occurs for the first time when processing the posting lists, an accumulator associated with each document (i.e. database object) is initialized with $(l_{\text{idx}} + 1) \cdot l_{\text{q}}$. For the matching, a variant of Spearman's Footrule Distance is applied and for each posting $(l_{\text{idx}} + 1) - |L_q^{l_{\text{q}}}(c_i) - L_o^{l_{\text{idx}}}(c_i)|$ is substracted from the accumulator of the very object o.

In total, $|C|$ reference objects are used—the l_{idx} closest to a database object are applied during indexing for determining the document representations $L_o^{l_{\text{idx}}}$, and the l_{q} closest reference objects to the query object are used for determining the query representation $L_q^{l_{\text{q}}}$ with $1 \leq l_{\text{q}} \leq l_{\text{idx}} \leq |C|$. As a further modification, the authors in Amato and Savino [2008] propose to only partially scan the posting lists. These are then sorted by the weights and only a range of postings is scanned where the weights differ by at most some constant value.

To extend the approach from Amato and Savino [2008] for parallel query processing, Mohamed and Marchand-Maillet [2012] analyzes different index partitioning techniques on how to best partition the inverted file. Concepts from text IR such as term and document partitioning are applied where images act as documents and reference objects—or more particular corresponding clusters—act as terms.



3.3.1 — Search example based on a Voronoi-like space partitioning with $|C| = 4$.

$L_{o_1} = \langle 3, 1, 2, 4\rangle$ $L_{o_2} = \langle 3, 4, 2, 1\rangle$ $L_{o_3} = \langle 4, 2, 3, 1\rangle$ $L_q = \langle 3, 1, 2, 4\rangle$

$L^3_{o_1} = \langle 3, 1, 2\rangle$ $L^3_{o_2} = \langle 3, 4, 2\rangle$ $L^3_{o_3} = \langle 4, 2, 3\rangle$ $L^2_q = \langle 3, 1\rangle$

3.3.2 — Construction of the document representations and the query representation assuming $l_{\text{idx}} = 3$ and $l_{\text{q}} = 2$.

$c_1 \rightarrow (o_1,2)$	$dist(q,o_1) = 8 - (4 - \lvert 1 - 1\rvert) - (4 - \lvert 2 - 2\rvert) = 0$
$c_2 \rightarrow (o_1,3),\ (o_2,3),\ (o_3,2)$	$dist(q,o_2) = 8 - (4 - \lvert 1 - 1\rvert) - 0 = 4$
$c_3 \rightarrow (o_1,1),\ (o_2,1),\ (o_3,3)$	$dist(q,o_3) = 8 - (4 - \lvert 1 - 3\rvert) - 0 = 6$
$c_4 \rightarrow (o_2,2),\ (o_3,1)$	$(l_{\text{idx}}+1)\cdot l_{\text{q}}$ $[c_3]$ $[c_1]$

3.3.3 — Inverted file (left) and query processing (right) in case of Amato and Savino [2008].

$c_1 \rightarrow (o_1,2)$	$sim(q,o_1) = 2\cdot 3 + 1\cdot 2 = 8$
$c_2 \rightarrow (o_1,1),\ (o_2,1),\ (o_3,2)$	$sim(q,o_2) = 2\cdot 3 + 1\cdot 0 = 6$
$c_3 \rightarrow (o_1,3),\ (o_2,3),\ (o_3,1)$	$sim(q,o_3) = 2\cdot 1 + 1\cdot 0 = 2$
$c_4 \rightarrow (o_2,2),\ (o_3,3)$	$[c_3]$ $[c_1]$

3.3.4 — Inverted file (left) and query processing (right) in case of Gennaro et al. [2010].

Figure 3.3. — Search example for Amato and Savino [2008] and Gennaro et al. [2010]. Note that database objects are included in the postings instead of object IDs for visualization purposes.

Gennaro et al. [2010]. Gennaro et al. [2010] relies on the ideas presented in Amato and Savino [2008] and thus the parameters discussed above. A posting now stores a virtual term frequency as its weight—similar to the term frequency in traditional text retrieval—instead of a position weight as proposed in Amato and Savino [2008]. Text retrieval programming libraries directly become applicable. The virtual term frequency l_{idx} is assigned to the posting of $o \in O$ in the posting list of the closest center, a value of $l_{\text{idx}} - 1$ is assigned to the posting of o in the posting list of the second closest center, and so on.

Query processing is similar to text retrieval purely based on term frequency and the dot product. The query is transformed so that the closest center gets a weight l_{q}, the second closest a weight $l_{\text{q}} - 1$, etc. A search scenario in case of Gennaro et al. [2010] is visualized in figure 3.3.4.

Sznajder et al. [2008]. Sznajder et al. [2008] presents the Metric Inverted Index where multiple features are indexed for the description of a database object. Clustering is used to determine a set of centers which represent the terms of a feature-specific vocabulary. Every database object is considered as a virtual document. For every feature, object references are added to the posting lists of the e closest centers which are perceived as terms which make up the document.

During query processing, the e closest centers per feature are also determined for the query (i.e. the e query terms). Query modes based on Boolean retrieval are applied. In a strict mode, for all indexed features, one or more query terms must occur in a result document. As a second alternative, in a less strict mode, for at least two features, any of the query terms must occur in a result document. Feature-specific scores (e.g. based on distance information) are combined with an aggregate function (e.g. the sum).

The same authors also propose the Pivots Crossing Approximation [Mamou et al., 2009] which is conceptually similar to the approach in Sznajder et al. [2008] and provides an alternative query mode and in particular the ability to use text as an additional filter criterion.

Note that the use of inverted files for content-based media retrieval was introduced many years ago (see e.g. Müller et al. [1999]). As a main characteristic, the approaches presented in this section do not guarantee precise results. They try to find a balance between adequate retrieval quality and acceptable computational complexity. In this re-

gard, IF4MI (see chapter 4) is considered as a step toward a scalable and efficient retrieval framework which allows both precise and approximate query processing based on inverted files.

3.3. Distributed Metric Access Methods

There is a huge variety of distributed MAMs. According to Mancini et al. [2012, *sect.* III], Papadopoulos and Manolopoulos [2001] is the first approach which studies distributed metric space search. Four different query routing schemes are discussed for answering k-NN queries. Besides sending the query to all resources or accessing them sequentially, multi-step query routing schemes are analyzed where in the first round either the best k documents are obtained from a fraction of the resources or partial results are obtained from all resources.

The following section 3.3.1 provides an overview and discusses the diversity of distributed MAMs. It uses different classification schemes proposed in literature to structure related work in the field and to classify our RS4MI approach. A first classification scheme is based on Gil-Costa et al. [2009, *sect.* 4.2]. It distinguishes *global* and *local indexing* and it is primarily used to demarcate our work. In addition, section 3.3.1 also presents a classification scheme adopted from Lu [2007] for the analysis of different P2P IR systems distinguishing *brokered*, *structured*, *completely decentralized*, and *hierarchical architectures*. We use this second classification scheme and a characterization of different summary types also adopted from Lu [2007] to point out the applicability of resource descriptions in the context of distributed MAMs. Concrete approaches falling into the different groups are then outlined in the subsequent sections 3.3.2 to 3.3.5, respectively.

3.3.1. The Diversity of Distributed Metric Access Methods

In general, search techniques can be distinguished as being *informed* or *uninformed* [Russell and Norvig, 2010, *ch.* 3]. This thesis addresses informed distributed search techniques. Suitable information which can be successfully employed for efficient query routing may be obtained from the content of the resources, from past queries and how they were routed, from relevance feedback, and other sources. In this thesis, we focus on resource descriptions obtained from the content of the

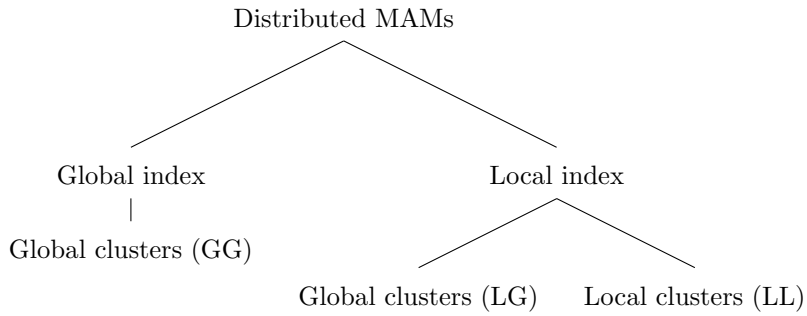


Figure 3.4. — Different types of distributed MAMs; classes GG, LG, and LL adopted from Gil-Costa et al. [2009, *sect.* 4.2].

resources[28]. Uninformed (i.e. blind) search such as for example in case of query flooding (i.e. forwarding the query to all available resources in every routing step; see e.g. Kirk [2003]) or random walk techniques (i.e. forwarding the query to a random subset of available resources in every routing step; see e.g. Lv et al. [2002]) are not addressed in this thesis. The interested reader is for example referred to Sedmidubský [2010, *sect.* 3.1].

Global and local indexing

Distributed MAMs can be built using global or local (document) indexing. Also hybrid combinations are possible. In case of global (document) indexing, that is, a *global index with global clusters* (GG) (see figure 3.4), the database objects of all resources are covered by a single index which is distributed. Every resource administers parts of the index and is thus responsible for a certain region of the feature space or possibly multiple of them. The index distribution is usually based on the space partitioning schemes outlined in section 2.3. In case of precise search, the query is routed to the resources administering regions of the feature space which cannot be pruned from search. Figure 3.5.1 on page 60 displays such a scenario where clusters are assigned to peers

[28] Other approaches, such as Picard et al. [2012] which applies relevance feedback and learns paths to resources with relevant images from past queries, are not considered. Such approaches are only discussed in this thesis if they are interesting from a metric space indexing and search perspective.

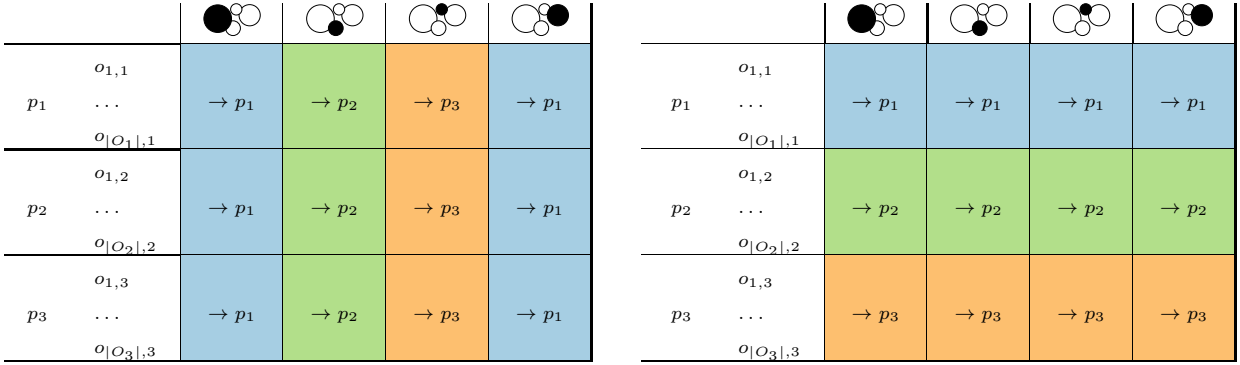
in a round robin fashion. The cluster structure is visualized in the first row of figure 3.5.1. From a pruning perspective, it can of course be desirable to assign clusters lying close together to the same peer(s).

Global indexing is frequently applied in case of P2P IR systems[29]. A prominent example allowing for metric space indexing is M-Chord [Novák and Zezula, 2006] using the mapping technique of the Metric iDistance presented in section 3.2.2 on pages 51–52 to enable the application of Chord [Stoica et al., 2001]. Another approach is the Metric Content-Addressable Network (MCAN) [Falchi et al., 2007] as an extension of the Content-Addressable Network (CAN) [Ratnasamy et al., 2001] for general metric space indexing. There is for example also a distributed extension of the M-Index [Novák et al., 2012] (for the M-Index see section 3.2.2 on pages 52–53) which relies on Skip Graphs [Aspnes and Shah, 2007].

Apart from these already mentioned P2P IR systems relying on distributed hashtables (DHTs) such as M-Chord, MCAN, and the distributed M-Index, distributed MAMs using global indexing can be built on top of various suitable index structures. Some distributed search approaches for example assume a global tree structure with the leaf nodes being distributed to network nodes (see e.g. Liu et al. [2007]). Which nodes are involved in query processing is in such cases determined by the search algorithms of the underlying index structure.

However, this thesis does not address global (document) indexing. Instead, we focus on distributed IR systems relying on local (document) indexing. Two types of local indexing can be distinguished as shown in figure 3.4. A *local index with local clusters* (LL) offers more autonomy for the resources when indexing their documents. Different resources might determine different clusters suitable for representing their data. There is no common agreement on the cluster structure (see figure 3.5.3). In case of a *local index with global clusters* (LG) all resources use the same cluster structure for indexing (see figure 3.5.2). The global cluster structure can be obtained in different ways. Resources can for example negotiate a suitable structure or it can be imposed from external sources.

[29] In another application context, Barrientos et al. [2012] for example analyzes range query processing in multi-GPU environments. One of the analyzed strategies uses cluster balls of the LC approach representing a global index. All GPUs know the entirety of cluster balls and which GPU is responsible for them since in a batch processing scenario any GPU should be able to determine which cluster balls and thus parts of the global index overlap with a query ball.



3.5.1 — Global index with global clusters (GG).

3.5.2 — Local index with global clusters (LG).

3.5.3 — Local index with local clusters (LL).

Figure 3.5. — Visualization of distributed metric space indexing schemes. Peers $p_a\,(1 \leq a \leq |P|)$ maintain documents $o_{j,a}\,(1 \leq j \leq |O_a|)$. Clusters are visualized as circles indicating metric balls, filled black circles denote the particular data region. The notation $\rightarrow p_a$ in combination with the same background color denotes the "responsibility" of a peer p_a.

Similar to PlanetP [Cuenca-Acuna et al., 2003], we assume a scenario with every resource knowing the summaries of all other resources. These distributed sets of resource descriptions can conceptually be perceived as a "*replicated global index*" [Tigelaar et al., 2012, *p.* 9:7]. Aberer et al. [2005] denotes such approaches more precisely as "*federated local* [*document*] *indexes with a global peer index*"—*global* because the global index is at least conceptually the same for all peers and *peer index* because peer descriptions are indexed instead of document descriptions.

In our scenario of a distributed metric space search system (see figure 1.2 on page 11), every resource maintains two types of indexes (see the cylinders in figure 1.2)—a document index mainly for local query processing using IF4MI and a peer index[30] (visualized by an RS4MI cylinder in figure 1.2) to speed-up the resource selection process. The construction of resource descriptions can be based on the local document indexes. During search, promising resources are determined with the help of the peer index where the resource descriptions are indexed and the resource selection techniques presented in chapter 5 are applied. When receiving a query, resources locally process the queries based on their document indexes. Afterwards, search results are sent to the enquirer which then merges them into a ranked list.

This thesis focuses on resource description and selection for metric space indexing and search. Thus, on the one hand, indexing techniques which rely on the metric space approach but send the query to all available resources are out of the scope of this thesis. Such approaches are for example still prominent in the field of metric space indexing and search on modern hardware architectures. Gil-Costa et al. [2009] discusses some indexing schemes for parallel query processing in this regard. It is argued in Gil-Costa et al. [2009, *p.* 9] that sending the query to all processors is feasible because the communication cost in such infrastructures is negligible compared to the cost for the distance computations. Mancini et al. [2012] analyzes indexing and search in an architecture where distributed processors provide the first level of parallelization and threads speed-up query processing at individual multi-core processors. Also here, no resource selection is performed.

[30] Note that the index layout of the peer index is not explicitly addressed in this thesis. The peer index can for example be designed similar to IF4MI (see chapter 4) with a posting list per cluster, but individual postings containing peer information instead of document information and organized in an adequate way (e.g. ordering postings by increasing covering radii of peer-specific cluster balls).

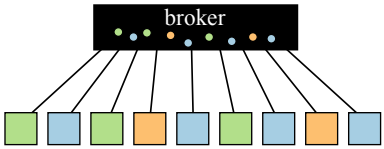
On the other hand, resource description and selection schemes which do not rely on the metric space paradigm are out of the scope of this thesis. In the domain of CBIR, there are for example probabilistic resource selection schemes (see e.g. El Allali et al. [2008] and Nottelmann and Fuhr [2004]) and resource selection schemes for vector spaces (see e.g. Chang et al. [1997], Chang and Zhang [1997], Kim et al. [2002], and Kim and Chung [2003]). Furthermore, distributed query processing in database systems assuming horizontally or vertically partitioned relational data (for references see e.g. Vlachou et al. [2012b]) is out of the scope of our work, if the metric space approach is not particularly addressed and supported.

Network architectures and levels of resource description

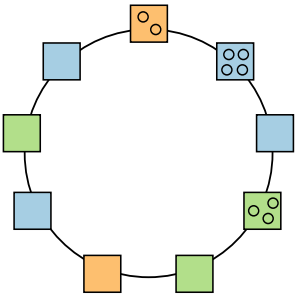
Resource description and selection schemes can be applied in different types of network architectures. Four types of general P2P architectures are identified by Lu [2007]. In the following, we adopt the classification from Lu [2007, *ch.* 2.1.1] to show the wide applicability of resource description and selection techniques. The distinction between a) *brokered*, b) *structured*, c) *completely decentralized*, and d) *hierarchical network architectures* is then also used to classify related work in the field of resource description and selection for similarity search in general metric spaces. Merits and drawbacks of the different network architectures are only briefly discussed here. The interested reader is referred to for example Lu [2007, *ch.* 2].

Brokered architectures (see section 3.3.2). Not only P2P IR systems, but also traditional distributed IR systems for federated text search rely on broker-based architectures (for broker-based architectures in the field of traditional distributed IR see Shokouhi and Si [2011]). The descriptions of participating resources are administered by an information broker (see figure 3.6.1). When a resource issues a query, it is first sent to the broker. The broker then decides—based on the resource descriptions—which of the resources to contact. Usually, the broker sends addresses of promising peers to the enquirer which then issues the query to the selected peers and waits for their results. Without replication, problems arise when the broker fails [Lu, 2007, *p.* 15].

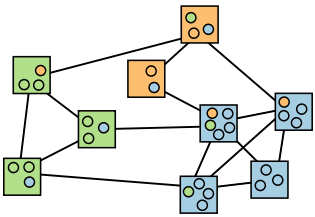
Aberer et al. [2005, *p.* 4] uses the concept of a "*global peer index*" to denote the administration of the set of resource descriptions of



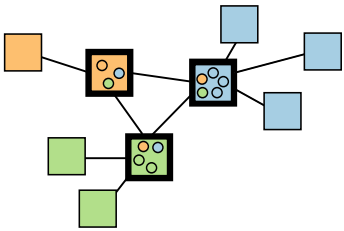
3.6.1 — Brokered.



3.6.2 — Structured.



3.6.3 — Completely decentralized.



3.6.4 — Hierarchical.

Figure 3.6. — Different P2P network architectures according to Lu [2007]; visualization inspired by Lu [2007, figure 2.1 on *p.* 7]. Squares □ denote peers; those with large black borders ◘ are super-peers. Circles ○ visualize resource descriptions. The same fill colors indicate a similar "thematic" focus of peers and/or resource descriptions.

participating peers in a P2P IR system. Since this set of resource descriptions is present in a centralized entity in a brokered P2P architecture, the notion of a *centralized global peer index* also reflects the brokered network architecture.

Structured architectures (see section 3.3.3). Network dynamics with resources frequently entering the system and/or updating their content can lead to network load a P2P IR system relying on global document indexing can hardly cope with [Lupu et al., 2007; Papapetrou et al., 2007; Vu et al., 2009]. Instead of dis-

tributing a global document index, several authors thus propose the indexing of resource descriptions in a distributed index structure such as a DHT. These types of systems are thus characterized by a *distributed global peer index*. Figure 3.6.2 shows an example scenario where similar summaries are administered by the same peer. In this small visual example, replication is not considered and we assume coinciding thematic foci of a peer's summary and the summaries the peer indexes.

Completely decentralized architectures (see section 3.3.4). Completely decentralized P2P IR systems are pure P2P IR systems. They lack the presence of a central authority or super-peers (for a description of super-peer architectures see section 3.3.5 on hierarchical architectures).

Semantic Overlay Networks (SONs) fall into this group of systems. They are introduced in Crespo and Garcia-Molina [2005]. Every peer maintains two types of links to other peers. Peers with similar content are connected by short links and form communities (see figure 3.6.3). Descriptions of peer content can be used in combination with a similarity measure to derive a peer's place within the network topology. In order to do so, clustering, classification, and gossiping techniques are applied [Doulkeridis et al., 2010b]. Long links are established between different communities to ensure the connectivity of the peers. During query execution, the query is forwarded to the most promising communities by multi-hop query routing. A comprehensive overview on different SON approaches is given in Doulkeridis et al. [2010b][31]. Approaches which are applicable for metric space indexing are discussed in section 3.3.4 on pages 69–72.

A second subtype of completely decentralized P2P IR systems assumes that every peer knows the resource descriptions of all other peers in the same (sub)net (see for example the four blue shaded peers in figure 3.6.3 which form such a subnet). Approaches belonging to this group such as PlanetP [Cuenca-Acuna et al., 2003] and its extension Rumorama [Müller et al., 2005b] are in the following denoted as PlanetP-like systems. PlanetP can be

[31] Note that SON may refer to both, the general P2P IR system where peers with for example similar content or query profiles are grouped together as well as the individual groups/communities which form the overlay network.

considered as a *fully replicated global peer index*. Resource description and selection mechanisms in PlanetP-like systems are briefly outlined in section 3.3.4 on pages 72–73. They are conceptually similar to the resource description and selection techniques in a brokered P2P IR architecture.

PlanetP assumes that a peer knows the resource descriptions of all other peers in the system. Rumorama [Müller et al., 2005b] tries to assure scalability by building hierarchies of PlanetP networks. In Rumorama, every peer sees a portion of the network as a single, small PlanetP network and furthermore maintains connections to other peers that see other small PlanetP networks. To this end, the peer stores a small set of links pointing to neighboring peers in other subnets in order to be able to forward queries beyond the boundaries of its own subnet. Each peer can choose the size of its PlanetP network according to local processing power and bandwidth capacity. Within its small subnet, a peer knows resource descriptions of all other peers' data in the same subnet. These descriptions are disseminated by randomized rumor spreading and provide the basis for query routing decisions in the local subnet.

Hierarchical architectures (see section 3.3.5). Hierarchical P2P IR architectures (see Yang and Garcia-Molina [2003] and the visualization in figure 3.6.4 on page 63) are designed to overcome some limitations of other types of P2P IR systems. Super-peers, as opposed to (normal) peers, make use of increased capabilities such as storage capacity, processing power, network bandwidth, or availability. Often, concepts known from other types of P2P IR systems are extended and adapted by hierarchical P2P IR architectures such as for example resource description and selection techniques.

Techniques from the first three types of network architectures outlined in this section can be applied and combined in hierarchical architectures. There are for example hierarchies of brokers (see e.g. Gravano and Garcia-Molina [1995]), super-peer networks based on structured P2P IR architectures (see e.g. Garcés-Erice et al. [2003]), and semantic overlay networks where peers with similar content are assigned to the same super-peer(s) (see e.g. Doulkeridis et al. [2009b]).

In the abovementioned scenarios, from a conceptual point-of-view, resource descriptions can be identified at different levels. The following three-level scheme is also adapted from Lu [2007, *pp.* 39–42].

Resource descriptions of peers. In the most basic form, only the content of an individual peer affects its resource description. An enquiring entity which knows the summary can then decide if it is promising to query the very peer. The query can directly be sent to the peer via single-hop query routing. PlanetP [Cuenca-Acuna et al., 2003] is designed with this goal in mind and summaries for text collections are proposed. Rumorama [Müller et al., 2005b] with its single-hop query routing in the PlanetP leaf nets provides a scalable extension of PlanetP.

Resource descriptions of super-peers. In super-peer networks, those peers which are assigned to a particular super-peer send their resource descriptions to the corresponding super-peer. Multiple resource descriptions can then further be summarize in a single super-peer description. This can then be employed to enable query routing among super-peers.

Resource descriptions of (super-)peer neighborhoods. A neighborhood description of a resource usually provides information in which direction a query should be routed in a multi-hop query routing scenario. Thus, indexing data is summarized along multiple hops which means that the resource descriptions of multiple (super-)peers affect the neighborhood summary. Routing indexes (RIs) [Crespo and Garcia-Molina, 2002] usually fall into this group. Metric space RIs are for example proposed in Doulkeridis et al. [2007] for query-routing among super-peers and in Gennaro et al. [2008] in the context of a completely decentralized system. Both approaches allowing for similarity search in general metric spaces are outlined in the remainder of this thesis.

This thesis focuses on individual peer descriptions. In the following, existing approaches for general distributed metric space indexing and search are recapitulated in this regard. Whenever it is appropriate and additional insights can be gained from the analysis of super-peer or neighborhood descriptions, these are also included in the analysis. The focus is on resource descriptions. Resource selection mechanisms are

only briefly discussed since, at least for precise search, they logically arise from the applicable pruning rules which depend on the type of used resource descriptions. The interested reader is referred to the cited publications.

3.3.2. Resource Selection in Brokered Architectures

There are early resource description and selection schemes for CBIR in brokered network architectures—although not applicable in arbitrary metric spaces—which use clustering algorithms for computing the resource descriptions (see e.g. Chang et al. [1997]). Similarly, clustering algorithms as presented in section 3.1 can be used in combination with the pruning rules outlined in section 2.4 for the design of resource description and selection techniques to allow for precise search in arbitrary metric spaces. In the following, such an approach is outlined as described in Berretti et al. [2002a,b, 2004].

Berretti et al. [2002a,b, 2004]. Berretti et al. [2002a,b, 2004] apply a special form of hierarchical clustering based on the M-tree (see section 3.2.1 on pages 47–48) to the set of database objects of a resource in order to generate a resource description. A cluster radius threshold θ is used for determining the cluster centers which are included in the resource description. Every path in the clustering tree built for the local collection is descended as long as the cluster radius of a node is bigger than the predefined threshold θ. The centers of the nodes where the search stops are included in the resource description. In addition, per cluster, the covering radius and the number of objects within the cluster are stored in the resource description (the latter may be beneficial for ranking peers when performing k-NN queries). By varying θ, the granularity and size of the resource descriptions can be adjusted. It is suggested in Berretti et al. [2004] to use different granularities θ and thus to rely—per resource—on multiple resource descriptions at different granularities. Besides θ, the block size of the M-tree nodes $\beta_{\mathrm{b}}^{\mathrm{M}}$ is the second tuning parameter of this approach, although not explicitly mentioned in Berretti et al. [2002a,b, 2004].

Range queries can be answered by applying the pruning rule based on the range-pivot distance constraint in order to check the overlap of the query ball with the cluster balls of a resource. If the query ball does not overlap any cluster ball, the very resource can be pruned from search. In case of k-NN queries, on a conceptual level, peers are ranked

by the sum of intersection volumes of the query ball and the cluster balls weighted by the number of database objects lying in a cluster.

3.3.3. Resource Selection in Structured Architectures

The idea of indexing local index statistics and thus resource descriptions instead of full index data in a DHT is already pursued by the Minerva system [Bender et al., 2005a,b] for federated text search. Lupu et al. [2007] proposes Hyper-M which is capable of indexing resource descriptions for CBIR in a distributed index structure. Hyper-M targets information sharing in mobile ad hoc networks. It is however not applicable for search in arbitrary metric spaces. SiMPSON [Vu et al., 2009] also indexes summary information in a distributed index structure and additionally allows for search in arbitrary metric spaces, although it is not explicitly designed for this purpose. The SiMPSON system is briefly outlined in the following.

SiMPSON [Vu et al., 2009]. The rationale behind SiMPSON is that every peer locally clusters its data. Although k-means clustering is used in Vu et al. [2009], SiMPSON can be extended for search in arbitrary metric spaces by replacing the k-means algorithm with an algorithm applicable in arbitrary metric spaces. The outcome of the clustering is a set of metric balls per peer. These cluster balls are mapped to one-dimensional key ranges by an extension of the iDistance [Jagadish et al., 2005], conceptually similar to the Metric iDistance outlined in section 3.2.2 on pages 51–52. Every cluster ball is represented by two one-dimensional index keys (i.e. a mapped inner and outer radius of a sphere with a particular reference object as its center and the sphere tightly enclosing the cluster ball). Both index keys are administered as cluster descriptions in the structured P2P overlay instead of full document index data. Note that both index keys may be stored on different peers which indicates that more than one peer is responsible for the very cluster. In addition to the cluster center and the cluster radius, also a peer identifier is associated with a cluster description to indicate the peer to which a cluster belongs. Storing the number of database objects of a cluster can furthermore be beneficial for k-NN query processing.

Every peer is responsible for a certain region of the feature space and has to maintain a so-called "*covering cluster counter*" [Vu et al., 2009, definition 2] which is used during query processing to check if the

search for cluster descriptions matching the query has to be extended to neighboring peers. This counter is increased whenever a cluster—mapped to a key range—fully overlaps the key range for which the peer is responsible.

During search, SiMPSON makes use of the pruning rule based on the range-pivot distance constraint (see section 2.4.2) for determining from the cluster descriptions—obtained from the DHT overlay—those peers where the query has to be sent to. Besides range query processing, algorithms for k-NN queries based on different methods for the estimation of the initial query radius are also described in Vu et al. [2009].

3.3.4. Resource Selection in Completely Decentralized Architectures

This section outlines approaches which make use of resource description and selection techniques within completely decentralized network architectures. The first part addresses SONs which are characterized by a grouping of peers sharing certain characteristics. In the second part of this section, the focus lies on PlanetP-like systems with every peer knowing the resource descriptions of all other resources in a (sub)net used for query routing but not for finding the place of a peer within the network topology.

Resource selection in SONs

MON [Linari and Patella, 2007]. The Metric Overlay Network (MON) extends the idea of SONs toward metric space indexing. Peers with similar content are grouped together and the similarity is determined by a distance metric. Every peer is represented by a metric ball, that is, a reference object and a corresponding covering radius. The experimental study in Linari and Patella [2007] does however not focus on the generation of the MON. It analyzes query routing in a setting with only few clusters/peers. Linari and Patella [2007] notes that every peer can also be represented by a set of balls instead of only a single ball. Query processing is performed by multi-hop query routing and several strategies are evaluated in Linari and Patella [2007]. A promising strategy for example forwards the query in every routing step to the peer in the direct neighborhood with the minimum distance between its cluster ball and the query ball.

Linari and Patella [2007] is related to an approach presented in Linari and Weikum [2006] for distributed text retrieval where a SON is generated by assigning peers to their nearest neighbors (NNs) according to a given distance metric. In Linari and Weikum [2006], peer content is summarized by approximations of peer-specific LMs. The LMs are approximated by Bloom filters and the dissimilarity of different LMs is determined by the square root of the Jenson-Shannon divergence (see section 2.2.10). Although query routing is not analyzed in Linari and Weikum [2006], it is mentioned that the use of a distance metric and thus the application of the triangle inequality allows for the pruning of those peers with no relevant results.

PREGO [Baraglia et al., 2010] and GROUP [Baraglia et al., 2011]. The P2P REcommender system based on Gossip Overlays (PREGO) applies resource descriptions to cluster peers/users with similar interests in a SON. The similarity of user profiles is measured by the Jaccard coefficient (see section 2.2.8). Resource descriptions can for example be derived from the set of visited URLs or the set of movies a user has watched. The process of community building is more explicitly addressed in Baraglia et al. [2011] where the Gossip-based peer-to-peeR cOmmUnity building Protocol (GROUP) is proposed. There, the most frequent terms of a peer's document collection make up the peer profile and again the Jaccard similarity is applied to compare different peer profiles. Both approaches do however not focus on resource selection aspects.

MRoute [Gennaro et al., 2008]. The Multimedia Routing Index (MRoute)[32] is a completely decentralized RI primarily addressing approximate similarity search in general metric spaces. A resource description in case of the MRoute consists at the peer level of $|C|$ histograms—a single histogram per reference object $c_i \in C$. To compute such a histogram, the distance $dist(c_i, o)$ from a particular reference object c_i to a database object $o \in O_a$ is computed for all database objects of a resource $p_a \in P$. The resulting distribution of object-to-pivot distances according to a certain pivot c_i is then quantized into a predefined number of bins. This defines multiple shell regions and

[32] Note that the MRoute is briefly described in this section although initially it is not proposed in the context of a SON. Pre-clustered images are assigned to nodes in a randomly generated tree structure. However, the MRoute is applied for query routing in a SON as described in this section.

a histogram then counts how many database object lie in a particular shell region around reference object c_i. Gennaro et al. [2008] notes that both, the number of histogram bins and the quantization intervals do not necessarily have to be the same for all reference objects. Peer-specific resource descriptions of the MRoute provide the basis for the computation of neighborhood descriptions which are in fact RIs. Query routing is based on the RIs from which the number of potential matches is estimated.

Semantic Peer [Lodi et al., 2008] and its extensions. Lodi et al. [2008] addresses the grouping of peers with similar schemata (i.e. concept hierarchies) in a peer data management system (PDMS)[33]. Strategies where peers can either connect to the most similar peers or all peers in a certain similarity range are proposed. To measure the difference between two concepts, the WordNet-based distance metrics mentioned in section 2.2.11 are used. Every peer can administer multiple semantic concepts. A peer is represented by a so called clustroid which represents the semantic concept minimizing the squared distance from itself to all other concepts the peer administers. The work in Lodi et al. [2008] focuses on efficient overlay construction and addresses maintenance issues. Query routing aspects are addressed in Mandreoli et al. [2006] where the Semantic Routing Index (SRI) is proposed. It indicates for every peer in the neighborhood if promising peers according to a given concept can be reached when following a particular link. Query routing in case of SRIs is based on concept scores maintained for neighboring peers and does not make use of the principles for metric space pruning or pivot filtering outlined in section 2.4. However, Semantic Peer could be extended in this regard and for example the resource description and selection mechanisms proposed in this thesis could be applied.

Gennaro et al. [2011] extends the approach presented in Lodi et al. [2008] by addressing multiple data types. In addition to the concept similarities arising from a schema definition, content-based similarities for example beneficial in the context of multimedia data types are additionally addressed. In the multimedia context, multimedia RIs are proposed which are adopted from the MRoute [Gennaro et al., 2008].

[33] In contrast to P2P IR systems, PDMS [Halevy et al., 2003] assume the existence of possibly heterogeneous peer schemata which every peer may independently define. Local schema mappings enable the reformulation of queries and thus query processing without centralized control [Tatarinov and Halevy, 2004].

MSN [Sedmidubský, 2010]. The Metric Social Network (MSN) is a self-organizing system for approximate similarity search in general metric spaces. Connections between resources are established based on past queries. Attached to a past query are among other things two types of relations—friends relations and an acquaintance relation. While friends are those resources which provided high-quality results (e.g. a non-empty set of result images), the acquaintance is the resource with the best answer for the very query (e.g. the resource providing the most number of result images). In addition, every resource maintains connections which ensure practicable query routing in case of an empty query history or no matching resources.

In order to determine which links to follow, the similarities between the actual query and all the queries from the query history are determined with the help of an appropriate confusability function. Different confusability functions are for example analyzed in Dohnal and Sedmidubský [2009]. The confusability between an actual and a past range query can for example be influenced by the distance between both query objects, both search radii, or the overlap of both query balls.

Resource Selection in PlanetP-like systems

Eisenhardt et al. [2006]. Cluster histograms as resource descriptions are initially proposed in Müller and Henrich [2003] and Müller et al. [2005a]. To compute a cluster histogram, a set with a moderate number of reference objects is applied (e.g. $|C| = 256$). Every feature object of a resource's collection is assigned to the closest reference object and a histogram captures how many objects have been assigned to a certain reference object. Eisenhardt et al. [2006] shows that a random selection of reference objects may replace distributed clustering. Resource selection performance slightly decreases, but network load can be reduced because distributed clustering becomes obsolete.

An improved resource selection technique compared to Müller and Henrich [2003] and Müller et al. [2005a] is proposed in Eisenhardt et al. [2006]. The list L_q of reference object IDs i ($1 \leq i \leq |C|$) sorted in ascending order according to $dist(q, c_i)$ is determined during peer ranking. As a result, the first element of L_q corresponds to the ID of the cluster center being closest to q. A peer with more documents in the corresponding cluster—indicated by the summary—is ranked higher than a peer with fewer documents in the very cluster. If two

peers p_a and p_b with $1 \leq a, b \leq |P|$ and $a \neq b$ administer the same amount of documents in the analyzed cluster, the next element out of L_q is chosen and—based on the indicated number of documents within the very cluster—it is tried to rank peer p_a before peer p_b or vice versa. When the end of the list L_q is reached, a random decision is made.

The resource descriptions of this approach are improved in this thesis (see e.g. Blank et al. [2007] and Blank and Henrich [2010a] and *sect.* 5.1). They are binarized and the number of reference objects is increased to for example $|C| = 8192$ or even more. Compression techniques are applied to prevent a huge increase in average summary size. Furthermore, we extend the approach from Eisenhardt et al. [2006] to precise query processing (see e.g. Blank and Henrich [2013b] and *sects.* 5.2 and 5.3).

In general, there is a convergence of structured and unstructured (i.e. completely decentralized) P2P IR systems with many hybrid approaches. Eisenhardt et al. [2006] proposes an approach where index data is stepwisely transferred among peers to make peers more focused and—as a consequence—summaries more selective. More selective summaries with peers having specialized on a certain region of the feature space lead to more efficient resource selection. Multiple strategies for this cluster specialization are for example evaluated in Eisenhardt et al. [2008].

3.3.5. Resource Selection in Hierarchical Architectures

This section outlines approaches which apply resource description and selection techniques in hierarchical network architectures. Two types of peers coexist in such systems. In addition to normal peers, there are super-peers with increased capabilities.

SIMPEER [Doulkeridis et al., 2007] and its extensions. In SIMPEER, every (normal) peer applies a clustering algorithm on its local data collection and indexes it based on the iDistance [Jagadish et al., 2005]. In the original paper, k-means clustering is applied. However, SIMPEER can easily be extended for metric space indexing by replacing the k-means clustering with an algorithm applicable in arbitrary metric spaces. As its resource description, every peer transfers a list of cluster balls to its super-peer, that is, cluster centers and corresponding covering radii.

When processing a range query $range(q, r)$, it is initially sent to the responsible super-peer (if not the super-peer itself issues the query). Whenever a super-peer receives a query—besides forwarding it to relevant super-peer neighbors[34]—the query is sent to the peers assigned to the very super-peer which maintain feature objects within distance r from q. This is checked by applying the pruning rule based on the range-pivot distance constraint. To speed-up query processing, the cluster descriptions which are sent to a super-peer by its assigned peers are indexed by the very super-peer in an index structure based on the iDistance. Again, this can easily be generalized to arbitrary metric spaces by applying for example the Metric iDistance (see section 3.2.2 on pages 51–52). Besides the processing of range queries, Doulkeridis et al. [2007] also presents techniques for the processing of k-NN queries. In order to do so, two heuristics for estimating the distance from the query object to the k-th NN are presented.

SIMPEER is extended in several directions. Doulkeridis et al. [2009a] addresses scenarios where in case of range queries the user is satisfied by obtaining a certain fraction of the precise search result. This might be tolerated by the user especially in cases when the result set is huge. In such situations, query routing among super-peers can be stopped early based on an analysis of statistics obtained from neighboring super-peers. Query processing costs can thus be reduced without displeasing the user. Doulkeridis et al. [2010a] extends the basic SIMPEER approach with ideas adopted from SONs. Peers with similar content are assigned to the same super-peer. Thus, the number of super-peers involved in query processing can be reduced.

Marin et al. [2009]. Marin et al. [2009] enhances the basic SIMPEER approach presented in Doulkeridis et al. [2007] by introducing an algorithm for the selection of the reference objects. Whereas the reference objects in case of SIMPEER arise from a clustering at the peer level (i.e. every normal peer clusters its data), Marin et al. [2009] uses a predefined set of reference objects which is the same for—at least—all peers assigned to a particular super-peer. The algorithm for determining the set of reference objects iteratively adds a new reference object to the candidate set so that the newly added center maximizes the sum of the distances to previously chosen reference objects. Afterwards, every peer sends its reference objects to the corresponding super-peer. Ap-

[34] Metric RIs are applied for the description of super-peer neighborhoods to allow for multi-hop query routing among super-peers.

plying the same selection mechanism, a super-peer determines a subset of reference objects from all reference objects obtained from its peers. This consolidated set of reference objects is then sent to neighboring super-peers. Again, the selection algorithm is used by the super-peers for determining a set of centers from its centers and the centers of neighboring peers. Not necessarily all of the super-peers have to agree on a unique set of centers. Thus, the centers are called "*semi-global centers*" in Marin et al. [2009, *p.* 125].

Every super-peer sends its semi-global centers to assigned peers. These use the set of centers for indexing and the construction of the resource descriptions. Here, rather than k-means which is used by SIMPEER, Marin et al. [2009] apply the LC clustering algorithm (see section 3.2.1 on pages 49–50). Since the semi-global centers are already known by a super-peer, assigned peers do not have to include the centers into their resource descriptions. Only identifiers of the centers and metric shell radii (i.e. the minimum and maximum distance from a center to objects assigned to it) are included in the summary.

Since shell information is stored in the resource descriptions, the pruning rule based on the range-pivot distance constraint can fully be employed during query processing for checking the overlap between the query ball and cluster shells in contrast to cluster balls only in case of SIMPEER. Among other aspects, also the routing mechanism between super-peers changes compared to SIMPEER.

Vlachou et al. [2010, 2012a]. Also Vlachou et al. [2010, 2012a] modify SIMPEER in several directions. Instead of an index based on the iDistance, every peer maintains an M-tree (see section 3.2.1 on pages 47–48) as its local index. Thus, the resource descriptions which peers send to its super-peer are also modified. A peer uses the cluster balls maintained in the root node of its local M-tree as the resource description. Hence, the particular resource description and selection technique can conceptually be considered as a special case of the approach presented in Berretti et al. [2002a,b, 2004] (see section 3.3.2 on pages 67–68).

Query routing is performed by checking the overlap between the query ball and the cluster balls which represent the peer summaries. To speed-up the peer selection process, a modified M-tree for indexing the peer summaries is maintained at every super-peer. Query routing among super-peers is based on RIs derived from M-tree indexes maintained by the super-peers. Varying block sizes of the M-trees are not

analyzed in Vlachou et al. [2010, 2012a]. It is further unclear if the authors are aware of the approach presented in Berretti et al. [2002a,b, 2004] since it is not cited in Vlachou et al. [2010, 2012a].

Chapter 4.

IF4MI—Inverted File for Metric Indexing and Search

This chapter presents IF4MI [Blank and Henrich, 2012b, 2013a] which brings capabilities for precise search to the MAMs based on inverted files. If IF4MI is applied by resources as their choice of a local document index structure, information already maintained by IF4MI can directly be used as a resource description in a distributed retrieval scenario (see chapter 5). IF4MI is built on top of an inverted file and thus inherently provides a multi-feature MAM with text as an additional search criterion. The main characteristics and benefits of IF4MI are outlined in section 4.1. Afterwards, the applicability of IF4MI is evaluated in section 4.2.

4.1. Outline of IF4MI

The basic idea of IF4MI is straightforward. It uses $m_{\text{if}} = |C|$ pivots and maintains one posting list per pivot containing references to those database objects for which the very pivot is the closest. IF4MI uses all the rules presented in section 2.4 for pruning certain regions of the feature space (i.e. complete posting lists of the structure) or individual feature objects (i.e. postings) from search. The pruning rules in section 2.4 are outlined in the context of range queries. How these pruning rules are used for k-NN queries in case of IF4MI is shown in this section.

Since IF4MI is conceptually simple, it can easily be implemented on top of existing inverted file libraries such as Apache Lucene[35] and benefit from the extensive knowledge in the field of query processing based on inverted files (see e.g. Büttcher et al. [2010] and Zobel and Moffat [2006]). To give just one example, we show how the execution of multi-feature queries combining metric space similarity search

[35] see *http://lucene.apache.org/*, last visit: 12.10.2014

$$R^{\text{in}} = \begin{pmatrix} r^{\text{in}}_{1,1} & \dots & r^{\text{in}}_{1,m_{\text{if}}} \\ \dots & \dots & \dots \\ r^{\text{in}}_{m_{\text{if}},1} & \dots & r^{\text{in}}_{m_{\text{if}},m_{\text{if}}} \end{pmatrix} \quad R^{\text{out}} = \begin{pmatrix} r^{\text{out}}_{1,1} & \dots & r^{\text{out}}_{1,m_{\text{if}}} \\ \dots & \dots & \dots \\ r^{\text{out}}_{m_{\text{if}},1} & \dots & r^{\text{out}}_{m_{\text{if}},m_{\text{if}}} \end{pmatrix}$$

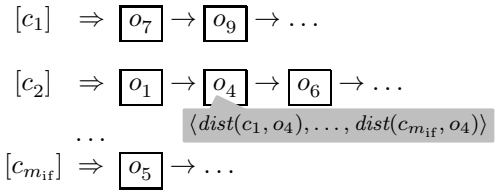


Figure 4.1. — Conceptual outline of the IF4MI structure. Database objects are included in the postings instead of object IDs for visualization purposes.

with a textual filter criterion can make use of the inverted file concept. IF4MI outperforms existing MAMs such as the M-tree and the PM-tree (for both concepts see section 3.2.1 on pages 47–48) in certain scenarios according to the number of necessary distance computations. Additionally, when applying the space partitioning technique of the M-Index (see section 3.2.2 on pages 52–53)—considered as a current state-of-the-art MAM [Lokoč et al., 2014]—the pruning power of the M-Index can be brought to inverted files. All feature objects of a particular cluster are assigned to the same posting list without mapping them to one-dimensional values for storing them in adequate data structures such as a B^+-tree [Comer, 1979]. We also show that the runtime performance of pivot filtering—used by many MAMs to avoid unnecessary distance computations—can be further improved with an adequate heuristic initially proposed in Celik [2006, *ch.* 4].

IF4MI applies a set of $m_{\text{if}} = |C|$ pivots and assigns a database object $o \in O$ to its closest cluster center $c^* = \arg\min_{c_i \in C} dist(c_i, o)$. Cluster IDs are used as virtual terms[36]. Hence, we obtain a vocabulary size of m_{if} (see figure 4.1). During insertion, an object reference is only inserted into the posting list of c^*. Note that every object ID is thus

[36] Note that cluster centers can be perceived as visual words. They are derived from existing reference images. Similarly, from a conceptual perspective, some of our RS4MI resource descriptions presented in chapter 5 can be interpreted as BoVW histograms.

Data:

k	the desired number of NNs
q	the query object
C	the set of reference objects
$topk[\,]$	result array of length k with $\langle oid(o), dist(q,o)\rangle$ pairs, initially: $dist(q,o) = \infty$ for all entries

```
1  L_q = determinePermutationList(q, C)
2  pos = 1
3  while topk[k].dist ≥ (L_q[pos].dist − L_q[1].dist)/2 do
4  |  if clusterPruningNotPossible(L_q[pos], topk[k].dist) then
5  |  |  processPostingList(L_q[pos], topk)
6  |  end
7  |  if pos == |C| then
8  |  |  break
9  |  end
10 |  pos = pos+1
11 end
```

Figure 4.2. — An algorithm for k-NN queries with IF4MI.

contained only in a single posting list of the inverted index. By default, the postings of a posting list are sorted by object/document ID in ascending order.

Two additional matrices R^{in} and R^{out} as explained in section 2.4.2 are administered in main memory and are used for cluster pruning, that is, the pruning of posting lists without processing them. This requires $\mathcal{O}(m_{\text{if}}^2)$ additional space. At the object level and thus for each posting, IF4MI maintains up to m_{if} object-pivot distances $dist(c_i, o)$. By doing so, we are able to apply pivot filtering when traversing posting lists which could not be pruned before.

An algorithm for k-NN query processing is outlined in figure 4.2. First, the list L_q is computed in line 1. Note that L_q for every entry contains the computed $dist(q, c_i)$ value in addition to the cluster ID so that distance values do not have to be recomputed. This is a slight modification to L_q as introduced in definition 4 on page 28. The while-condition in line 3 of the algorithm is a direct application of the

pruning rule based on the double-pivot distance constraint. Following from the fact that L_q is ordered by ascending distance values $dist(q, c_i)$, the search can stop as soon as the condition is not fulfilled for the first time since $L_q[pos].dist$ monotonically increases with increasing pos. In line 4, it is checked if the cluster at position pos of the list L_q can be pruned. Here, the pruning rule based on the range-pivot distance constraint is tested with the help of R^{in} and R^{out} in an intra- as well as an inter-cluster fashion[37]. If a cluster cannot be pruned, its posting list is processed and pivot filtering is applied to the objects whenever possible (using at most m_{if} pivots). If an object cannot be pruned, $dist(q, o)$ has to be computed and $topk$ is possibly updated. Hereby, the current search radius (i.e. $r = topk[k].dist$) used for object and cluster pruning might also be updated.

The algorithm for precise range query processing is straightforward, replacing $topk[k].dist$ in line 3 with the search radius r and $topk$ with a data structure of variable length. So is an extension of IF4MI to approximate search. Instead of testing all $|C|$ clusters (see line 7 of the algorithm outlined in figure 4.2), the algorithm can stop early by analyzing only the first few clusters in the list L_q. A conceptually similar approach in this regard is analyzed in section 5.1.1 on pages 107–108 of this thesis in the context of distributed search.

As mentioned before, the M-Index (see section 3.2.2 on pages 52–53) can be considered as a current state-of-the-art MAM. With respect to the necessary number of distance computations, Lokoč et al. [2014, *p.* 18] denotes the M-Index as "*the currently best MAM*". Its superior performance according to the number of necessary distance computations is influenced by the fact that it applies all the pruning rules outlined in section 2.4. IF4MI uses the same pruning rules. However, an obvious difference between IF4MI and the M-Index is the space partitioning used by these approaches. Thus, we extend IF4MI by applying the permutation-based space partitioning of the M-Index (see figure 2.3 on page 28) and compare it in section 4.2.5 with the initial Voronoi-like space partitioning of IF4MI (see figure 2.1.2 on page 25). We use an algorithm outlined in Myrvold and Ruskey [2001] to map l-permutations which identify the clusters of the M-Index to integers in the range of

[37] Here, the two matrices R^{in} and R^{out} are used since their memory footprint is affordable. The influence of individual pruning rules on the resource selection performance is analyzed in section 5.2.4 in the context of our distributed resource selection scheme RS4MI.

$[1, m^l_{\text{mi}}]$ in order to be able to store inner and outer shell radii in the two matrices R^{in} and R^{out} as before. Query processing is implemented as a k-NN algorithm similar to the one described in Novák et al. [2011] which relies on a priority queue ordering clusters by a penalty score. This penalty score captures the "proximity" between a cluster and the query object.

4.2. Evaluating IF4MI

After having outlined the general concepts of IF4MI, it is now evaluated. First, section 4.2.1 provides an overview of the experimental setup. A quantitative evaluation of IF4MI is presented in section 4.2.2. Section 4.2.3 shows that in certain scenarios IF4MI can outperform the M-tree and the PM-tree—two alternative MAMs (for both concepts, see section 3.2.1 on pages 47–48)—according to the number of necessary distance computations. The memory requirements of the different approaches are analyzed in section 4.2.4. In section 4.2.5, we compare our approach with the M-Index (see section 3.2.2 on pages 52–53) and apply its space partitioning technique to our inverted file-based approach. Many MAMs such as the PM-tree, the M-Index, and IF4MI rely on pivot filtering. In section 4.2.6, we apply a heuristic proposed in Celik [2006, *ch.* 4] to reduce the number of pivots which are tested during pivot filtering. Finally, section 4.2.7 analyzes the benefits of IF4MI when performing multi-feature queries comprised of a k-NN similarity query and an additional textual filter criterion.

4.2.1. Experimental Setup

In the following, we assume a CBIR task. Experiments are based on the CoPhIR dataset [Bolettieri et al., 2009]. Our collection consists of the first 100 000 images from CoPhIR archive no. 1. Twenty runs with varying sets of cluster centers are performed which are randomly chosen from the remaining 900 000 images of the first archive. In each run, we use the same set of 200 query objects randomly selected from CoPhIR archive no. 106 (CoPhIR consists of 106 archives in total.). We perform 200 20-NN queries[38] per run and use the features which come

[38] When evaluating k-NN queries, $k = 20$ is assumed throughout this thesis. 20 is a typical value for k in the context of image retrieval and for example also used in Eisenhardt et al. [2006] and Müller et al. [2005a].

with the CoPhIR dataset (distances in brackets according to Manjunath et al. [2002]): Scalable Color ($dist_{L_1}$), Color Structure ($dist_{L_1}$), Color Layout (weighted $dist_{L_2}$), Edge Histogram (variant of $dist_{L_1}$), and Homogeneous Texture (variant of $dist_{L_1}$).

We use four feature combinations with different intrinsic dimensionality values ρ (see formula 2.20 on page 37): FC_1 (Color Layout only, $\rho = 4.4$), FC_2 (Edge Histogram only, $\rho = 7.7$), FC_3 (Color Layout and Edge Histogram, $\rho = 10.9$), FC_4 (all features, $\rho = 13.7$). In case of FC_3 and FC_4, features are normalized and weighted equally. If not stated otherwise, FC_4 is used in the remainder of this chapter.

4.2.2. The Number of Distance Computations of IF4MI

At first, we consider the number of distance computations as the dominating cost factor. This is frequently the case in the literature on MAMs [Skopal, 2010]. In addition, it is motivated by the fact that IF4MI can be used as a main memory indexing technique because of its potential for a relatively small index size[39]. Of course, IF4MI can also be applied as a disk-based index. In this case, cluster pruning is beneficial to avoid the unnecessary scanning of posting lists which are stored on disk. When clusters cannot be pruned from search and posting lists are accessed, object pruning comes into place to further reduce the number of database objects for which distances $dist(q, o)$ have to be computed.

Figure 4.3 shows the average fraction of distance computations, the average fraction of pruned objects via cluster pruning, and the average fraction of pruned objects via object pruning. 100% correspond to $|O| + m_{\text{if}} = 100\,000 + 1024 = 101\,024$ distance computations because the number of distance computations is always at least $m_{\text{if}} = 1024$ since the list L_q (see the algorithm in figure 4.2) is computed during query processing in any case. $\text{IF}(m_{\text{if}}, m'_{\text{if}})$ will in the following denote the parameter values of IF4MI with m_{if} indicating the total number of reference objects used and m'_{if} ($m'_{\text{if}} \leq m_{\text{if}}$) indicating the number of $dist(c_i, o)$ distances stored per database object $o \in O$ and used for pivot filtering.

In case of IF(1024,1024), based on 1024 cluster centers, 1024 precomputed $dist(c_i, o)$ distances stored in the postings of any database object $o \in O$ are used for object pruning (see figure 4.3). For IF(1024,128),

[39] The memory requirements of IF4MI are analyzed in section 4.2.4.

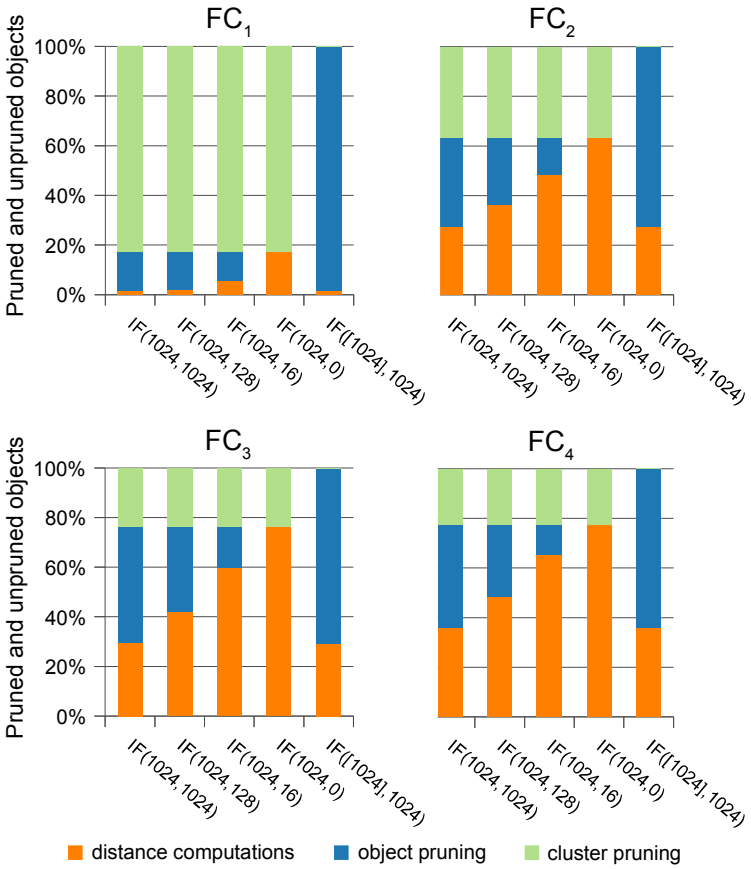


Figure 4.3. — Necessary distance computations and pruned feature objects (via object pruning and cluster pruning) for IF($m_{\text{if}}, m'_{\text{if}}$) [in %]. IF($[m_{\text{if}}], m'_{\text{if}}$)—see the rightmost bar in each group—is a parameter setting with no cluster pruning where all posting lists are processed to prune objects based on pivot filtering.

cluster pruning is still based on the whole set of $m_{\text{if}} = 1024$ centers while only a random subset of $m'_{\text{if}} = 128$ centers and thus 128 precomputed $dist(c_i, o)$ distances are applied for pivot filtering. Of course, reducing the number of distance values stored in the postings for example from 1024 to 128 reduces the memory size of the index and I/O cost in case of the posting lists being stored on secondary memory. But, at the same time, the number of distance computations increases since fewer objects can be pruned. Both, m'_{if} and m_{if} are tuning parameters of IF4MI and a more detailed analysis in this regard is part of the remainder of this chapter.

IF([1024],1024)—corresponding to the rightmost bar in each group in figure 4.3—is a parameter setting similar to LAESA (see section 3.2.1 on pages 46–47) where no cluster pruning is applied at all, although the cluster structure based on 1024 reference objects is used here for indexing. In this case, all posting lists are processed in order to prune objects $o \in O$ based on $m'_{\text{if}} = 1024$ precomputed $dist(c_i, o)$ distances. From figure 4.3, it can be observed that pivot filtering at the posting level is very effective. IF(1024,1024) requires the same amount of distance computations as IF([1024],1024). Nevertheless, in a setting where the posting lists are stored on secondary memory, the latter is more expensive according to disk I/O costs since no database objects are pruned via cluster pruning. In this case, we would like to reduce the number of pruned objects through more effective cluster pruning instead of massively relying on object pruning and thus disk I/O reads. The parameter settings in figure 4.3 with $m'_{\text{if}} < m_{\text{if}}$ represent compromises reducing the memory requirements and disk I/O costs while increasing the number of distance computations. An adequate value for m'_{if} can be found based on the computational complexity of the distance measure, the available storage space, and the general scenario of an index in main or secondary memory.

To give an impression, figure 4.4 shows—for a randomly selected run—the number of postings per posting list. For this particular run, only a single cluster does not contain any postings. We can see that the distribution of the number of postings per cluster is skew. It is of course influenced by the technique for choosing the reference objects.

To further analyze the characteristics of IF4MI, we vary the parameter values for m_{if} and m'_{if}. Figure 4.5 shows the fraction of necessary distance computations compared to a sequential scan for different parameter values of m_{if} and m'_{if}. Results are shown for FC_4. It can be

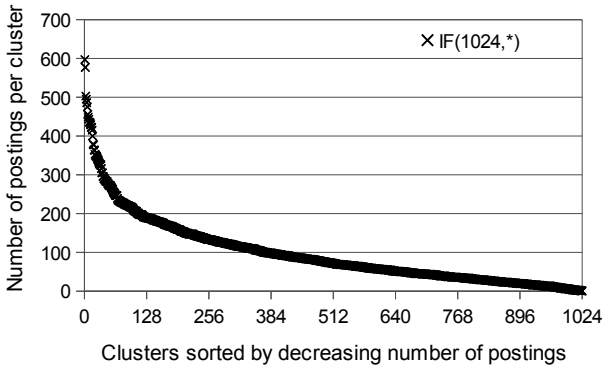


Figure 4.4. — Number of postings per posting list for IF(1024,*) and $|O| = 100\,000$ postings in total. The placeholder $*$ for m'_{if} indicates that these measurements are independent of m'_{if}.

observed that increasing m_{if} and thus the number of clusters only pays off—according to the number of necessary distance computations—if m'_{if} is relatively small (e.g. $m'_{\text{if}} \leq 40$). In these situations, there is a steady decrease in the fraction of distance computations with increasing m_{if}, especially when m'_{if} is very small such as in case of IF(m_{if},1). For bigger numbers of m'_{if} (e.g. $m'_{\text{if}} = 128$), pivot filtering is able to prune large amounts of non-relevant database objects and thus a very small $m_{\text{if}} = 256$ already offers the best efficiency among the measured parameter settings in terms of the number of distance computations. The increase in the fraction of distance computations for $m'_{\text{if}} = 128$ when m_{if} becomes large can be explained by the fact that m_{if} distance computations are already performed per query upfront when computing the list L_q before actually entering the pruning process (see line 1 of the algorithm in figure 4.2 on page 79). From this perspective, the space partitioning used by the M-Index seems to be a suitable alternative when a large number of clusters is desired. It is analyzed in section 4.2.5.

As can be observed from figure 4.5, a large number of clusters is inevitable to reduce the number of distance computations when memory requirements do not permit the use of a sufficiently large number of reference objects for pivot filtering. Otherwise, whenever it is affordable to apply a rather large number of reference objects for pivot filtering (e.g. $m'_{\text{if}} = 128$), the influence of choosing an adequate m_{if} is not that

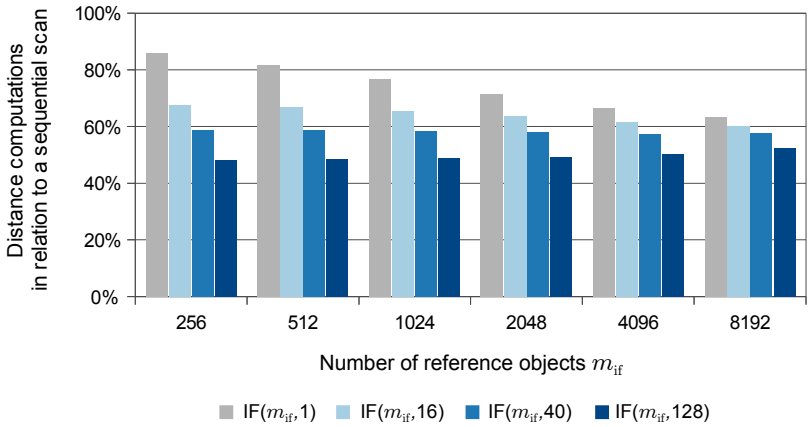


Figure 4.5. — Fraction of distance computations compared to a sequential scan when varying m_{if} and m'_{if} for IF(m_{if}, m'_{if}).

crucial, assuming that distance computations are the dominant cost factor, and even a small $m_{if} = 256$ leads to a reduction in the total number of distance computations; IF(256,128) performs best among the different parameter settings displayed in figure 4.5.

To reduce disk I/O costs when storing the posting lists of IF4MI in secondary memory and to speed-up query processing even in a main memory indexing scenario, it becomes crucial to prune as many clusters as possible from search. Table 4.1 shows for a randomly selected run the average number of pruned clusters for different values of m_{if}. In case of many clusters (i.e. $m_{if} = 8192$), almost 50% of the clusters are pruned. However, the remaining 50% have to be processed. The number of empty clusters is also displayed in table 4.1. Empty clusters can be avoided by an optimized selection strategy for the reference objects replacing the random selection.

Figure 4.6 shows the number of distance computations and the number of database objects pruned from search by cluster and object pruning for IF(m_{if},40) when varying the number m_{if} of reference objects. It can be observed that larger numbers of clusters lead to more objects pruned by cluster pruning. On the other hand, less database objects are pruned by object pruning. A larger number of clusters might thus justify the use of fewer reference objects for object pruning.

	empty clusters	pruned empty clusters	pruned clusters total
$m_{\text{if}} = 256$	0	0	39.5
$m_{\text{if}} = 512$	0	0	104.2
$m_{\text{if}} = 1024$	1	1.0	270.6
$m_{\text{if}} = 2048$	15	13.9	682.2
$m_{\text{if}} = 4096$	76	70.9	1677.9
$m_{\text{if}} = 8192$	395	369.1	3948.2

Table 4.1. — Number of pruned clusters of IF4MI for different values of m_{if}.

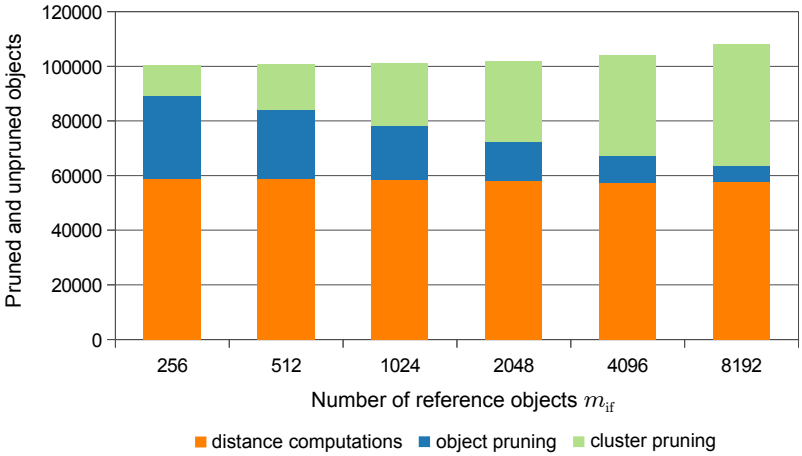


Figure 4.6. — Pruned and unpruned database objects for IF(m_{if},40).

4.2.3. Comparing the Number of Distance Computations of IF4MI with the M-tree and the PM-tree

To compare IF4MI with alternative approaches, we use the M-tree library from *http://mufin.fi.muni.cz/trac/mtree/* (last visit: 9.10.2014). This library already provides some improvements to the original M-tree such as a PM-tree implementation, a multi-way insertion algorithm, and uses an improved split policy adopted from the Slim-tree (for ref-

erences see section 3.2.1 on pages 47–48). The node size of the M-tree is set to 4096 bytes—the default block size of many current file systems—since for an index based on secondary memory, disk I/O can become an additional cost factor besides the necessary distance computations.

In contrast to the M-tree, the PM-tree relies on two parameters m_{pm} and m'_{pm}. Inner node entries of the PM-tree make use of a set of m_{pm} reference objects in order to be able to trim the covering region of a subtree through m_{pm} metric shells (i.e. hyper-rings). Hereby, a hyper-ring is described by a reference object c_i and a pair of minimum and maximum distances from the database objects in the subtree to the reference object c_i. Therefore, $\mathcal{O}(m_{\text{pm}})$ additional space has to be administered per node entry in the inner nodes of a PM-tree. The second parameter m'_{pm} affects the representation of the leaf nodes. A set of m'_{pm} reference objects is used to apply pivot filtering. Hence, m'_{pm} additional $dist(c_i, o)$ distance values are stored per database object in a leaf node entry.

Figure 4.7 visualizes the fraction of necessary distance computations compared to a sequential scan for different M-tree and PM-tree variants (assuming a block size of 4096 bytes) as well as for IF4MI. Different approaches use the same set of m' reference objects for pivot filtering to make results more comparable. The PM-tree, denoted as PM(m_{pm},m'_{pm}), corresponds to an M-tree if both $m'_{\text{pm}} = m' = 0$ and $m_{\text{pm}} = 0$. PM′(0,0) refers to the M-tree using the multi-way insertion algorithm presented in Skopal et al. [2003].

We can observe the benefits of applying pivots since from figure 4.7 the differences in the number of distance computations between the M-tree approaches which do not apply pivots, PM(0,0) and PM′(0,0), and the remaining approaches that do so are clearly noticeable in case of for example $m' = m'_{\text{if}} = m'_{\text{pm}} = 128$. Furthermore, under the current parameter setting, IF(1024,m'_{if}) with $m'_{\text{if}} = 40$ or $m'_{\text{if}} = 128$ is able to outperform the corresponding M-tree and PM-tree approaches when considering the necessary distance computations. Note that it was not feasible to include results for PM(1024,m'_{pm}) in figure 4.7 because of the dramatically increased memory requirements.

Figure 4.8 visualizes the fraction of distance computations in relation to a sequential scan for different tree-based approaches when varying

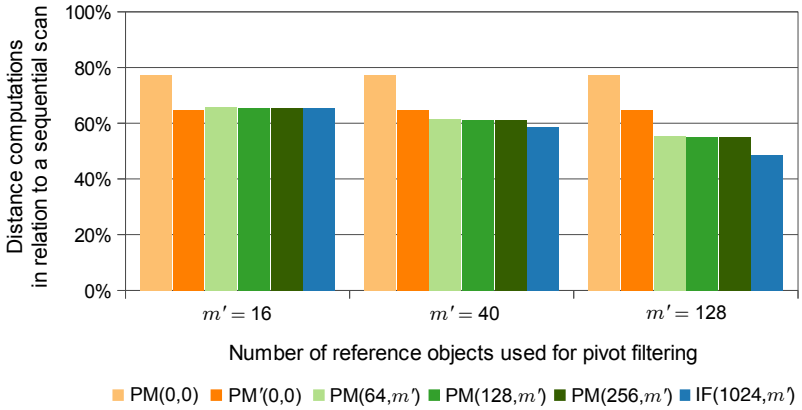


Figure 4.7. — Distance computations compared to a sequential scan for PM(m_{pm}, m') with a block size of 4096 bytes and for IF(m_{if}, m').

the node/block size $\beta_{\text{b}}^{\text{M}}$ of the trees. In case of PM(128,128)[40] and a block size of 2048 bytes, there are many nodes with only a single entry since $128 \cdot 8$ bytes, that is, half of the block size, is already occupied for storing the hyper-rings. In figure 4.8, according to the number of necessary distance computations, block sizes larger than 2048 bytes are more promising. It can be observed that when solely trying to reduce the total number of distance computations, an intermediate block size performs best among the measured block sizes for the M-tree approaches (16 384 bytes for PM(0,0) and 8192 bytes for PM′(0,0)). Thus, from this perspective, there seems to be an adequate clustering, that is, assignment to subtrees, where feature objects are grouped under a common node entry which can successfully be pruned from search. With larger block sizes, a higher branching factor might—especially in case of PM′(0,0)—lead to a less adequate clustering and thus an increase in the number of necessary distance computations (see 32 768 or 65 536 for PM′(0,0) in figure 4.8). When applying a PM-tree (i.e. PM(128,128)), the number of distance computations does not increase again in case

[40] We include measurements for PM(128,128) since measurements for PM(256,128) and a block size of 2048 bytes failed because of increased memory requirements. Nevertheless, for block sizes of 4096 bytes and more, these measurements as well as those of the other PM-tree settings displayed in figure 4.7 show a similar trend with less distance computations the higher the block size.

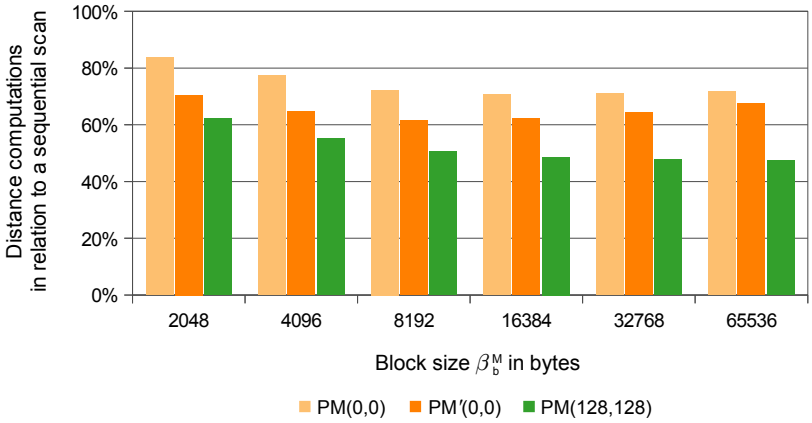


Figure 4.8. — Influence of different block sizes β_b^M on the number of distance computations in relation to a sequential scan for the tree-based approaches.

of larger block sizes. Pivot filtering applied at the leaf level is capable of still pruning many feature objects from search. A similar effect is observed in figure 4.3 on page 83 for IF4MI (e.g. IF([1024],1024)). Also the hyper-rings maintained in the inner nodes support a more effective pruning of subtrees.

It might be argued that especially in case of an indexing scenario in main memory, a PM-tree approach with a big enough block size and an adequate number of feature objects used for pivot filtering (e.g. $m'_{\text{pm}} = 128$) can be used instead of IF4MI as well, since the performance of IF4MI according to the number of distance computations can be achieved (see figure 4.5). In this regard, it is however important to notice that also an approach purely based on pivot filtering such as for example LAESA (see section 3.2.1 on pages 46–47) might fit the needs when the only goal is to reduce the number of distance computations and other costs such as disk I/O are ignored. Here, the memory requirements of the different approaches become important. They are analyzed in the following section 4.2.4.

4.2.4. Memory Requirements of the Approaches

The space complexity of IF4MI consists of several parts. $|O| \cdot m'_{\text{if}}$ precomputed distance values are stored in the posting lists of the inverted

	IF(1024,16)	IF(1024,40)	IF(1024,128)
memory requirements	65.1 MB	74.8 MB	110.0 MB

Table 4.2. — Measured memory requirements of IF4MI for a database with $|O| = 100\,000$ database objects.

file since only m'_{if} instead of m_{if} distance values are stored per indexed database object. Besides this, $\mathcal{O}(m_{\text{if}}^2)$ additional space is used for storing R^{in} and R^{out}. Furthermore, a negligible small amount of directory information is maintained. Finally, the database objects are stored in a separate field of the index (e.g. $\beta_{\text{o}} = 0.5$ kB per database object in case of FC_4). The total memory requirements β_{IF} of IF4MI can thus be determined by formula 4.1 where β_{d} represents the storage space needed for a single distance value, in our case 4 bytes when no binning technique is applied. This leads for example to approximately 110 MB in case of IF(1024,128) which is also measured as shown in table 4.2.

$$\beta_{\text{IF}} = \beta_{\text{d}} \cdot (m'_{\text{if}} \cdot |O| + 2 \cdot m_{\text{if}}^2) \; + \; \beta_{\text{o}} \cdot |O| \tag{4.1}$$

The memory requirements of IF4MI can be estimated in advance with the use of formula 4.1 or formula 4.2 on page 94 when adapted to the space partitioning of the M-Index (see section 4.2.5). Thus, depending on the runtime complexity of the distance computation and the available main memory, adequate numbers of applied reference objects, that is, parameter values for m_{if} and m'_{if}, can be determined. It should also be noted that with an unoptimized implementation, the query response time for the hardest feature combination FC_4 is on average 0.5 seconds in case of IF(1024,1024) (single CPU on Intel i7 860, 8 MB cache, 2.8 GHz).

When estimating the memory requirements of the M-tree, the statistics shown in table 4.3 are used. The size of an inner node entry is considered to be 512 bytes and a leaf node entry is estimated to occupy 508 bytes. If a block size of 4096 bytes is assumed, eight entries fit in inner nodes as well as in leaf nodes. Of course, not all of the nodes are fully occupied. In the following calculations, an average load factor of $ln\,2 \approx 69.3\%$ is assumed (see Yao [1978]). This results—on average—in 5.54 and 5.59 node entries for inner and leaf nodes respec-

routing object	500 bytes
reference to subtree	4 bytes
radius	4 bytes
distance to parent	4 bytes
sum	512 bytes

4.3.1 — Inner node entry.

routing object	500 bytes
distance to parent	4 bytes
object identifier	4 bytes
sum	508 bytes

4.3.2 — Leaf node entry.

Table 4.3. — Memory requirements for M-tree node entries.

tively[41]. There are $|O| = 100\,000$ database objects to be indexed. Using these numbers, we can roughly estimate $\frac{100\,000}{5.59}$ nodes on the leaf level. One level above, there are $\frac{100\,000}{5.59 \cdot 5.54}$ nodes. Two levels above, there are $\frac{100\,000}{5.59 \cdot 5.54 \cdot 5.54}$ nodes, and so on. This leads to an estimated M-tree index size of $\beta_{\text{M-tree}} \approx 89.4$ MB. So, more memory is used in comparison to IF(1024,40) (see table 4.2) at the same time leading to more distance computations (see figure 4.7 on page 89).

Of course, the PM-tree approaches need more memory than the M-tree for representing the hyper-rings as distance pairs in the inner nodes and storing object-pivot distances in the leaf nodes. Memory requirements considerably increase.

In large-scale scenarios with potentially millions of database objects where it is inevitable to store parts of the index on secondary memory, there might be the need for adapting the node size of the tree structures to the physical block size of the underlying file system to reduce disk I/O costs. As shown before, IF4MI can outperform the PM-tree with a typical block size of 4096 bytes according to the number of necessary distance computations. Furthermore, the required storage space of the PM-tree due to the use of the reference objects might become a serious problem. When applying MAMs, no assumption is made about the representation of the database objects. Thus, their memory requirements can become arbitrary complex. Of course, references can be used instead of the objects themselves. However, this requires additional disk accesses when no caching mechanisms are applied.

[41] We believe that this is an optimistic estimate which leads to an underestimation of the true M-tree index size. As an indicator, the M-tree library offers statistics which show that the average number of leaf node entries is 4.94 for PM(0,0) and 3.24 for PM′(0,0).

	distance computations	memory requirements
$\mathrm{MI}_{\mathrm{mod}}$(40,3)	58 828.3	66.50 MB
IF(347,40)	58 822.0	66.49 MB

Table 4.4. — Comparing the space partitioning of IF4MI and the M-Index on a database with 100 000 objects in case of 20-NN queries.

4.2.5. Using the Space Partitioning of the M-Index

Table 4.4 compares two different space partitioning approaches applied to IF4MI. $\mathrm{MI}_{\mathrm{mod}}(m_{\mathrm{mi}}, l)$ denotes the variant where the space partitioning of the M-Index is applied. In contrast to the original M-Index, $\mathrm{MI}_{\mathrm{mod}}(m_{\mathrm{mi}}, l)$ is built upon the inverted file library and does not rely on the mechanism for mapping database objects to one-dimensional values and storing them in a B^+-tree [Comer, 1979]. $\mathrm{MI}_{\mathrm{mod}}(m_{\mathrm{mi}}, l)$ can on level l theoretically maintain up to m_{mi}^{l} clusters. Every cluster is mapped to a single posting list to which an object reference is added if a database object lies in the particular cluster. Within a posting, the $dist(c_i, o)$ values used for pivot filtering are stored as before.

The parameters of $\mathrm{MI}_{\mathrm{mod}}$ are set to $m_{\mathrm{mi}} = 40$ and $l = 3$. Experiments in Novák et al. [2011, *sect.* 4.3.1] based on the CoPhIR dataset with 100 000 database objects show that a static structure with a constant value of $l = 3$ already achieves good efficiency measured by the number of distance computations. Even a dynamic tree structure evolving by limiting the number of feature objects per cluster cell cannot outperform the static structure in case of 50-NN queries when the number of reference objects m_{mi} is bigger than 20. In order to compare $\mathrm{MI}_{\mathrm{mod}}$(40,3) with IF4MI, both approaches use the same set of pivots for pivot filtering and thus $m'_{\mathrm{if}} = m_{\mathrm{mi}} = 40$. The value of m_{if} is set to 347—which is the maximum possible value so that the measured memory size of IF(m_{if},40) remains below the corresponding memory requirements of $\mathrm{MI}_{\mathrm{mod}}$(40,3). Results according to the number of necessary distance computations are similar (see table 4.4) which was expected since the same set of reference objects is used for pivot filtering and the same pruning rules are applied. The efficiency of the static M-Index in terms of the number of distance computations is thus possible for IF4MI which can be extended by the Voronoi-like space partitioning of the M-Index.

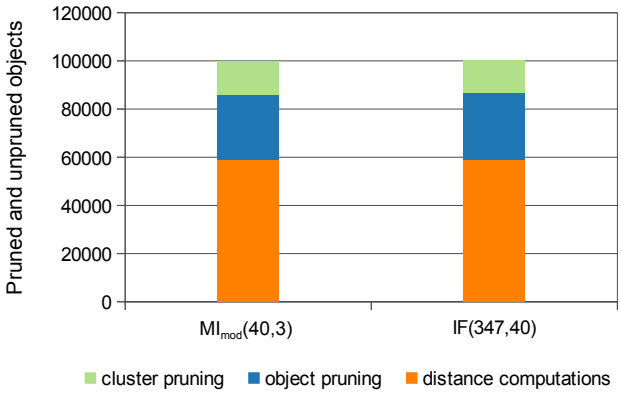


Figure 4.9. — Pruned and unpruned database objects for IF(347,40) and MI_{mod}(40,3) on the database with 100 000 objects when performing 20-NN queries.

To show that the similar performance of IF(347,40) and MI_{mod}(40,3) according to the number of necessary distance computations is not completely due to the pruning power of pivot filtering, figure 4.9 visualizes the number of database objects which are pruned by cluster pruning. It can be observed that MI_{mod}(40,3) offers a slightly better pruning power according to cluster pruning. However, IF(347,40) can compensate this by excluding more database objects through pivot filtering. The two numbers of necessary distance computations in figure 4.9 correspond to the numbers displayed in table 4.4. The height of the MI_{mod}(40,3) bar in figure 4.9 is $347 - 40 = 307$ units smaller than the height of the IF(347,40) bar, since for every query IF(347,40) requires 347 distance computations to compute the list L_q (see line 1 of the algorithm in figure 4.2 on page 79) whereas MI_{mod}(40,3) only needs 40 distance computations to determine the pivot permutation for the query.

The memory requirements of $\text{MI}_{\text{mod}}(m_{\text{mi}}, l)$ can be determined by slightly modifying formula 4.1 on page 91. Since $m'_{\text{if}} = m_{\text{mi}}$ and since the number of clusters resulting from the l-permutations is at most $m_{\text{mi}}^{\underline{l}}$, the memory requirements of $\text{MI}_{\text{mod}}(m_{\text{mi}}, l)$ can be estimated by:

$$\beta_{\text{MI}_{\text{mod}}} = \beta_{\text{d}} \cdot (m_{\text{mi}} \cdot |O| + 2 \cdot m_{\text{mi}}^{\underline{l}}) \; + \; \beta_{\text{o}} \cdot |O| \qquad (4.2)$$

4.2.6. Improvements to Pivot Filtering

Many MAMs rely on pivot filtering such as the M-Index, the PM-tree, and IF4MI, to name only a few. The number of database objects which can be pruned through pivot filtering depends on how many and which pivots are selected (see section 2.6). Furthermore, the order in which the selected pivots are processed becomes crucial for speeding-up the pivot filtering. Thus, in this section, it is analyzed how to best evaluate the pivot filtering condition outlined in formula 2.19 on page 35. This is of special importance to IF4MI since the number of reference objects used for pivot filtering can be large. In the following, four different strategies are analyzed for determining the order in which the centers $c_i \in C$ are processed so that $|dist(q, c_i) - dist(c_i, o)| > r$ is fulfilled as early as possible (see formula 2.19).

Random: In this case, no specialized strategy for determining the order of how to evaluate different centers $c_i \in C$ is applied. Thus, the centers c_i are processed from $i = 1, 2, \ldots, m'_{\text{if}}$ which corresponds to a random ordering since centers are initially chosen at random. This random ordering represents the baseline against which the three alternative approaches presented in the following are compared.

***L_q* reverse** and ***L_q* order**: During pivot filtering, lower bound distances $\check{d}_i = |dist(q, c_i) - dist(c_i, o)|$ of the true distance $dist(q, o)$ are computed with the help of reference objects c_i to determine if the possibly expensive computation of $dist(q, o)$ can be avoided. The consideration of the cluster centers c_i in the formula 2.19 can be stopped as soon as $\check{d}_i > r$ is fulfilled for the first time with r denoting the current search radius. Thus, it is desirable to first check centers c_i with large lower bound distances $\check{d}_i$. An analysis of the formula $\check{d}_i = |dist(q, c_i) - dist(c_i, o)|$ shows that the resulting absolute value can be high either because $dist(q, c_i)$ is high and $dist(c_i, o)$ is low or because $dist(q, c_i)$ is low and $dist(c_i, o)$ is high. As a consequence, especially medium values for $dist(q, c_i)$ are candidates with a limited potential for selective lower bounds. Since the list L_q is computed anyway at the beginning of the query process, L_q can directly be applied exploiting this observation. L_q *reverse* corresponds to an ordering of the centers c_i by decreasing distance $dist(q, c_i)$—in opposition to the increasing order denoted as L_q *order* typical for many scenarios (see e.g. Eisenhardt et al. [2006] or the algorithm in figure 4.2). Results in figure 4.10 indicate that both strategies cannot outperform the processing of the

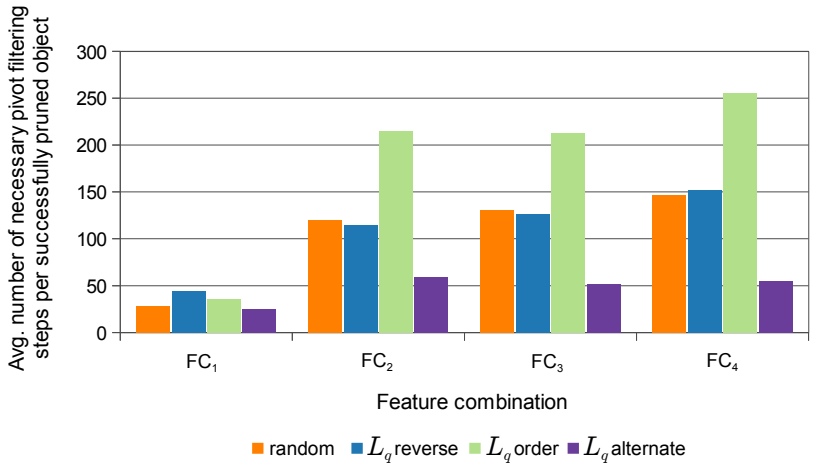


Figure 4.10. — Necessary pivot filtering steps per successfully pruned database object for different pivot filtering heuristics in case of IF(1024,1024).

reference objects in random order. However, a hybrid combination is investigated, too.

***L_q* alternate:** This approach follows the observation from Celik [2006, *ch.* 4] that good pivots lie either close to or far from the query object (see section 2.6 and section 3.2.1 on pages 46–47). Thus, the two approaches L_q *reverse* and L_q *order* are combined and the centers are chosen from L_q (see definition 4 on page 28) in an alternate way. First, the center at position m'_{if} of the list L_q is selected, then it is proceeded with the center at position 1, the center at position $m'_{\text{if}} - 1$, the center at position 2, and so on. Consequences for query processing can be observed from figure 4.10. This approach clearly outperforms the other techniques. In case of FC_4, if a database object can be successfully pruned from query processing by applying pivot filtering, approximately 55 lower bound distances are computed on average compared to 146 without optimization. Since on average 42 198.1 database objects are pruned for this particular setting (see object pruning for FC_4 and IF(1024,1024) in figure 4.3 on page 83), the number of lower bound distance computations can on average be reduced per query from approximately 6.2 million to 2.3 million.

Instead of using L_q and thus $dist(q, c_i)$, the decision of choosing reference objects for pivot filtering can also be based on L_o, that is, an ordering of all $c_i \in C$ according to $dist(o, c_i)$. L_o is computed during the insertion of a database object $o \in O$, but it is not explicitly present in the current index structure at query time. Cluster indices i could be stored in addition to now sorted $dist(o, c_i)$ values (either in L_o order or reversely) to be able to compute the lower bound distances without having to recompute L_o at query time. Alternatively, with the initial design of an individual posting as displayed in figure 4.1 on page 78, the list of $dist(o, c_i)$ values could be sorted at query time in order to identify the largest value, the smallest value, the second largest value, the second smallest value, etc. While the first alternative would increase the size of the index, the second would negatively influence runtime performance. Furthermore, when testing a random run with 200 queries, the alternate strategy based on L_o could not reduce the number of distance computations compared to L_q *alternate*.

4.2.7. Processing Filter Queries

Studies in CBIR show that query processing purely based on content-based search techniques is not always sufficient for effective retrieval. Textual information provides an important additional search criterion which is frequently applied [Paramita et al., 2009; Popescu et al., 2010; Tsikrika et al., 2011]. Not only in the domain of CBIR, it is necessary to integrate various search criteria and filter searches. The same applies for web search where for example file type, language, or date are recognized as important filter criteria.

In this section, we show exemplarily how IF4MI can benefit from the extensive knowledge in the field of inverted files (see e.g. Büttcher et al. [2010] and Zobel and Moffat [2006]). Skip pointers are used in order to make query processing more efficient since they prevent the unnecessary reading of posting list entries. Textual filter conditions are applied in combination with content-based k-NN queries. Therefore, also the tags of the CoPhIR collection are indexed. Of course, instead of applying tags as filters, other criteria are also possible.

We analyze a random run with 200 queries. The queries are constructed as follows. We use randomly chosen images as described in section 4.2.1 and define the tag with the highest document frequency among all tags associated with the image as our filtering query term.

	$m_{\text{if}} = 1024$	$m_{\text{if}} = 512$	$m_{\text{if}} = 256$
average	9766	24 855	35 670
minimum	2494	5017	9779
maximum	19 906	67 655	72 680
median	7631	20 046	32 775
25th percentile	6212	13 514	23 623
75th percentile	15 401	28 878	43 896
#queries	26	70	88

Table 4.5. — Skipped postings for IF(m_{if},*) in case of textual filter queries.

Among the 200 randomly chosen query images, only 138 are tagged at all, leaving 138 test queries.

IF4MI-based query processing considers the inverted file indexing image content properties only if the document frequency of the filter term is higher than the number of clusters m_{if}. Otherwise, the number of distance computations $dist(q, c_i)$ in order to compute the list L_q (see line 1 of the algorithm outlined in figure 4.2 on page 79) already exceeds the number of necessary distance computations when calculating the object-query distances $dist(q, o)$ directly for all objects fulfilling the textual filter criterion. As shown in the last line of table 4.5, only for 26 query images the document frequency of the most frequent tag is higher than 1024, only for 70 query images the document frequency of the most frequent tag is higher than 512, and only for 88 query images the document frequency of the most frequent tag is higher than 256[42].

Table 4.5 additionally shows for FC_4 how many postings are skipped when performing 20-NN queries and additionally applying the textual query filter. The default parameter values of Lucene in version 3.0.2 such as a skip interval of 16 are directly applied without optimization. We can see that with decreasing m_{if} and thus fewer clusters and on average longer posting lists, the number of skipped postings increases. This is especially beneficial in case of a main memory index. Here, not

[42] To benefit from skipping—under the assumption that distance computations are the dominant cost factor—multiple indexes with different values of m_{if} such as 1024, 512, 256, etc. might be maintained simultaneously.

disk access but the reading and decoding of posting lists becomes the dominating cost factor [Boldi and Vigna, 2005, *p.* 26]. Note that in case of $m_{\text{if}} = 512$, an average value of approximately 25 000 skips does not mean that approximately 75 000 database objects are accessed since cluster pruning is applied and posting lists are excluded from query processing whenever possible.

Chapter 5.

RS4MI—Resource Description and Selection for Metric Indexing and Search

This chapter introduces RS4MI—a resource description and selection scheme for similarity search in general metric spaces. The chapter consists of three major sections. Section 5.1 focuses on approximate search techniques (see Blank et al. [2007], Blank and Henrich [2009], and Blank and Henrich [2010a,b, 2012a]) which extend the approach from Eisenhardt et al. [2006] outlined in section 3.3.4 on pages 72–73. Afterwards, algorithms for precise search are discussed. Section 5.2 addresses range query processing (see Blank and Henrich [2013b]) whereas section 5.3 discusses the processing of k-NN queries.

5.1. RS4MI for Approximate Search

The approach from Eisenhardt et al. [2006] briefly outlined in section 3.3.4 on pages 72–73 is an approximate search technique. Leaving the resource selection algorithm unchanged, there are several possibilities for improving the resource descriptions. This issue is addressed in the following section 5.1.1 where the Highly Fine-grained Summaries (HFS) and the Ultra Fine-grained Summaries (UFS) are presented. As an excursus, section 5.1.2 shows that these resource descriptions can also be successfully applied in the domain of geographic IR where usually concepts related to SAMs are used. Section 5.1.2 compares the performance of HFS and UFS with basic alternative techniques derived from centralized SAMs.

5.1.1. Extending Cluster Histograms to HFS and UFS

We extend the work from Eisenhardt et al. [2006] outlined in section 3.3.4 on pages 72–73 in several directions. In case of HFS_m with m indicating the number of reference objects being used, we increase the number of reference objects for computing the cluster histograms for example from $m = 256$ to $m = 8192$ or even more (see Blank et al. [2007]). By doing so, the feature space is partitioned in a more fine-grained way offering improved resource selection performance. Since a higher number of reference objects leads to less space efficient summaries, we apply compression techniques. Thus, we can achieve better resource selection performance with more space efficient resource descriptions compared to the approach from Eisenhardt et al. [2006]. Reference objects are selected from an external source and transferred to the peers together with updates of the P2P software. This leads to a decrease in overall network load and makes distributed selection mechanisms of the reference objects obsolete. Resource selection performance is only slightly affected by this change as it is shown on pages 104–105. Binary histograms (UFS_m) can outperform integer histograms (HFS_m) [Blank and Henrich, 2010a]. In contrast to HFS, UFS are based on a bit vector with the bit at position i ($1 \leq i \leq m$) indicating if center c_i is the closest center to one or more of a peer's database objects. Hence, we obtain a bit vector of size m. Of course, there is some loss of information when switching from HFS to UFS with m staying constant. However, UFS have the potential of resulting in more space efficient resource descriptions. This allows for more reference objects being used which can result in similar or even improved resource selection performance compared to HFS when utilizing the same amount of memory.

Experimental setup

We use a collection of 233 827 images crawled from Flickr during the years 2005 and 2006. The images are assigned to peers based on the Flickr user ID to reflect a realistic scenario. Hence, we assume that every Flickr user operates a peer of its own. The images are mapped to 10 601 users/peers which are used in our simulation. Figure 5.1 shows the distribution of peer sizes, that is, the number of images which are maintained per peer. The general characteristic of the distribution is typical for P2P file sharing applications with few peers managing large

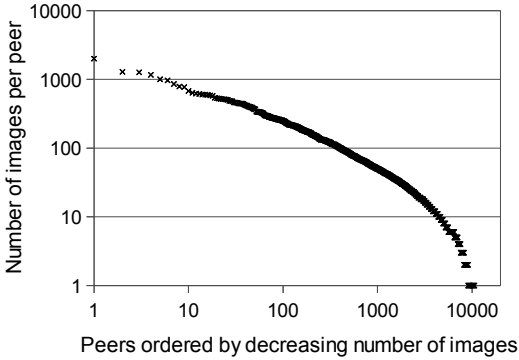


Figure 5.1. — Distribution of peer sizes, that is, the number of database objects maintained per peer.

amounts of the images and many peers administering only few images [Saroiu et al., 2002].

We analyze a scenario where every peer knows the resource description of every other peer. Of course, such an approach does not scale. However, this scenario is for example typical in a PlanetP subnet of a scalable Rumorama-based P2P IR system. Rumorama [Müller et al., 2005b] can cope with multiple subnets and thus scale to much higher workloads than the ones analyzed here.

In the experiments in this section 5.1.1, every image is described by a 166-dimensional uniformly quantized color histogram based on the HSV color space with 18 hues, three saturations, and three values, plus four levels of gray (for an outline of the feature which is e.g. also used in Eisenhardt et al. [2006], see Smith [1997, *p.* 16f.]). Image feature objects are compared using the Euclidean distance. We analyze 20 runs where we change the reference objects used. During a run, we perform 100 queries and we randomly select a query image from the underlying collection. The set of queries stays constant over all runs. We analyze the number of queried peers for retrieving the 20 closest feature objects to a given query feature object.

Reference objects for summary creation and peer ranking are chosen from the underlying collection (denoted as UFS/HFS in the following figures) or from a second collection of 45 931 Flickr images (denoted in the following as UFSe/HFSe, with "e" indicating the use of an external collection for obtaining the reference objects). It is important to note

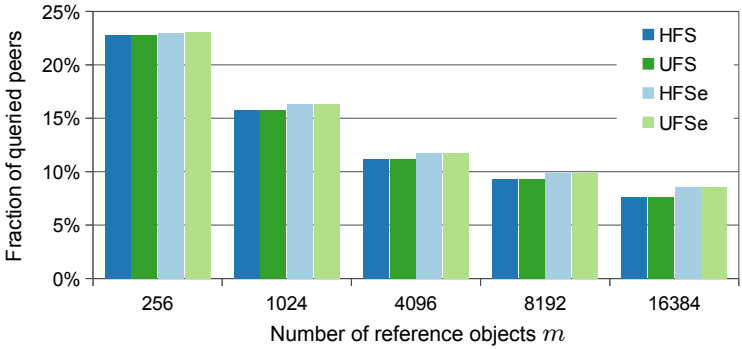


Figure 5.2. — Fraction of queried peers to retrieve the top-20 image feature objects, i.e. the 20-NNs.

that both collections are disjoint according to the unique Flickr image IDs and user IDs, but there is some minor natural overlap among the collections; 24 of the 233 827 images also appear in the external collection because some images are independently uploaded by multiple users on Flickr. Selecting the reference objects from the external collection reflects a scenario where the reference objects are transferred to the resources upfront, for example with updates of the P2P software, to reduce network load in the operation phase. All resources use the same set of pivots and thus rely on the same global Voronoi-like space partitioning.

Analysis of resource selection performance

Figure 5.2 shows the fraction of peers which are queried on average to retrieve the 20 closest feature objects (i.e. the top-20 or 20-NNs) according to a given query object. Resource selection performance increases degressively with increasing m. When comparing HFSe to HFS and UFSe to UFS within each group of bars, that is, for a fix value of m, there is a small gap in resource selection performance. This increase in the fraction of visited peers when choosing the reference objects from the external collection is especially noticeable in case of $m = 16384$ (i.e. the rightmost bar group in figure 5.2). For UFS/HFS, the probability of choosing a reference object which is used also as a query object increases with increasing m. Such situations can lead to improved re-

m	256	1024	4096	8192	16384
HFSe	82.1 B	110.3 B	141.3 B	162.5 B	195.2 B
UFSe	62.9 B	74.5 B	87.3 B	96.2 B	107.5 B

Table 5.1. — Average summary sizes in bytes (zipped).

source selection performance since queries are randomly chosen from the underlying data collection. Also the general characteristics of the two data collections may differ, reflected by the distribution of the feature objects within the feature space.

Figure 5.2 additionally shows slightly improved resource selection performance for HFS(e)_{256} compared to UFS(e)_{256} respectively[43], which is due to the use of non-binary histogram information during peer ranking in case of HFS(e). In general, this slight gap between HFS(e) and UFS(e) more and more diminishes with an increasing number of centers because HFS(e) histograms more and more pass into binary histograms. Already for HFS_{1024} compared to UFS_{1024} and HFSe_{1024} compared to UFSe_{1024}, there is no noticeable difference in resource selection performance.

Analysis of summary sizes

The size of the resource descriptions after zipping is analyzed in table 5.1 and figure 5.3. Table 5.1 shows average summary sizes $\beta_{\text{rd}}^{\text{avg}}$ when using UFSe instead of HFSe. As expected, summaries in case of UFSe are on average clearly more space efficient than in case of HFSe[44].

Figure 5.3 visualizes the different quartiles and minimum/maximum values of the summary sizes in a box plot. It shows that the median in case of HFSe is higher than for UFSe. Interquartile ranges of HFSe and UFSe become more and more similar when increasing m. However, summaries in case of UFSe are clearly more space efficient than in

[43] HFS(e)_m is used as an abbreviation for "HFS_m and HFSe_m". The same notation is also adopted for UFS throughout this thesis. In a similar way, HFS/UFS abbreviates "HFS and UFS".

[44] The numbers for HFS and UFS when no external collection of pivots is used are not displayed here because they show similar characteristics and provide no additional insights.

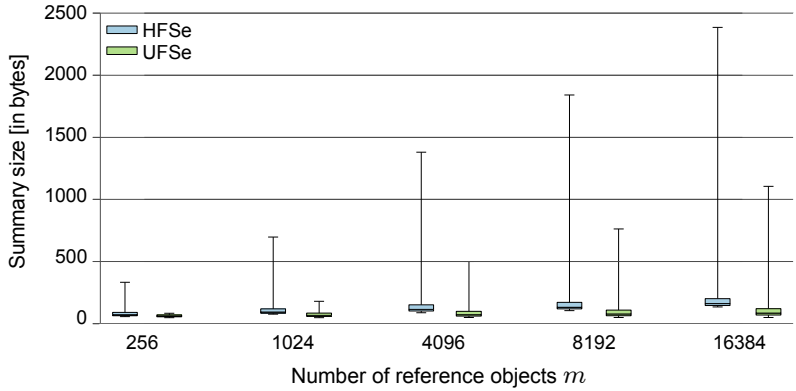


Figure 5.3. — Box plot of summary sizes (zipped).

case of HFSe. Also the overall range of HFSe summary sizes is bigger than the range of summary sizes in case of UFSe. The distributions of summary sizes are positively skewed indicating many peers with small summaries and few peers with big summaries. Thus, the distribution of peer sizes visualized in figure 5.1 is roughly reflected in the distribution of summary sizes shown in figure 5.3.

A mixed peer ranking scheme using for example HFSe for the peers with few documents and UFSe for the peers with many documents can also be an alternative. The cost of one round of rumor spreading where every peer sends its resource description to all other peers in the system can roughly be estimated by $\beta_{\text{rd}}^{\text{avg}} \cdot n \cdot (n-1)$ with n being the number of peers in a PlanetP-like setting. Hence, the estimated cost is proportional to $\beta_{\text{rd}}^{\text{avg}}$. This is the reason why we mostly focus on the analysis of average summary sizes within this thesis. Since a decrease in resource selection performance can be perceived only for small values of m when switching from HFS(e) to UFS(e) respectively (see figure 5.2), the UFS(e) alternative with a smaller $\beta_{\text{rd}}^{\text{avg}}$ compared to HFS(e) can safely be chosen for all peers in the network in case of big values of m.

In a more comprehensive cost analysis, the cost for query processing has to be additionally taken into account such as for example in Blank and Henrich [2010b] where we analyze if sending summary data and performing resource selection trades-off a naïve approach where the peers directly transfer full index data instead of summarizing low-dimensional latitude and longitude coordinates in case of geographic IR.

Analysis of time complexity

In general, it is important that peer ranking consumes only a reasonable amount of time. We use the peer ranking scheme proposed by Eisenhardt et al. [2006] as described in section 3.3.4 on pages 72–73. When applying this scheme, ranking peers mainly means sorting m-dimensional vectors, that is, the HFS(e)$_m$/UFS(e)$_m$ histograms, where the importance of the single dimensions is defined by the list L_q which contains the IDs i of the reference objects $c_i \in C$ sorted by the $dist(q, c_i)$ distances in ascending order. The earlier a cluster ID i appears when scanning L_q from the beginning to the end, the more important the very vector dimension is. In a first run, the peers are sorted according to the dimension representing the closest reference object. Of course, this sorting can be done in $\mathcal{O}(n \log n)$ where n stands for the number of peers in the considered PlanetP network. In a worst case scenario, all peers would be identical in the number of database objects maintained in each of the m clusters ending up in a complexity of $\mathcal{O}(m\, n \log n)$. Thus, the worst case complexity for the peer ranking depends on m which is disadvantageous for HFS(e)/UFS(e) with high values of m.

To test whether this worst case scenario has practical implications, we compare the original approach considering—if necessary—all clusters until the end with a modified variant using at most the 256 clusters with its centers closest to the query. If no decision is possible after comparing the histogram values for these 256 clusters, a random choice is made.

In the following, UFS(e)$_m$(256) denotes the modified approach considering m centers for summary creation and at most the 256 closest centers to the query object for peer ranking. When increasing m to a rather large number such as $m = 16384$, the feature space is partitioned in a fine-grained way. If only 256 centers are used for the peer ranking, the fraction of unused clusters which potentially contain useful information increases; for example in case of UFS(e)$_{16384}$(256), at least $1 - \frac{256}{16384} = 98.4\%$ of the summary bins are ignored during peer ranking.

Figure 5.4 indicates that there is no noticeable difference in resource selection performance between UFS(e)$_m$ and UFS(e)$_m$(256) for summaries with up to $m = 8192$ centers. Only UFS(e)$_{16384}$ performs slightly better than UFS(e)$_{16384}$(256). The results in figure 5.4 demonstrate that only few of the m histogram bins are usually considered

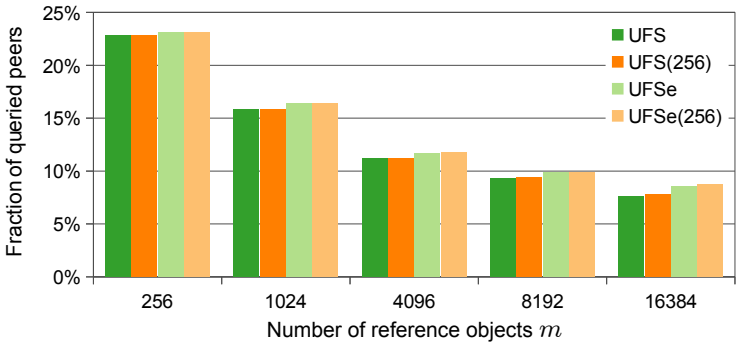


Figure 5.4. — Fraction of queried peers for retrieving the top-20 images with UFS(e)(256) vs. UFS(e).

for peer ranking. Thus, the worst case time complexity for the peer ranking of in this case $\mathcal{O}(256\, n \log n)$ is no longer crucial.

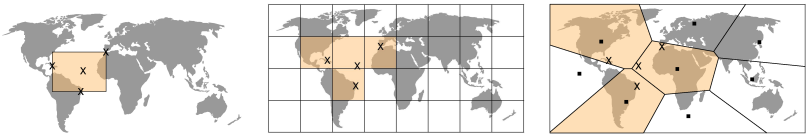
5.1.2. Excursus: HFS and UFS for Geographic IR

In this excursus, we compare two basic resource description and selection techniques for geographic IR—a technique based on bounding boxes and a grid approach—with our HFS and UFS. Resource selection performance of those approaches is briefly discussed in Blank and Henrich [2009]. Approximate k-NN query processing is analyzed in Blank and Henrich [2010b, 2012a]. The results are summarized in this section.

Bounding Box (BB) summaries

With the BB approach, every peer computes a bounding box around the geographic coordinates of its image collection (see figure 5.5.1). A latitude/longitude-pair (for short: lat/lon-pair) is encoded with eight bytes, four for latitude and four for longitude. Therefore, we require $8 \cdot 2 = 16$ bytes of raw data for the bounding box (i.e. two lat/lon-pairs, e.g. the lower left and upper right corner).

Peer ranking is performed as follows. If a peer p_a contains the query location within its bounding box whereas peer p_b does not, peer p_a is ranked higher than peer p_b, and vice versa. When the query location lies within the bounding box of both peers p_a and p_b, the size of a peer



5.5.1 — The BB approach.

5.5.2 — The grid approach with GRID_4.

5.5.3 — $\text{HFS}_8/\text{UFS}_8$, ■ indicates reference points.

Figure 5.5. — Visualizing summary creation for geographic IR. Four images are geotagged in this example, indicated as **x**.

(i.e. the number of images a peer administers) is used as an additional criterion. Peers with more images are ranked higher than peers with fewer images. If neither the resource description of peer p_a nor the resource description of peer p_b overlaps the query location, the peer with the smaller minimum distance from the query location to its bounding box is preferred.

In general, we assume a spherical model of the earth with a radius of 6371 kilometers. If not stated otherwise, a peer uses the Haversine distance [Sinnott, 1984] to compute the distance between two points on the sphere.

Grid-based summaries (GRID_τ)

In a second approach, the geographic coordinate space is represented as a grid (see figure 5.5.2). A parameter τ defines the number of grid rows. The number of columns is twice the number of rows since the longitude range is twice as big as the latitude range. The range of a grid cell (in degrees) is determined by $\frac{180^\circ}{\tau} = \frac{360^\circ}{2\tau}$ in the latitude and longitude domain. Such a simplified view is for example also applied in Dolin et al. [1997] and results in non-uniform grid cell sizes on the sphere. We gain selectivity and thus resource selection performance by increasing the number of grid cells at the price of additional storage overhead partially compensated through compression techniques. Every grid cell is represented by a single bit. If one or more image locations fall into a certain cell, the corresponding bit is set to 1. Otherwise, it remains 0. Bit positions in the summary are determined horizontally from left to right and from bottom to top.

During peer ranking, the grid cell containing the query location is determined. If peer p_a has an image within this cell whereas peer p_b

has not, peer p_a is ranked higher than peer p_b, and vice versa. We also consider neighboring grid cells. If either both or none of peer p_a and peer p_b have an image located within the cell containing the query location, GRID considers the neighboring cells recursively until a ranking decision can be made. So, in the first round the ranking decision is always based on a single cell; in the second round it is in most cases[45] based on $1 + 8 = 9$ cells and in the third round on $1+8+16 = 25$ cells, and so on. The ranking criterion in every round is the number of grid cells containing one or more image locations—the more the better. When based on the summary information no ranking decision can be made, peers are ranked at random.

Experiments

In the experiments in this section, we analyze BB, GRID, HFS, and UFS (see figure 5.5). We use a collection of publicly available geotagged Flickr images which had been uploaded to Flickr and which we crawled in 2007[46]. In our scenario, again, every Flickr user operates a peer of its own. We therefore assign images to peers by means of the Flickr user ID. The Geoflickr collection consists of 406 450 geotagged image locations from 5951 different users/peers.

Figure 5.6 shows the distribution of the number of images per peer. Again, the distribution is skewed which is typical for P2P networks [Saroiu et al., 2002]. Approximately the first 1% of the biggest peers, that is, the 60 biggest peers, administer 42.0% of the images. The biggest peer maintains 8.8% of the images. In opposition, approximately 20.7% of the peers administer only a single image. Approximately 50% of the images are maintained by 1.8% of the peers.

Figure 5.7 shows the geographic distribution of the image locations. The Geoflickr collection consists of photos taken in various parts of the world with hotspots in North America, Europe, and Japan.

[45] This is not always the case since there might be no neighboring cells in a certain direction, for example as soon as a cell in the north or south is reached. Of course, at the 180-degree meridian we assume that there is no boundary and neighborhood relations are valid in both directions.

[46] A second collection with 246 937 images from the United Kingdom mapped to 2609 peers and obtained from *http://www.geograph.org.uk/* (last visit: 24.9.2014) is used and analyzed in Blank and Henrich [2010b, 2012a] leading to similar results as the collection obtained from Flickr on which we focus here.

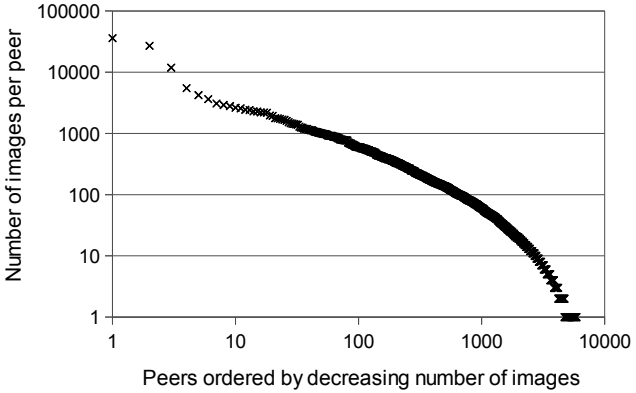


Figure 5.6. — Number of images per peer for the Geoflickr collection.

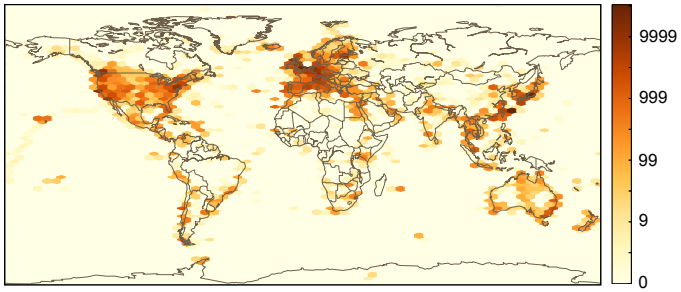


Figure 5.7. — Geographic distribution of image locations for Geoflickr.

In the experiments, we use 200 image locations as queries. These are randomly selected from the underlying data collection. For HFS and UFS where the outcome of the experiments is affected by the selection of reference points, we perform ten runs with the 200 queries each, varying the set of randomly chosen reference objects in each run. Space efficiency of different resource description approaches is analyzed by looking at average summary sizes. For compressing the summaries, we apply the Java[47] default gzip implementation. Our measurements include serialization overhead necessary to distribute the resource descriptions within the network. We randomly choose a geo-location of an image from the entire document collection as the query location.

[47] see *http://www.java.com/*, last visit: 2.9.2014

Since we do not remove the image with the query location, it is—on average—more likely that a big peer contributes to the final search result than a small peer because—on average—it is more likely to choose the query from a big peer than from a small peer[48].

As before, when measuring resource selection performance, we determine the fraction of peers that is contacted to retrieve the global top-k image locations ($k = 20$) according to a given lat/lon-pair as query location. The global top-k geo-locations are computed using the Vincenty distance [Vincenty, 1975]. Assuming an ellipsoidal shape of the earth for determining the ground truth seems reasonable as we are looking for the true NNs on a global scale. In opposition, within our algorithms we assume a spherical shape of the earth for runtime performance reasons[49]. Since we are interested in the quality of the resource selection techniques, we analyze all of a peer's image locations as soon as it is contacted because the top-k image locations of a peer determined using the Haversine distance might differ from the top-k image locations computed using the Vincenty distance. In a real-world application, a peer will transfer only the top-k image locations (together with some additional information such as the peer ID, etc.).

When analyzing resource selection performance in table 5.2 and figure 5.8, it can be observed that the minimum values are the same for all approaches; 0.02% means that only one peer needs to be queried for finding all 20-NNs. In general, both HFS_{8192} and UFS_{8192} perform better than BB and GRID_{64}. HFS_{8192} and UFS_{8192} as well as GRID_{64} result in the same number of summary bins ($64 \cdot 2 \cdot 64 = 8192$). Interquartile ranges of HFS_{8192} and UFS_{8192} are smaller than for GRID_{64} and especially BB. GRID_{64} offers better resource selection performance than BB which is a too simplistic approach and representing a peer by multiple bounding boxes seems necessary. The median and the 75th percentile of GRID_{64} are clearly below corresponding values for BB. Also the interquartile range is smaller. Nevertheless, at least for some queries, it is very difficult for GRID_{64} to offer adequate resource se-

[48] An alternative mode for selecting the query location where we select a random peer and from this peer we choose a geo-location of an image at random is additionally analyzed in Blank and Henrich [2010b]. In this case, it is more likely that also a small peer contributes to the top-k query result since peers are chosen equiprobable.

[49] This seems reasonable and even the use of the Euclidean distance instead of the Haversine distance results in similar resource selection performance [Blank and Henrich, 2012a].

	$\mathbf{HFS_{8192}}$	$\mathbf{UFS_{8192}}$	**BB**	$\mathbf{GRID_{64}}$
average	0.20%	0.34%	1.89%	1.30%
minimum	0.02%	0.02%	0.02%	0.02%
maximum	2.91%	4.23%	9.14%	17.69%
median	0.03%	0.07%	0.91%	0.55%
25th percentile	0.02%	0.03%	0.08%	0.12%
75th percentile	0.15%	0.20%	2.91%	1.68%
avg. summary size	121.8 B	61.1 B	45.0 B	58.4 B

Table 5.2. — Resource selection performance and average summary sizes for the HFS, UFS, BB, and GRID approaches.

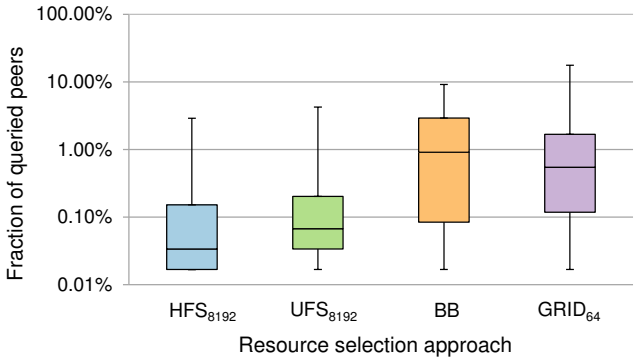


Figure 5.8. — Resource selection performance of HFS, UFS, BB and GRID.

lection performance as for example indicated by the maximum value of almost 18%. In such cases, queries lie in a very populated grid cell, that is, a grid cell where many peers assign documents to. When looking at queries which offer poor resource selection performance for GRID_{64} in more detail, it can be observed that usually these queries lie in cells where many of the documents reside (i.e. cells which contain metropolises such as London for example).

Average summary sizes for the different peer selection schemes are also displayed in table 5.2. Compression is applied to all approaches but

BB since it is beneficial in these cases. The size of a BB summary is always 16 bytes for the bounding box plus 27 bytes serialization overhead, so 43 bytes in total. As it is desirable to use the number of documents of a peer in the peer ranking process, we assume additional two bytes for peer size information which leads to overall summary sizes of 45 bytes for BB. HFS_{8192}—using integer histograms—results in clearly less space efficient resource descriptions than $GRID_{64}$ which also relies on 8192 summary bins but uses binary histograms. $GRID_{64}$ also offers slightly more space efficient resource descriptions than UFS_{8192}. Densely populated data regions are partitioned in a more fine-grained way in case of UFS resulting in more bits being set compared to $GRID_{64}$. However, this also results in better resource selection performance for UFS_{8192} compared to $GRID_{64}$.

So far, we have assumed that the m reference points are chosen from the underlying data collection. Although this approach is feasible in general, we now evaluate different sources for the reference points similar to the HFSe and UFSe approaches analyzed in section 5.1.1 where an external collection of reference images is used.

Here, we employ United Nations per-country statistics from the year 2002 obtained through Worldmapper[50] about mens' income (INC), gross domestic product (GDP), population (POP), and WWW usage (WEB). Based on these statistics, we proportionally select the number of reference points from a certain country using the GeoNames gazetteer[51]. Reference points are selected among all populated places of a certain country at random. So, for example, if 5% of the world's total mens' income is earned in a certain country, 5% of the reference points are randomly chosen among all populated places of the specific country according to the information provided by the GeoNames gazetteer.

When using external sources for the reference points, selecting them according to GDP is the most promising approach followed by WEB and INC, respectively (see figure 5.9). These techniques adapt best to the data collection that is used. In general, figure 5.9 shows that resource selection performance relies on an adequate selection of the reference points according to the expected origin of the query locations as well as the administered image locations. A monitoring can

[50] see *http://www.worldmapper.org/*, last visit: 18.9.2014
[51] see *http://www.geonames.org/*, last visit: 14.10.2014

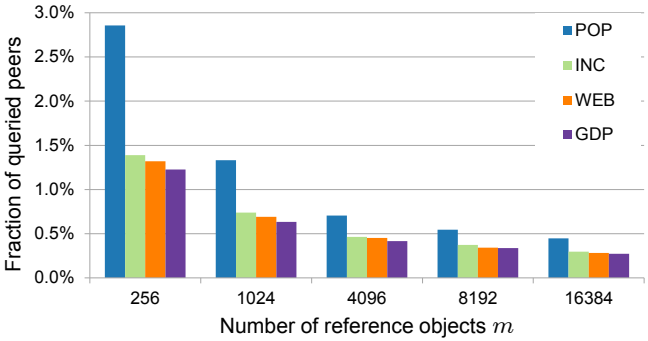


Figure 5.9. — Selected strategies for choosing the reference points for UFSe when analyzing 20 runs with the 200 queries each.

for example be established into the system, analyzing the geographic distribution of query and image locations.

Resource selection for precise search in geographic IR

Blank and Henrich [2010b, 2012a] analyze the applicability of HFS(e) and UFS(e) for approximate k-NN query processing and compare those techniques with the basic BB and GRID approaches. This work is extended in Kufer et al. [2012, 2013] and Kufer and Henrich [2014] to precise search and more elaborate resource description and selection techniques are compared with HFS(e)/UFS(e). Kufer et al. [2012] focuses on the extension to precise search and the comparison of UFS(e)/HFS(e) with alternative techniques adopting ideas from computational geometry (see Preparata and Shamos [1985]) and SAMs (see Samet [2006]) such as the R-tree [Guttman, 1984], the grid file [Nievergelt et al., 1984], and the k-d-tree [Bentley, 1975]. Kufer et al. [2013] and Kufer and Henrich [2014] combine different resource description techniques in the context of precise k-NN query processing and adopt additional quantization schemes inspired by the Buddy-tree [Seeger and Kriegel, 1990].

5.2. RS4MI for Precise Range Query Processing

In this section, we compare three different resource description and selection schemes for precise range query processing in arbitrary metric spaces. The experimental setup is outlined in section 5.2.1. In section 5.2.2, the technique from Berretti et al. [2002a,b, 2004], as introduced in section 3.3.2 on pages 67–68, is analyzed. Another approach—based on k-medoids clustering and used as a second comparison baseline for RS4MI—is presented in section 5.2.3. RS4MI range query processing for precise search is described and analyzed in section 5.2.4. Finally, section 5.2.5 subsumes the main results of the experimental comparison.

5.2.1. Experimental Setup

Again, we analyze a scenario where each peer knows the resource description of every other peer. As underlying data collection, we use the same image collection as described in section 5.1.1 on pages 102–104 consisting of 233 827 images crawled from Flickr. Assigning images to peers based on the Flickr user ID results in 10 601 peers. The distribution of the number of images per peer is visualized in figure 5.1 on page 103. Pivots for summary creation and peer ranking in case of RS4MI are randomly chosen from the external image collection with 45 931 images which is also introduced in section 5.1.1 on pages 102–104.

As feature descriptor, we use the unquantized version of the Color and Edge Directivity Descriptor (CEDD)[52]. It results in a 144-dimensional feature vector of four-byte floats and thus in total $\beta_o = 576$ bytes per image. CEDD has the potential to outperform the MPEG-7 features for CBIR [Chatzichristofis et al., 2010]. The Hellinger metric $dist_H$ outlined in section 2.2.6 is applied—converting the non-metric squared chord distance $dist_{sc}$ into a metric. It is shown in Liu et al. [2008] that $dist_{sc}$ provides good retrieval results in case of CBIR. However, our analysis does not focus on search effectiveness in CBIR and thus the choice of an effective feature descriptor in combination with a suitable distance metric is out of the scope of our work.

[52] CEDD features [Chatzichristofis and Boutalis, 2008] are extracted using the Lire library obtained from *http://www.semanticmetadata.net/lire/* (last visit: 17.10.2014).

	min	q25	median	mean	q75	max
result size	1	4	21.5	126.7	94.3	2028
peers	1	3	18	72.0	76.3	654

Table 5.3. — Statistics of the *result size* and the number of *peers* administering result documents for the 200 range queries with a search radius of $r = 0.5$.

Our general setting offers an intrinsic dimensionality as defined in formula 2.20 on page 37 of almost ten ($\rho = 9.9$) and thus represents a rather hard indexing task.

In the experiments, query objects are randomly chosen from the underlying data collection. This seems reasonable in case of range queries $range(q, r)$ relying on the query-by-example paradigm. Again, resource selection performance is measured by analyzing the fraction of queried peers to retrieve all images with feature objects lying within distance r from q. In addition to search efficiency, the size of the resource descriptions is analyzed. If not mentioned otherwise, summaries are compressed with gzip as before[53].

We evaluate 200 range queries with search radius $r = 0.5$ for every parameter setting. Table 5.3 shows statistics on the number of database objects in the search result and from how many peers they are obtained. Relevant documents are on average found at 72 peers. An optimal resource selection would thus on average only contact $\frac{72}{10601} \approx 0.7\%$ of the peers to retrieve the relevant documents.

5.2.2. M-tree-based Local Clustering for Comparison

As a first comparison baseline for RS4MI, we analyze the approach from Berretti et al. [2002a,b, 2004] outlined in section 3.3.2 on pages 67–68. In order to do so, we use revision 27 of the publicly available M-tree library[54] which is also used in chapter 4. The approach mainly depends on two parameters. A cluster radius threshold θ and the block/node size of the M-tree $\beta_{\mathrm{b}}^{\mathrm{M}}$ are the keys for trading-off the selectivity of the

[53] The runtime complexity for building the resource descriptions is not analyzed in this work. This task is parallelized in a real-world scenario with every peer computing its resource description and hereby all promising approaches subsumed in section 5.2.5 are suitably fast.

[54] see *http://mufin.fi.muni.cz/trac/mtree/*, last visit: 3.10.2014

resource descriptions versus their size. The effects when varying these two parameters are evaluated in the following.

An insertion of all database objects of a resource into an M-tree and the threshold-based search algorithm for generating the resource description results in multiple, possibly overlapping cluster balls. A pivot together with an associated covering radius (representing a metric ball maintained in a node entry of the M-tree) is stored in the resource description for every cluster to enable precise range query processing. With this information, the pruning rule based on the range-pivot distance constraint (see section 2.4.2) testing the overlap between the query ball and the cluster balls is then applied during search for the pruning of non-relevant peers, that is, peers with no relevant documents.

Analysis of M-tree-based local clustering

In the upper left quadrant of figure 5.10, the resource selection performance of the summaries arising from the M-tree-based local clustering approach is shown. The lower left quadrant depicts the average summary sizes. To understand this figure, the following aspects have to be considered: (1) A block size of 576 bytes corresponds to leaf nodes containing one object each. In this case, the M-tree implementation assures that inner nodes (including the root node) are bigger and the degree of each inner node is two. In general, an M-tree block size of $\beta_{\mathrm{b}}^{\mathrm{M}}$ means that a leaf node contains at most $\lfloor \beta_{\mathrm{b}}^{\mathrm{M}} / \beta_{\mathrm{o}} \rfloor$ objects. Hence, for example a node size of 18 432 corresponds to leaf nodes containing at most 32 objects in case of $\beta_{\mathrm{o}} = 576$. (2) A cluster radius threshold of 0.01 has the consequence that the summary roughly contains clusters describing exactly the leaf nodes. At the other extreme, a cluster radius threshold of 16 384 is big enough to yield a summary with only a single cluster representing the root node of the M-tree containing the complete set of objects on the peer[55].

With the above information in mind, we can interpret the left side of figure 5.10. If we consider the average summary size in dependence of the cluster radius threshold (lower left quadrant), it becomes obvious that the summary sizes decrease for higher threshold values. The reason

[55] Please note that in our case, all pairwise object distances in case of $dist_{\mathrm{H}}$ are at most 2. However, due to heuristic upper bound approximation of the cluster radii in the inner nodes of the M-tree, values bigger than 2 exist in the tree.

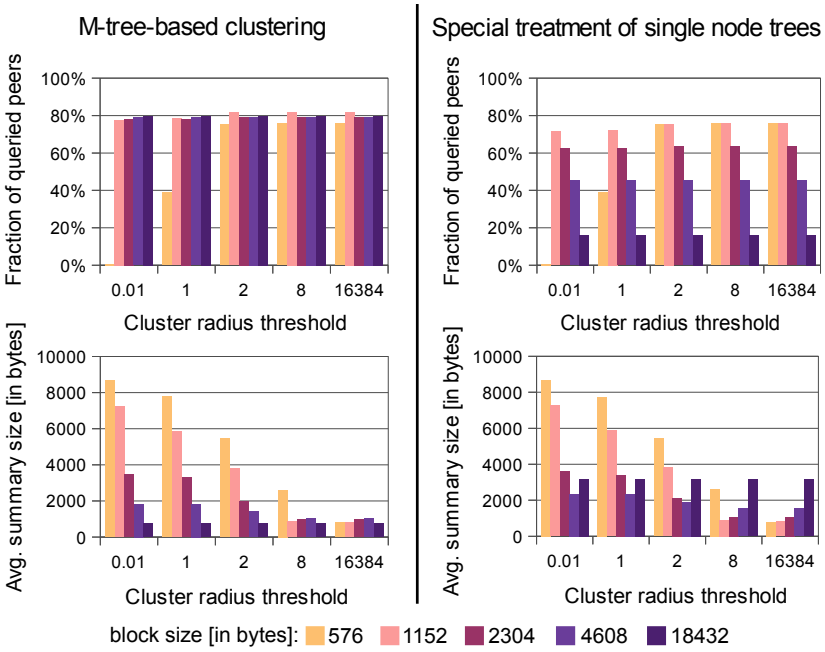


Figure 5.10. — M-tree-based local clustering (left) with special treatment of single node trees (right).

is that for higher threshold values the clusters for the summaries are taken from higher levels of the M-tree. Obviously, this effect is only given for small block sizes because for higher block sizes (e.g. the utmost right bars) the height of the M-trees is extremely low anyway.

The upper left quadrant of figure 5.10 shows the resource selection performance measured by the fraction of queried peers. Let us first consider the fraction of queried peers in dependence of the cluster radius threshold. As a special case, the block size of 576 together with a cluster radius threshold of 0.01 has to be considered. In this situation, each leaf node contains only a single item and because of the low threshold value, the clusters describing the leaf nodes are included in the summaries. Consequently, the summaries exactly represent the objects on each peer. Based on this information, a querying peer can exactly determine the peers containing objects in the query ball and therefore the fraction of queried peers corresponds to the theoretical optimum

of 0.7%. However, this result is achieved by a complete replication of the database objects within the network on all peers. Unfortunately, these parameter settings are not realistic for huge networks. Neither a threshold value yielding only leaf nodes nor a node size storing only one object per node are practical.

Despite from these special cases, the fraction of queried peers is roughly between 70% and 80%. It is also interesting to consider the effect of the block sizes for example for a cluster radius threshold of 8. With this threshold, only in very rare cases the clusters used in the summary are taken from lower levels of the tree. With the block size of 576 bytes, peers with only one image are represented by one cluster in the summary and peers with two or more images are (with some exceptions) represented by two clusters since the fan-out of the root node is two in this case. With the block size of 1152 bytes, peers with one or two images are represented by one cluster in the summary and peers with three or more images are (with few exceptions) represented by two clusters. The less precise representation of peers with two images results in an increase of the peers which have to be considered from 75.8% to 81.8% and at the same time reduces the average summary size drastically. With the block size of 2304 bytes, peers with one to four images are represented by one cluster in the summary and peers with five or more images are (again with few exceptions) represented by two to four clusters. Hence, the summaries of small peers become less accurate but the summaries of bigger peers become more accurate, since the root node of the M-tree now has up to four children. Obviously, these considerations can be continued for bigger block sizes.

The above results inspired us to change the approach marginally to exploit the long-tail distribution of images on peers (see figure 5.1 on page 103). Over 50% of the peers contain seven or less images. As a consequence, if the summaries of these small peers contain the exact database objects, only the peers out of these 50% which really contribute to the range query result would have to be visited. With such a technique, we can easily outperform the approaches presented above which have to address 70% to 80% of the peers.

To integrate this idea into the M-tree-based local clustering approach, we use a special treatment for situations where the M-tree consists of only one (leaf) node—which is typical for small peers. In this case, the summary now contains one cluster with zero radius for each object in this leaf node instead of one single cluster with a comparatively large

radius describing the whole node. As a consequence, for example with a block size of 18 432 bytes, a peer maintaining 32 objects fitting into one single leaf node is now represented by a summary containing these 32 objects as 32 single clusters with the objects as centers and zero radius.

The effect of this variation is shown on the right hand side of figure 5.10. Let us—again at a cluster radius threshold of 8—consider the bars representing a node size of 4608 bytes, respectively at most eight objects. In this case 5643 (i.e. 53%) of all peers are represented exactly in the summaries. This allows to reduce the number of queried peers to 45.3%. The average summary size is now 1.5 kilobytes compared to 1.0 kilobytes without the special treatment of small peers.

Although the improvements achieved with this variation are impressive, it remains problematic that we have such indirect and hard to handle parameters; the threshold value θ, the block size $\beta_{\mathrm{b}}^{\mathrm{M}}$ of the M-tree, and the special treatment of trees comprising only one node. In fact, it seems easier to use an explicit clustering approach with more intuitive parameters. This leads us to the local k-medoids clustering presented in the following.

5.2.3. Local k-medoids Clustering for Comparison

Some of the resource selection approaches apply k-means clustering to cluster the database objects of a peer (see e.g. Doulkeridis et al. [2007] and El Allali et al. [2008]). However, k-means—due to the mean calculation—is not applicable in arbitrary metric spaces. When using k-medoids clustering instead (or any other suitable algorithm applicable in general metric spaces, see section 3.1), an additional baseline technique for the comparison with RS4MI can be designed. In this case, each peer clusters its local data collection and stores cluster balls, that is, cluster centers and corresponding covering radii in its resource description. This results in similar resource descriptions as the approach discussed earlier in section 5.2.2. The resource description of a peer thus consists of a list of cluster center and covering radius pairs.

There are two general options for determining k—the desired number of clusters of a peer needed as an input parameter to k-medoids clustering. As one alternative, the maximum number of allowed clusters per peer k can be set as a global threshold being identical for all peers. Of course, peers with less than k distinct database objects directly transfer

them and do not apply clustering. On the other hand, algorithms which automatically detect an appropriate number of clusters can be used by the peers. Multiple of these algorithms are presented in literature (for references see e.g. Tibshirani et al. [2001]). Our choice of algorithms in the following is by no means exhaustive. It is our intention to evaluate different techniques which return a range of average numbers of clusters per peer when applied to our scenario.

Rule of thumb (r.o.t.): A coarse rule of thumb is presented in Mardia et al. [1979, *p.* 365]. It suggests to calculate the number of clusters of a dataset of size $|O|$ as $k \approx \sqrt{|O|/2}$. Thus, we use $k = \lceil \sqrt{|O_a|/2} \rceil$ when applying this rule of thumb to the database O_a of a peer $p_a \in P$.

This rule of thumb directly calculates the number of desired clusters. In contrast, the techniques presented in the following are applied in an iterative process. A single key figure results for a specific value of k. Various values of k are thus to be tested in order to select the best k minimizing/maximizing the key figure. To improve the runtime performance, also when applying the rule of thumb, the FAMES extension to k-medoids clustering is used (see section 3.1). For determining the initial candidate set of medoids, in ten runs we minimize the sum over all clusters of within-cluster object-to-medoid distances.

Besides the rule of thumb, we apply three variants of the well-known gap statistic. The gap statistic [Tibshirani et al., 2001] is frequently used and offers the property that—in contrast to many alternative approaches—it can also detect the presence of only a single cluster.

GAP: The gap statistic as originally proposed in Tibshirani et al. [2001] is based on a sampling process which is not directly applicable in arbitrary metric spaces. However, as suggested in Tibshirani et al. [2001], when only distance information is available, a specific mapping technique such as multidimensional scaling can be used to obtain feature vectors in a low-dimensional space, which provide the basis for the sampling process. In our experiments, we directly apply ten sampling steps on the initial feature vectors without the use of an additional mapping technique in order to obtain a best case comparison baseline against which we compare our approach RS4MI.

GAP$_\mathbf{w}$ as introduced in Yan and Ye [2007] slightly modifies the weighting scheme of the gap statistic.

GAP$_\mathbf{n}$ represents another modification of the gap statistic and all logarithms used in the formulas of the gap statistic are removed [Mohajer et al., 2011].

	r.o.t.	GAP	GAP_w	GAP_n	SIL_1	SIL_2
visited peers	67.4%	73.7%	75.3%	76.4%	76.4%	65.7%
clusters per peer	3.1	2.3	1.9	1.5	2.3	2.8
summary size [in B]	1350.3	1048.4	880.7	722.9	1029.5	1232.2

Table 5.4. — Results for local k-medoids clustering with an automatic determination of the number of clusters k.

SIL_1 and SIL_2: The silhouette technique [Rousseeuw, 1987] is also adapted as a means for calculating the desired number of clusters of a peer. It is only applicable in case of $k > 1$. Thus, two alternatives are used in our experiments. If two is indicated as the optimum cluster number, we set $k = 1$ in case of SIL_1; $k = 2$ is used in case of SIL_2. Peers with only a single database object of course only store a single cluster in the resource description.

To determine an appropriate value for k, the above-mentioned approaches based on the gap statistic and the silhouette technique are iteratively tested on every peer $p_a \in P$ until $k = \lfloor min(2\sqrt{|O_a|}, |O_a|) \rfloor$ with $|O_a|$ denoting the number of documents/images of a peer p_a.

Analysis of local k-medoids clustering

Table 5.4 shows the average fraction of visited peers, the average number of clusters per peer, as well as average summary sizes in case of local k-medoids clustering when automatically determining the number of clusters per peer. The rule of thumb (r.o.t.) leads to decent resource selection performance at the cost of comparatively big summaries. A better performance is achieved by the SIL_2 approach with at the same time more space efficient resource descriptions.

Using the gap statistic for determining the number of clusters of a peer results in average summary sizes of approximately 1 kB and 73.7% of peers being contacted for retrieving all relevant documents. GAP_w and GAP_n lead to fewer numbers of clusters per peer and thus more space efficient resource descriptions. However, both perform worse than GAP.

SIL_1 offers similar average summary sizes as GAP. The average number of clusters per peer is in both cases approximately 2.3 but GAP

k	1	2	4	8	32	128
queried peers	79.9%	68.7%	54.4%	37.7%	13.1%	2.7%
clusters per peer	1.0	1.9	3.4	5.6	12.2	18.2
summary size	525.0 B	867.2 B	1.4 kB	2.3 kB	4.9 kB	7.3 kB

Table 5.5. — Results for local k-medoids clustering with all peers using the same global k.

shows a better resource selection performance with fewer visited peers. SIL_1 always assumes one cluster when there might be two which GAP can detect. SIL_2 visits fewer peers than the other competing approaches at the cost of storing on average 2.8 clusters per peer in the summaries, and is even better than the rule of thumb which identifies on average 3.1 clusters per peer.

A main drawback of the k-medoids approaches analyzed in this section so far is that the summary sizes cannot be influenced by any kind of design parameter of the approach. An alternative in this respect is to globally specify k, the maximum allowed number of clusters per peer. In this case, peers p_a with $|O_a| \leq k$ store all database objects in their summary. Since for some peers the number of database objects is smaller than k, the average number of clusters per peer becomes smaller than k as well. This scenario which is thus similar to the special treatment of single node trees in section 5.2.2 is evaluated in table 5.5.

The explicit definition of an upper bound for the number of clusters allows for a direct and accurate adjustment of summary sizes and selectivity. This gives a clear advantage over the M-tree-based approach and also over the approaches outlined in this section which automatically determine a suitable number of clusters per peer. However, if very small summaries are necessary, the flexibility is restricted by the discrete values of k.

It can be observed from table 5.5 that only in cases where the maximum desired number of clusters per peer is set to $k = 1$ or $k = 2$, average summary sizes with less than 1 kB can be achieved. If a maximum of two clusters is allowed, 68.7% of the resources are visited with an average summary size of 867 bytes. To further reduce this number, only a single cluster per peer can be allowed. However, almost 80% of

the peers are queried in this case with an average summary size of 525 bytes.

5.2.4. RS4MI Variants and their Evaluation

RS4MI can make use of all pruning rules mentioned in section 2.4. A global set of m reference objects C—the same for all resources—is applied to assign a database object $o \in O_a$ of a resource $p_a \in P$ to the closest cluster center $c^* = \arg\min_{c_i \in C} \ dist(c_i, o)$.

The set of reference objects is transferred to the peers together with updates of the P2P software so that no additional network load is imposed during the operation phase of the P2P IR system. Such an approach is for example proposed in Blank et al. [2007].

Multiple variants of RS4MI resource descriptions are evaluated in the following to find the best alternative. These variants can make use of different pruning rules and thus differ in resource selection performance and average summary size.

RS4MI$_{\mathbf{1xxxx}}$: Here, only a single bit is stored per cluster to indicate if any database objects lie in the very cluster or not. This results in a bit vector of size m and thus resource descriptions with $\mathcal{O}(m)$ space complexity, conceptually the same resource descriptions as UFS$_m$ introduced in section 5.1.1. The pruning rule based on the double-pivot distance constraint outlined in section 2.4.1 is the only rule which can be used in this case to prune peers from search.

RS4MI$_{\mathbf{x??xx}}$: Resource descriptions offering $\mathcal{O}(m)$ space complexity can also be designed by storing the inner and/or outer cluster shell radii. By doing so, the pruning rule based on the range-pivot distance constraint can be applied on an intra-cluster level (see section 2.4.2). In addition to storing both, inner and outer cluster shell radii for the m clusters (i.e. RS4MI$_{\text{x11xx}}$), we test parameter settings of RS4MI$_{\text{x1xxx}}$ and RS4MI$_{\text{xx1xx}}$ where only inner or outer cluster shell radii are stored, respectively. A single distance value is represented as a four-byte float.

Of course, the pruning rule based on the double-pivot distance constraint can also be applied in this case. If no inner and/or outer cluster shell radius is set for a particular cluster, it is indicated by the summary that the corresponding peer does not administer any database objects within the very cluster. Similarly, the pruning rule based on the double-pivot distance constraint is used by all of the following resource selection schemes whenever applicable.

RS4MI$_{\text{xxx??}}$: If the rules for cluster pruning described in section 2.4 are fully applied, two matrices R^{in} and R^{out} are administered by every peer as its resource description (RS4MI$_{\text{xxx11}}$). This requires $\mathcal{O}(m^2)$ space per resource. As before, a single matrix cell requires four bytes for storing radius information. Both matrices are sent as a resource description and used for the pruning of resources without querying them. We also test parameter settings where only a single matrix R^{in} (RS4MI$_{\text{xxx1x}}$) or R^{out} (RS4MI$_{\text{xxxx1}}$) is used.

Two further combinations are included in the analysis. RS4MI$_{\text{x1xx1}}$ stores inner cluster shell radii and the matrix R^{out} as a resource description. In opposition, RS4MI$_{\text{xx11x}}$ applies outer cluster shell radii and the R^{in} matrix to potentially discard a peer during search.

Finally, we also evaluate a hybrid[56] resource selection scheme where either per-cluster or per-object information is stored in the resource description of a peer—depending on the particular database size of the peer.

Analysis of RS4MI

In the following, different ways of how to best design summaries in case of RS4MI are evaluated. First, summaries storing only per-cluster information are analyzed. Later, a hybrid setting is evaluated.

RS4MI approaches storing per-cluster information. The top left part of figure 5.11 visualizes the resource selection performance for resource descriptions with $\mathcal{O}(m)$ space complexity. It can be observed that RS4MI$_{\text{1xxxx}}$ and thus only applying the pruning rule based on the double-pivot distance constraint does not lead to an acceptable resource selection performance. However, RS4MI$_{\text{1xxxx}}$ with a bit vector as an underlying data structure results in very space efficient resource descriptions, even in case of larger values of m (e.g. $m = 1024$ in figure 5.11, bottom left).

When comparing RS4MI$_{\text{x1xxx}}$ with RS4MI$_{\text{xx1xx}}$, it can be observed that although both approaches have similar average summary sizes,

[56] Note that the term *hybrid* could also denote a resource selection scheme where both per-cluster and per-object information is stored in a resource description of a peer. However, in our case hybrid refers to the case where either per-cluster or per-object information is stored. Of course, RS4MI can also be extended to store feature objects directly in the summaries (e.g. for peers with few documents) similar to the approaches evaluated in section 5.2.2 and section 5.2.3.

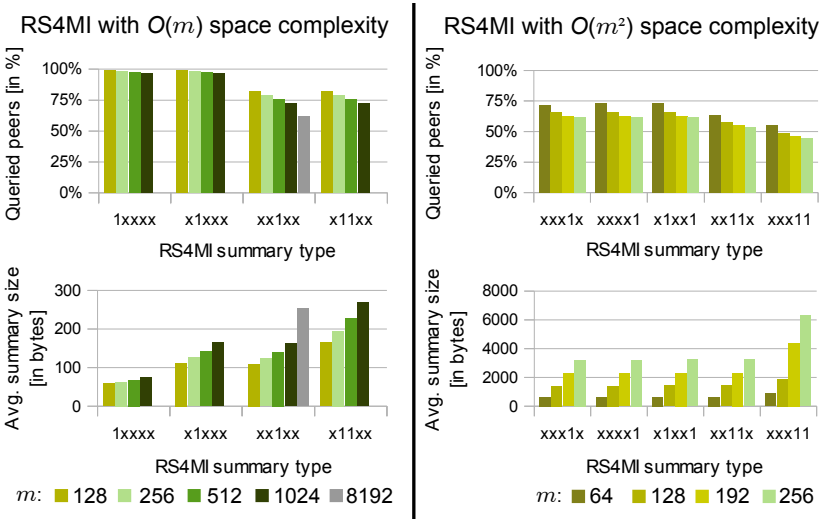


Figure 5.11. — Results of RS4MI for summaries with space complexity $\mathcal{O}(m)$ (left) and $\mathcal{O}(m^2)$ (right).

$\text{RS4MI}_{\text{xx1xx}}$ can discard clearly more peers than $\text{RS4MI}_{\text{x1xxx}}$. Even $\text{RS4MI}_{\text{x11xx}}$ cannot noticeably improve resource selection performance. Thus, $\text{RS4MI}_{\text{xx1xx}}$ with a very large number of reference objects being used (e.g. $m = 8192$ or even more) seems to be a good choice among the approaches considered in the left part of figure 5.11.

Resource descriptions with $\mathcal{O}(m^2)$ space complexity are analyzed in figure 5.11 on the right. For these approaches, a binning technique is applied to reduce the summary sizes. Every four-byte distance value is quantized into a single byte. For quantization, the minimum and maximum distance value from the feature objects of the external collection to every cluster center c_i is determined. The range between these two boundaries per reference object c_i is uniformly quantized into 253 intervals. From the remaining three values, two are used to represent distance values below and above the boundaries. The third remaining value is used to indicate an empty cluster with no entry. Again, it is assumed that the minimum and maximum distances from feature objects of the external collection to the cluster centers c_i are known to all peers in advance and transferred to them with updates of the P2P software so that all peers can estimate the distance bounds from the

quantized values. This information is also small enough to be transferred to participating peers during the operation phase of the P2P IR system.

Figure 5.11 (bottom right) shows that the average summary sizes in case of $\text{RS4MI}_{\text{xxx1x}}$, $\text{RS4MI}_{\text{xxxx1}}$, $\text{RS4MI}_{\text{x1xx1}}$, and $\text{RS4MI}_{\text{xx11x}}$ are very similar. According to the resource selection performance (see figure 5.11, top right) $\text{RS4MI}_{\text{xx11x}}$ applying the R^{in} matrix and an array of length m with covering cluster radii clearly outperforms the other three approaches. Also $\text{RS4MI}_{\text{xxx11}}$ encoding the quantized R^{in} and R^{out} matrices with a small value of for example $m = 64$ is promising. For the feature set being indexed, $\text{RS4MI}_{\text{xxx11}}$ with m being small and $\text{RS4MI}_{\text{xx1xx}}$ with m being big constitute the most promising RS4MI approaches.

To further reduce the summary sizes of $\text{RS4MI}_{\text{xx1xx}}$ with $m = 8192$, alternative compression algorithms can be used. When changing the compression algorithm, summary sizes are reduced on average from 253.2 bytes (gzip) to 222.1 bytes (bzip2), and to 234.2 bytes in case of lzma. Thus, a reduction of approximately 10% seems easily possible[57].

Summary sizes for $\text{RS4MI}_{\text{xxx11}}$ with $m = 64$ can also be reduced. Table 5.6 shows the results. The bzip2 implementation seems to be inappropriate with average summary size noticeably increasing and also lzma does not lead to a significant reduction. Thus, in addition to gzip, bzip2, and lzma, three image compression algorithms are tested where the concatenation of the quantized R^{in} and R^{out} matrices is interpreted as a 2-dimensional 256 bit gray-scale image of size 64×128 pixels. Standard png compression provides some overhead, but paq8o8[58] and especially webp[59] lossless image compression provide more space efficient resource descriptions than gzip; webp in particular by significantly reducing the memory requirements of the summaries of peers with images in few clusters.

Hybrid RS4MI approaches storing per-object information. The RS4MI approaches presented so far solely rely on cluster pruning

[57] Additional compression results are based on the At4J library (*http://at4j.sourceforge.net/*, last visit: 17.10.2014) and contributing libraries such as 7-Zip (*http://www.7-zip.org/*, last visit: 17.10.2014) and Apache Commons Compress (*http://commons.apache.org/compress/*, last visit: 17.10.2014).

[58] see *http://mattmahoney.net/dc/#paq*, last visit: 17.10.2014

[59] see *https://developers.google.com/speed/webp/*, last visit: 17.10.2014

	gzip	bzip2	lzma	png	paq8o8	webp
summary size	867.0 B	1020.1 B	863.4 B	880.4 B	803.1 B	777.7 B

Table 5.6. — Average summary sizes for $\text{RS4MI}_{\text{xxx11}}$ with $m = 64$.

m	1	2	4	8	12	16
queried peers	97.3%	95.5%	90.9%	82.5%	75.8%	73.5%
summary size [in bytes]	136.5	216.9	372.5	676.2	976.1	1274.6

Table 5.7. — Results when solely applying pivot filtering.

principles. Object pruning and thus the storage of per-object information in the resource descriptions is not considered. However, the distribution of peer sizes visualized in figure 5.1 on page 103 indicates many peers with few documents. Thus, at least for peers with very few documents it seems beneficial to store per-object summary information and apply pivot filtering (see section 2.4.3).

First, we analyze settings where only object-pivot distances (and thus no per-cluster information) are used in the resource descriptions. Table 5.7 shows the results for different numbers of reference objects. Such an undifferentiated approach is inappropriate and results in very big summary sizes for peers with many documents. When using 16 reference objects and thus encoding 16 object-to-pivot distance values per database object, 73.5% of peers are queried with resource descriptions of 1.3 kB on average.

We also analyze a hybrid resource description scheme with peers choosing either per-object or per-cluster summarization, depending on $|O_a|$, the number of documents a peer p_a administers. To roughly estimate the number of possible reference objects per database object for which object-pivot distances are stored in the summary of a peer p_a, the formula $\mathit{numRefsPerObject} = \lfloor \tilde{\beta}_{\text{rd}}^{\text{avg}} / (4 \cdot |O_a|) \rfloor$ is applied. The parameter $\tilde{\beta}_{\text{rd}}^{\text{avg}}$ hereby denotes the desired average summary size in bytes and a factor of four in the denominator is used since a single distance value is represented as a four-byte float. From table 5.8, it can be observed that this estimate of the average summary size roughly holds. If $\mathit{numRefsPerObject} > 0$, pivot filtering is applied on the basis

$\tilde{\beta}_{\mathrm{rd}}^{\mathrm{avg}}$	**100**	**200**	**400**	**600**	**800**	**1000**
	(8439)	(9612)	(10 202)	(10 385)	(10 459)	(10 503)
$m = 512$	81.1%	76.0%	69.9%	66.0%	63.4%	61.4%
	174.9 B	251.9 B	426.9 B	585.1 B	759.6 B	919.2 B
$m = 4096$	80.3%	75.7%	69.8%	65.9%	63.4%	61.4%
	214.8 B	277.8 B	443.0 B	596.8 B	768.9 B	926.8 B

Table 5.8. — Results for hybrid RS4MI summaries. Table cells show the fraction of queried peers (top) and the average summary size (bottom). The number of peers applying pivot filtering is given in brackets (10 601 peers in total).

of per-object resource descriptions. Otherwise, per-cluster summaries $\mathrm{RS4MI}_{\mathrm{xx1xx}}$ are used as before.

Table 5.8 visualizes results of the hybrid resource selection scheme when varying $\tilde{\beta}_{\mathrm{rd}}^{\mathrm{avg}}$ and m. The number of peers applying pivot filtering is denoted in brackets. If these results are compared with the ones applying only per-cluster information (see figure 5.11 on page 127), several approaches can be outperformed; for example a parameter settings with $m = 512$ and $\tilde{\beta}_{\mathrm{rd}}^{\mathrm{avg}} = 600$ seems promising. However, resource selection performance of $\mathrm{RS4MI}_{\mathrm{xx1xx}}(m = 8192)$ can only be achieved with much bigger average summary sizes since compression techniques in case of $\mathrm{RS4MI}_{\mathrm{xx1xx}}(m = 8192)$ can dramatically reduce the summary size of peers with documents in only few clusters.

5.2.5. Brief Comparison of the Approaches

We now focus on the most promising techniques of the different approaches discussed so far to assess their potential in large scale networks. In sections 5.2.2 and 5.2.3, the techniques yielding an exact representation of small peers—either applying a special treatment for single node M-trees or defining a desired value for k—are promising in situations with a long-tail distribution of the objects over the peers. Small peers directly transferring the database objects is also possible for RS4MI to further improve resource selection performance.

Table 5.9 gives a brief overview of different approaches discussed in section 5.2.2, section 5.2.3, and section 5.2.4. All approaches result in

	queried peers	avg. summary size
M-tree($\beta_{\mathrm{b}}^{\mathrm{M}}$=576, θ=16384)	75.8%	813.2 B
2-medoids	68.7%	867.7 B
$\mathrm{RS4MI}_{\mathrm{xx1xx}}$($m$=8192)	62.2%	253.0 B
$\mathrm{RS4MI}_{\mathrm{xxx11}}$($m$=64)	57.2%	880.7 B

Table 5.9. — Comparing different approaches with results averaged over ten runs with the 200 queries each.

average summary sizes below 1 kB. Conceptually, the resource selection techniques based on M-tree and k-medoids clustering are similar to each other, both applying local clustering and transferring cluster centers and corresponding covering radii. The parameterization of the techniques is crucial for both approaches. In this regard, the k-medoids-based local clustering approach with its easy to interpret design parameter k is more handy than the M-tree-based clustering and also resource selection performance (as briefly summarized in table 5.9 and in more detail outlined in sections 5.2.2 and 5.2.3) does not give a clear evidence for using the approach based on the M-tree. $\mathrm{RS4MI}_{\mathrm{xx1xx}}(m = 8192)$ leads to better resource selection performance with significantly smaller average resource description sizes. The number of queried peers is further reduced by $\mathrm{RS4MI}_{\mathrm{xxx11}}(m = 64)$ at the cost of larger summaries, comparable with those of 2-medoids. Of course, it is also possible to use different RS4MI summary types within a single P2P IR system.

5.3. RS4MI for Precise k-Nearest Neighbors Query Processing

Precise RS4MI range query processing is addressed in section 5.2. As another important query type for similarity search with RS4MI, this section focuses on precise k-NN query processing. At first, a conceptual algorithm is presented in section 5.3.1. Afterwards, the resource selection performance of the algorithm is assessed in section 5.3.2.

Data:

k	the desired number of NNs
q	the query object
C	the global set of reference objects
L_P	initially unranked list of peers
n_{pip}	number of peers queried in parallel
$topk[\,]$	result array of length k with $\langle oid(o), dist(q,o)\rangle$ pairs, $topk_p$ indicates a local result array from a peer p

```
1  L_q = determinePermutationList(q, C)
2  r = determineUpperBoundOfkNN(k, L_q, L_P)
3  L_P = rankPeers(r, L_q, L_P)
4  while L_P ≠ ∅ do
5  |   L'_P = fetchNextPeers(n_pip, L_P)
6  |   for p in L'_P do
7  |   |   if peerPruningNotPossible(p, r, L_q) then
8  |   |   |   topk_p = queryPeer(k, p, q, r)
9  |   |   |   topk = update(topk, topk_p)
10 |   |   end
11 |   end
12 |   L_P = rankPeers(r, L_q, L_P)//only for particular ranking schemes
13 end
```

Figure 5.12. — A conceptual algorithm for precise RS4MI k-NN queries.

5.3.1. A Conceptual Algorithm for Precise RS4MI k-Nearest Neighbors Query Processing

An algorithm for precise RS4MI k-NN query processing is outlined in figure 5.12. It assumes access to the set of resource descriptions in a peer index. The algorithm starts in line 1 by determining the list L_q as introduced in definition 4 on page 28 and used in the algorithm in figure 4.2 on page 79. Note that also here L_q contains—in addition to the cluster ID—precomputed distance values for every entry so that they do not have to be recomputed. An upper bound for the distance between the query object q and the k-th NN is determined from the summaries in line 2. This upper bound distance is used as the initial

search radius r and it is determined from the set of resource descriptions of all peers to be as tight as possible[60]. In our scenario, the set of reference objects $C = \{c_i \mid 1 \leq i \leq m\}$ is known to all peers. Upper bound distances $\hat{d}_i = dist(q, c_i) + r_i^{\text{out}}$ can thus for example in case of $\text{RS4MI}_{\text{xx1xx}}$ be calculated for every populated cluster of a resource. The best performing RS4MI resource descriptions $\text{RS4MI}_{\text{xx1xx}}$ and $\text{RS4MI}_{\text{xxx11}}$ mentioned in section 5.2.5 maintain covering radius information r_i^{out}. Thus, they are directly applicable here. If this is not the case, the estimation of a tight initial search radius can also be skipped and it can be set to positive infinity. In line 3, the peers are ranked. The list L_P is sorted for determining the order in which peers are queried. A particular algorithm for the peer ranking step is the approach from Eisenhardt et al. [2006] presented in section 3.3.4 on pages 72–73. It is evaluated together with alternative ranking algorithms in the following section 5.3.2. The list of peers L_P is processed as long as there are more entries (line 4). A list segment L'_P with up to n_{pip} peers is fetched in each round and removed from the beginning of the list L_P (line 5). For all peers from this list segment, it is then tested if the very peer can be pruned from search by applying the pruning rules outlined in section 2.4 (line 7). If this is not the case, the query is sent to the very peer and the enquirer waits for the results (line 8). Multiple peers can of course be queried in parallel in a distributed scenario. We simulate such a behavior when analyzing the quality of different peer ranking mechanisms and fetch up to n_{pip} peers in every round before reranking the peers in line 12[61]. This reranking is only necessary for some peer ranking schemes (see section 5.3.2). The array with the global result is updated whenever a peer which responds to the query provides local results which can improve the global result (line 9). In this case, if the $dist(q, o)$ value of the k-NN changes, also the search radius r decreases.

With some modifications and optimizations, the runtime complexity of our k-NN algorithm is not an issue. If in line 2 of the algorithm the runtime for considering all populated clusters of all peers becomes too high, sampling techniques can be applied when determining the upper bound distance or the initial search radius can even be set to

[60] Here, an algorithm is presented which successively reduces the search radius for determining the result set. Alternative algorithm designs for example with an increasing search radius are of course also possible.

[61] Of course, in a distributed scenario, additional peers can be queried before the results from all n_{pip} peers are obtained.

positive infinity. In case of a big number of peers, the computation in line 3 reduces to $\mathcal{O}(n \log n)$ if only the first few clusters and not all m clusters are considered for the peer ranking (see section 5.1.1 on pages 107–108). Testing all n peers if they have to be queried is in $\mathcal{O}(m\,n)$ when we assume that in the worst case no cluster pruning can be performed. However, such cases are hypothetical. Furthermore, a peer index can be applied to speed-up the peer pruning. The cost for querying a remote peer which depends on network delay and other aspects cannot be influenced by the algorithm design. Finally, the additional peer ranking steps in line 12 of the algorithm become obsolete if the most promising peer ranking schemes from the following section are applied.

5.3.2. Evaluating Precise RS4MI k-Nearest Neighbors Query Processing

For an evaluation, we use the same setup as presented in section 5.2.1 when evaluating precise RS4MI range query processing. We analyze the resource selection performance of different peer ranking approaches for the algorithm presented in figure 5.12 on page 132 in case of RS4MI_{00100}, a particularly promising RS4MI resource selection scheme for precise range query processing (see section 5.2.5).

As a first resource ranking approach, we slightly adapt the algorithm from Eisenhardt et al. [2006] which is outlined in section 3.3.4 on pages 72–73. RS4MI_{00100} resource descriptions do not explicitly contain frequency counts such as HFS or binarized frequency information such as UFS. However, the presence of a covering radius in the RS4MI_{00100} resource descriptions indicates one or more feature objects in a particular cluster. Thus, we can apply the algorithm from Eisenhardt et al. [2006] outlined in section 3.3.4 on pages 72–73 in the same way as in case of UFS. This approach is in the following evaluation tables denoted as *stable*.

As an alternative, we use the *mindist* function applied by the M-tree for identifying promising subtrees in case of a k-NN query [Ciaccia et al., 1997, *p.* 429]:

$$mindist(q, [c_i]_{r_i^{\text{out}}}) = max\big(dist(q, c_i) - r_i^{\text{out}}, 0\big) \tag{5.1}$$

Formula 5.1 computes a lower bound distance from the query object to database objects in a particular cluster ball of a peer by subtracting the radius of the cluster ball r_i^{out} from the distance $dist(q, c_i)$ between the query object and the cluster center. In case of a negative value, *mindist* is set to zero because this implies that the cluster ball contains the query object.

When comparing two peers p_a and p_b, *mindist* values are determined for all m clusters of the two peers and ascendingly ordered per peer. In case of an unpopulated cluster, the *mindist* value is set to positive infinity. The two lists of ordered *mindist* values are processed in parallel from the beginning to the end. If the corresponding *mindist* value of peer p_a is smaller than the value of peer p_b, p_a is ranked before p_b, and vice versa. If the two *mindist* values are equal, the decision is made by taking the next *mindist* value from each list and comparing these two values, and so on. If the end of the list is reached, the two peers are ranked at random. This is a very rare case, though.

In another approach, we rank peers based on the definition of the proximity *prox* between two metric balls from Amato [2002, *p.* 72] and Amato et al. [2003, *p.* 197] given in formula 5.2:

$$prox([q]_r, [c_i]_{r_i^{\text{out}}}) = \begin{cases} 0 & \text{if } r + r_i^{\text{out}} < dist(q, c_i) \\ \frac{r + r_i^{\text{out}} - dist(q,c_i)}{2 \cdot d^{\max} - dist(q,c_i)} & \text{if } max(r, r_i^{\text{out}}) \leq min(r, r_i^{\text{out}}) + dist(q, c_i) \\ \frac{2 \cdot min(r, r_i^{\text{out}})}{2 \cdot d^{\max} - dist(q,c_i)} & \text{otherwise} \end{cases} \tag{5.2}$$

If the sum of the two radii $r + r_i^{\text{out}}$ is smaller than the distance $dist(q, c_i)$ between the two ball centers, there is no overlap between a query ball $[q]_r$ and a cluster ball $[c_i]_{r_i^{\text{out}}}$. Thus, the proximity is zero in this case (see line 1 in the case analysis in formula 5.2). Line 2 of the case analysis in formula 5.2 addresses partial overlap and line 3 analyzes the covering of one ball by the other where twice the radius of the smaller ball is normalized and used as the ranking criterion. The proximity definition in formula 5.2 assures normalized values in the range $[0, 1]$. If both metric balls completely overlap, the proximity becomes one. This normalization is however not needed in our scenario

and thus formula 5.2 can be adapted for the resource ranking even in cases when the distance metric has no upper bound $d^{\max}$.

The ranking of the peers is based on the sum of the *prox* values over all populated clusters of a peer—the higher the corresponding cumulated *prox* value, the better for the ranking. Also here, if cumulated *prox* values of two peers are equal, a random ranking decision is made.

The proximity definition in formula 5.2 produces values "*linearly proportional to the overlap of the regions*" [Amato, 2002, *p.* 72]. It is used here to mimic an approach only roughly sketched in Berretti et al. [2002b, *p.* 199]. This approach estimates the number of relevant database objects a resource can provide and can conceptually be described by the formula $\sum_{i=1}^{m} f(r, r_i^{\text{out}}, dist(q, c_i))$. The function f is influenced by two aspects: a) the amount of overlap between the query ball and the cluster ball and b) the number of database objects in the particular cluster (i.e. the cluster population). In our evaluation, a) is reflected by the proximity definition given in formula 5.2. We do however not consider the encoding of cluster populations b) in the RS4MI_{00100} resource descriptions. Since we propose to use a large number of reference objects m, cluster population counts beyond zero and one become rarer and rarer with increasing m. In a particular run, for example when using the Euclidean distance and $\text{RS4MI}_{00100}(m = 256)$, 28.6% of the populated clusters have population counts bigger than one whereas in case of $\text{RS4MI}_{00100}(m = 16384)$ this number reduces to 7.0%. Of course, extending RS4MI with population counts is possible.

As additional comparison baselines, we also rank peers at *random* and based on the peer *size*, that is, the number of documents a peer administers.

Results are shown in table 5.10 for a run with 200 queries using the Hellinger distance $dist_{\text{H}}$ and a value of $n_{\text{pip}} = 10$. It can be observed that resource rankings based on the algorithm from Eisenhardt et al. [2006] (*stable*) and based on the *mindist* function are the two most promising approaches. For retrieving 16 of the 20 NNs (i.e. 80%), 3.1% and 3.2% of the peers must be queried on average. If all 20 NNs are to be retrieved, these numbers increase up to 9.4% for both approaches. As can also be observed from table 5.10, the k-NN query processing ends for both approaches after contacting 64.0% of the peers. Even if all 20 NNs have already been found, it is impossible to stop immediately and additional peers must be queried before being able to safely discard all successive peers.

	stable	*mindist*	*prox*	*size*	*random*
60% found	1.5%	1.6%	4.3%	9.4%	40.2%
70% found	2.2%	2.3%	6.5%	13.9%	46.6%
80% found	3.1%	3.2%	9.9%	20.2%	52.7%
90% found	5.0%	5.1%	15.2%	30.0%	58.7%
100% found	9.4%	9.4%	26.4%	47.4%	64.7%
end	64.0%	64.0%	64.0%	64.5%	68.4%

Table 5.10. — Fraction of peers visited [in %] for retrieving a fraction of the 20-NNs in case of $\mathsf{RS4MI}_{00100}$ with $m = 8192$ when using the Hellinger distance.

With respect to resource selection performance, there is only a marginal gap between these two ranking approaches. Also from a runtime perspective, both approaches are promising since they do not apply the search radius r for the peer ranking and thus a reranking as described in line 12 of the algorithm shown in figure 5.12 becomes obsolete.

A proximity based resource ranking as indicated in the *prox* column of table 5.10 clearly performs worse. In addition, since the proximity definition in formula 5.2 relies on the search radius r, this approach is also worse from a runtime perspective if line 12 of the algorithm shown in figure 5.12 is used.

As additional baselines, the *size* and the *random* columns indicate clearly more inefficient ranking mechanisms.

To see the effects of a decreasing intrinsic dimensionality, we replace the Hellinger distance by the Euclidean distance. This reduces the intrinsic dimensionality of the scenario as defined by formula 2.20 on page 37 from 9.9 to 2.9. This decrease in intrinsic dimensionality leads to fewer peers being contacted on average in order to retrieve the same amount of NNs as can be observed from table 5.11.

In particular, the decrease of the intrinsic dimensionality leads to noticeably more peers being pruned from search. For *stable*, *mindist*, and *prox*, only 41.0% in contrast to 64.0% of the peers are queried before the algorithm for precise search stops. Furthermore, by comparing both tables it can be observed that the proximity-based ranking *prox* is more affected by an increase in intrinsic dimensionality than the approaches displayed in the columns *stable* and *mindist*. When comparing *prox*

	stable	*mindist*	*prox*	*size*	*random*
60% found	1.2%	1.3%	3.4%	8.5%	28.0%
70% found	1.8%	1.9%	5.0%	12.5%	32.0%
80% found	2.5%	2.6%	7.4%	17.1%	36.4%
90% found	3.9%	3.9%	10.8%	23.4%	40.2%
100% found	7.8%	7.9%	17.2%	33.8%	43.9%
end	41.0%	41.0%	41.0%	41.8%	45.9%

Table 5.11. — Fraction of peers visited [in %] for retrieving a fraction of the 20-NNs in case of RS4MI_{00100} with $m = 8192$ when using the Euclidean distance.

	$\textit{stable}_{\mathrm{L}_2}$	$\textit{mindist}_{\mathrm{L}_2}$	$\textit{stable}_{\mathrm{H}}$	$\textit{mindist}_{\mathrm{H}}$
60% found	1.2%	1.3%	1.5%	1.6%
70% found	1.7%	1.8%	2.2%	2.3%
80% found	2.5%	2.6%	3.2%	3.3%
90% found	3.8%	3.9%	5.0%	5.0%
100% found	7.4%	7.4%	9.7%	9.7%
end	41.0%	41.0%	63.9%	63.9%

Table 5.12. — Fraction of queried peers [in %] for retrieving a fraction of the 20-NNs in case of RS4MI_{00100} with $m = 8192$ averaged over ten runs.

with *stable* or *mindist*, differences in resource selection performance in case of the Euclidean distance are smaller than in case of the Hellinger distance. The fact that also the columns *size* and *random* show a better resource selection performance when using the Euclidean distance confirms that a lower intrinsic dimensionality in general leads to an easier indexing and search scenario.

For assessing the performance differences between *stable* and *mindist* in more detail, we analyze the results averaged over ten runs with the 200 queries each in table 5.12. The numbers confirm that the *stable* approach slightly outperforms the *mindist* approach in case of the Hellinger distance $dist_{\mathrm{H}}$ as well as the Euclidean distance $dist_{\mathrm{L}_2}$.

It is worth analyzing the differences between the *stable* and the *mindist* approach on a conceptual level. If we assume a Voronoi-like partitioning (see section 2.3.1 on pages 25–26) as in case of *stable*, we can determine lower bound distances for objects in clusters $[c_i] \in \{[c_i] \,|\, 1 \leq i \leq m\}$. For database objects in the query cluster, without any additional information, the lower bound distance is zero. The center of the query cluster is determined by $c^* = \arg\min_{c_i \in C} dist(c_i, o)$. Thus, when lower bounding potential objects in any cluster besides the query cluster, that is, clusters $[c_i] \in \{[c_i]\} \backslash [c^*]$, the difference $dist(q, c_i) - dist(q, c^*)$ is non-negative—zero if $dist(q, c_i) = dist(q, c^*)$. Since both $dist(q, c^*)$ and the denominator of two in equation 2.14 of lemma 1 on page 25 are constant when lower bounding potential objects in any cluster besides the query cluster, an ordering of clusters by the lower bound distance stated in lemma 1 yields the same ordering as in case of L_q which is only based on $dist(q, c_i)$. It can thus be concluded that the resource ranking approach from Eisenhardt et al. [2006] outlined in section 3.3.4 on pages 72–73 and denoted as *stable* in this section is equivalent to a ranking based on the lower bound distance given by lemma 1.

With this observation in mind, *stable* refers to a resource ranking based on lower bound distances arising from the Voronoi-like space partitioning. In contrast, a resource ranking based on *mindist* uses lower bound distances derived from the (possibly overlapping) cluster balls. As our measurements indicate, in case of a rather large number of reference objects ($m = 8192$), the lower bound based on the Voronoi-like space partitioning is slightly more effective than the lower bound based on the cluster balls.

Chapter 6.

Applicability of the Approaches

Potential application fields for IF4MI and RS4MI are versatile. In this chapter, we name some research fields where resource description and selection techniques for arbitrary metric spaces and thus RS4MI including HFS and UFS can provide a valuable contribution. We list domains where we see potential for the application of our resource description and selection techniques. In many application fields, resource selection techniques for text retrieval are already used and the application of techniques like ours is possible when extending those applications to other media types besides text.

In general, we see two modes of application—searching for similar feature objects and searching for similar resources, that is, collections of feature objects. The first application mode strictly follows the resource selection task as described in this thesis. Here, a similarity query with a particular query object is issued and matched against a set of resource descriptions. RS4MI including HFS and UFS can be applied. On the other hand, there are applications where a particular collection—represented as a resource description—is matched against other resource descriptions. Here, any particular dissimilarity measure can be applied (see section 2.2). If the dissimilarity measure is a distance metric, IF4MI can for example be used as an access method. These two modes of application are reflected in the following list of application fields.

Centralized access methods. The relatedness of centralized and distributed access methods becomes evident throughout this thesis. A decision of choosing the best subtree in tree-based access methods is for example conceptually similar to the resource selection task. Summaries in distributed IR can correspond to aggregations maintained in the nodes of a tree such as bounding boxes in the case of an R-tree [Guttman, 1984], or more general in the context of MAMs, metric balls in case of an M-tree (see sec-

tion 3.2.1 on pages 47–48). In our case, aggregations maintained by IF4MI such as the R^{in} and R^{out} matrices or parts of them offer promising resource descriptions in case of RS4MI.

Centralized MAMs are used in various applications. Examples are performance prediction in grid environments [Li and Wolters, 2006], text retrieval [Skopal and Moravec, 2005], multimedia and 3D object retrieval [Bustos et al., 2007], similarity search on business process models [Kunze and Weske, 2011], malware detection [Hu et al., 2009], search in biological databases [Xu and Miranker, 2004], or more general, data compression, pattern recognition, machine learning, statistical data analysis, and data mining [Chávez et al., 2001b, *ch.* 2; Zezula et al., 2006, *p.* 3f.].

Expertise retrieval. Expert search [Balog et al., 2009] and expertise retrieval [Balog et al., 2012] use resource description and selection techniques. In expert search, a user is interested in finding human experts in an enterprise for example. Thus, documents a person has (co)authored can be modeled as a resource and finding an expert then results in selecting the most promising resource. In this context, RS4MI can offer possibilities when modeling multimedia user content.

Focused crawling. The idea of focused crawling is initially proposed in Chakrabarti et al. [1999]. Zhang and Nasraoui [2009] suggests a two-step profile-based approach for the focused crawling of social multimedia websites. In a first step, profile pages are identified for generating profile representations which later guide the crawling process. In the second step, a ranking based on the user profiles is applied for determining the order in which the pages with the actual images are crawled. The approach is based on textual information gathered on the pages. Content-based image properties are not considered in Zhang and Nasraoui [2009]. Here, RS4MI can find its application.

In a more structural change, the idea of an approach as proposed in Zhang and Nasraoui [2009] where profiles first have to be generated can be turned into an approach where a service automatically provides summaries of websites or individual pages including image content. A focused crawler can then directly estimate the potential usefulness of a resource for the focused

crawling task before actually visiting the source. This way, crawl efficiency can be improved by preventing the crawler from analyzing too many non-relevant sites or pages. Additional web traffic can thus be avoided.

Personal meta-search. Personal meta-search is a particular application of distributed IR where all the online resources of a person are queried (e-mail accounts, web pages, image collections, etc.). These resources are typically heterogeneous in size, media type, and update frequency and are often stored in different locations [Thomas and Hawking, 2009]. Selective and space efficient summaries can be used in this context to identify relevant resources. Thomas and Hawking [2009] addresses text search whereas RS4MI offers possibilities when content-based search on different media types such as image, audio, or video files is to be supported.

Recommender systems. Space efficient resource descriptions can also be beneficial in the context of recommender systems and social search for example to compute the similarity between different users of social network sites. Similar users can be determined not only based on having the same friends, using the same tags, bookmarking the same media items, etc. [Guy et al., 2010], but also depending on the similarity of media content or more general the data they administer. A user's collection can thus be represented by a RS4MI summary and the task of finding similar users can then be accomplished by comparing the RS4MI resource descriptions, possibly with the help of IF4MI.

Sensor and ad hoc networks. The resource selection techniques presented in this thesis can also be applied in (visual) sensor networks as well as in ad hoc networks. Elahi et al. [2009, *p.* 228] introduces the "*sensor ranking primitive*" for content-based sensor search and emphasizes the necessity of efficient sensor ranking. Lupu et al. [2007] presents an approach for ad hoc information sharing based on mobile devices when people meet at certain events or places. In this context, it is argued that it might not be feasible to transfer complete index data but only summarized information. Both approaches Elahi et al. [2009] and Lupu et al. [2007] are however not designed for search in arbitrary metric spaces. Here, RS4MI can be an option.

Theme identification. Another potential application field is automatic photo collection summarization (see e.g. Obrador et al. [2010]). Theme identification of photo sets for example in case of digital print products such as photo album creation is concerned with the task of finding suitable background themes for a given set of images. A user collection consisting of a set of photos can for example be modeled as a RS4MI summary. Finally, theme descriptions and the description of a user's photo collection can be compared to recommend the best matching theme(s).

Vertical search. Distributed IR techniques are also used for vertical selection in aggregated search [Arguello et al., 2009]. Vertical selection is the task of identifying relevant verticals, that is, focused search services such as image, news, video, or shopping search. A user issuing a textual query "music beatles" might also be interested in music videos and thus video search results or small previews should be integrated into the result presentation of classic web search. In this context, a vertical can be interpreted as a resource and the task of selecting relevant verticals is conceptually similar to resource selection in distributed IR requiring adequate features (i.e. resource descriptions) and corresponding selection mechanisms. Content-based multimedia information can be integrated into vertical selection by applying techniques outlined in this thesis. For a general introduction on aggregated search see Lalmas [2011].

Visual analytics. A main benefit of RS4MI with HFS and UFS is that the resource descriptions and the corresponding resource selection step can be visualized in a meaningful way. This opens doors for visual analytics which is defined in Keim et al. [2008, *p.* 157] as follows: "*Visual analytics combines automated analysis techniques with interactive visualizations for an effective understanding, reasoning and decision making on the basis of very large and complex data sets.*"

To give a tangible example, we will in the following exemplarily outline the concept of our visual analytics interface proposed in Henrich and Blank [2012] which gives additional insights in the resource selection process and which can visually support the resource selection task. The visual interface is outlined in figure 6.1. It has a minimalistic design consisting of a board

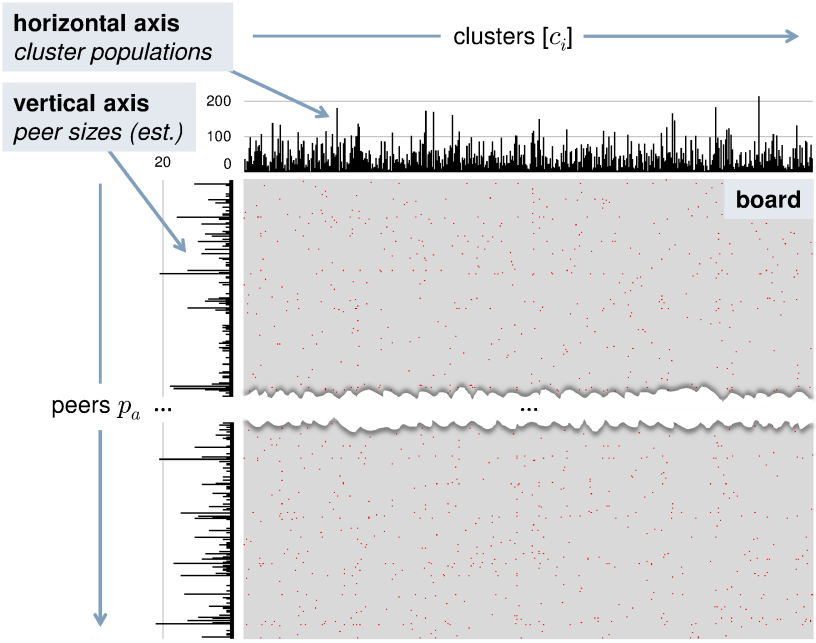


Figure 6.1. — Overview of our visual analytics interface.

and a horizontal and vertical axis. In the following, we assume UFS resource descriptions. However, the interface is by no means restricted to this particular summary type.

The resource descriptions of all participating resources are visualized on the board. To keep it clean without any additional overhead, a single row of pixels on the board is reserved for visualizing a particular resource description. The horizontal axis on top of the interface captures the distribution of the number of populated UFS bins. It shows the aggregated cluster population, that is, the number of summary bins containing at least one image representation aggregated over all resources/peers. On the other hand, the vertical axis on the left captures the distribution

of the number of images of the resources (i.e. the peer sizes). They are estimated here based on the UFS summaries[62].

For UFS, a red pixel on the board represents an occupied summary bin, that is, a summary bin value set to 1. When a summary bin is set to 0, the pixel representing the summary bin remains gray (see e.g. figure 6.1). In case of other RS4MI summary types, graduated pixel colors can for example indicate cluster counts in case of HFS and ball radii in case of RS4MI_{00100}.

Figure 6.2 shows a ranking by peer size where the top part of the board contains by far more red pixels than the bottom part. Also the estimated peer size distribution on the vertical axis indicates the ranking by the number of summary bins set to 1.

If the cluster centers are derived from real data objects which is common in metric space indexing, the centers can be visualized for example by hovering over the horizontal axis (see figure 6.2). Cluster centers can be selected as query images for issuing similarity queries.

Figure 6.3 shows the visualization of the resource ranking step when using the ranking mechanism initially proposed in Eisenhardt et al. [2006] and outlined in section 3.3.4 on pages 72–73 in case of UFS. The pixel order on the board from left to right changes as clusters are rearranged. This can also be noticed when looking at the horizontal axis.

Characteristic properties of the UFS/HFS ranking mechanism are shown on the board in figure 6.3 where it can be perceived that roughly spoken more populated clusters with centers close to the query lead to higher ranking positions. The cluster order is determined by the list L_q as introduced in definition 4 on page 28 and thus the distance from a cluster center to the query object. An example query result is shown in figure 6.4.

It should also be noted that resource descriptions can be visualized in different ways. If images are for example tagged with geographic locations, it is possible to provide an overview of the geographic distribution of a peer's image collection. This can be

[62] For this visualization, we use the images of the MIRFLICKR-25000 collection [Huiskes and Lew, 2008]. CEDD [Chatzichristofis and Boutalis, 2008] features are extracted and compared using the Hellinger distance. Images are again assigned to peers by the Flickr user ID.

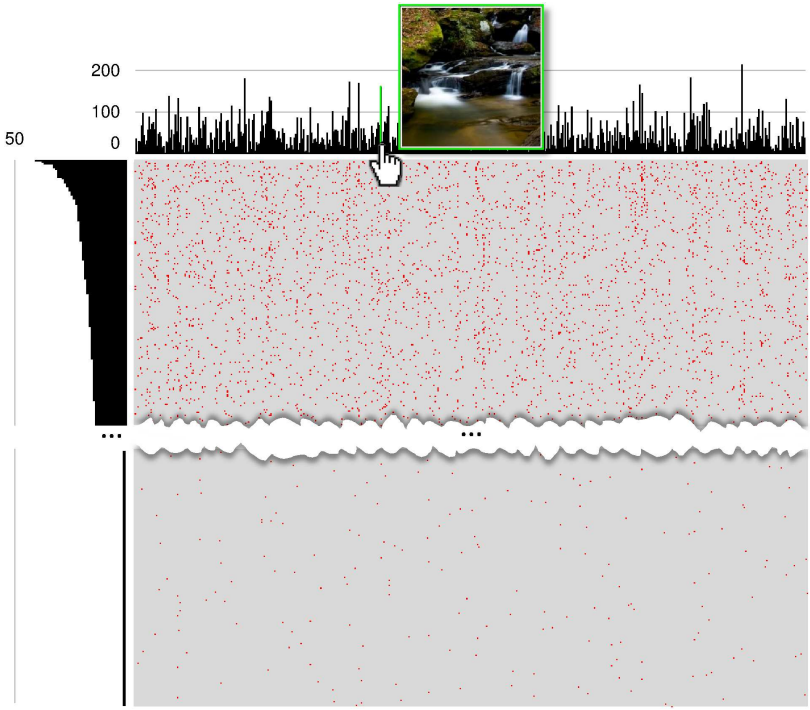


Figure 6.2. — Selecting a center as a query object after ranking the resources by estimated peer size.

achieved by visualizing geographic resource descriptions as briefly addressed in section 5.1.2 on page 115 (see figure 6.5 where UFS summaries are also used for the geographic domain).

In addition, we would like to emphasize that addressing the visual analytics definition given above, our interface allows to focus on certain characteristics of the data collection. It is possible to assess special parts of the data collection such as for example resources with the least similar images which are found at the bottom part of the resource ranking as shown in figure 6.6.

Following the general design paradigm, clicking on a bar in the vertical axis and thus selecting a certain peer triggers a search for similar resources. To do so, resource descriptions themselves are

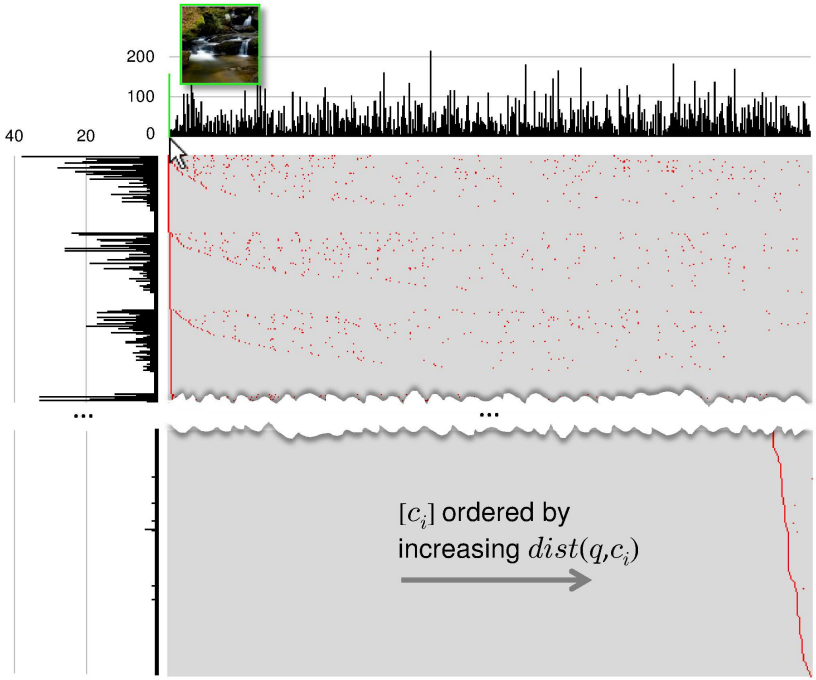


Figure 6.3. — The resource ranking in case of UFS.

perceived as feature objects and compared with the query. This is in contrast to the selection of a certain cluster center from the horizontal axis as a query image. Thus, both application modes briefly introduced at the beginning of this section are supported by the resource selection interface—searching for similar images to a query image and searching for similar resources/collections to a given image collection.

Web service discovery. The metric space model is also assumed when addressing the task of web service discovery. Web service representations including functional and non-functional semantics are compared with a distance metric in Wu et al. [2009]. In scenarios where multiple services are to be described, RS4MI can come into play. P2P systems are frequently proposed as decentralized

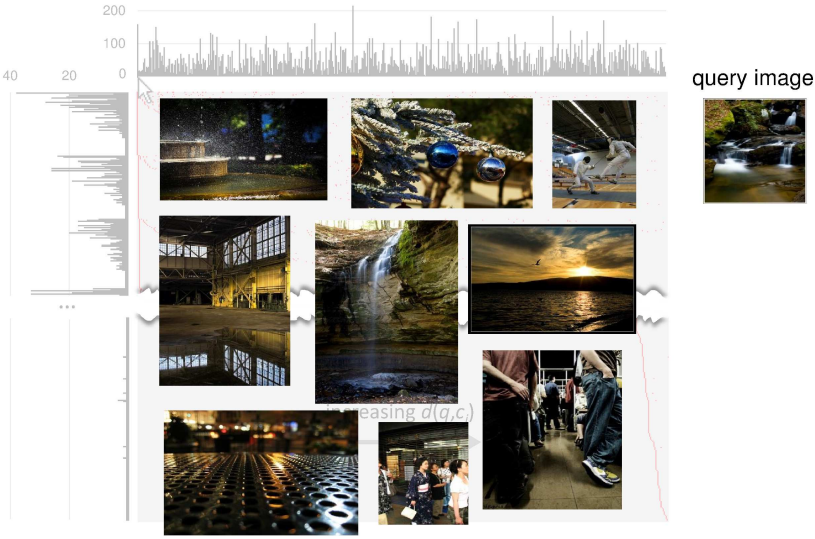


Figure 6.4. — An exemplary query result.

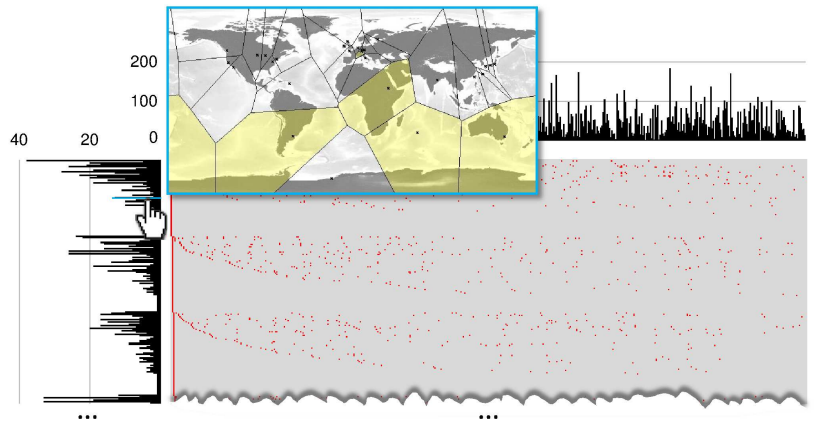


Figure 6.5. — Visualizing a second type of resource descriptions.

infrastructures for web service discovery (see e.g. Banaei-Kashani et al. [2004] and Verma et al. [2005]).

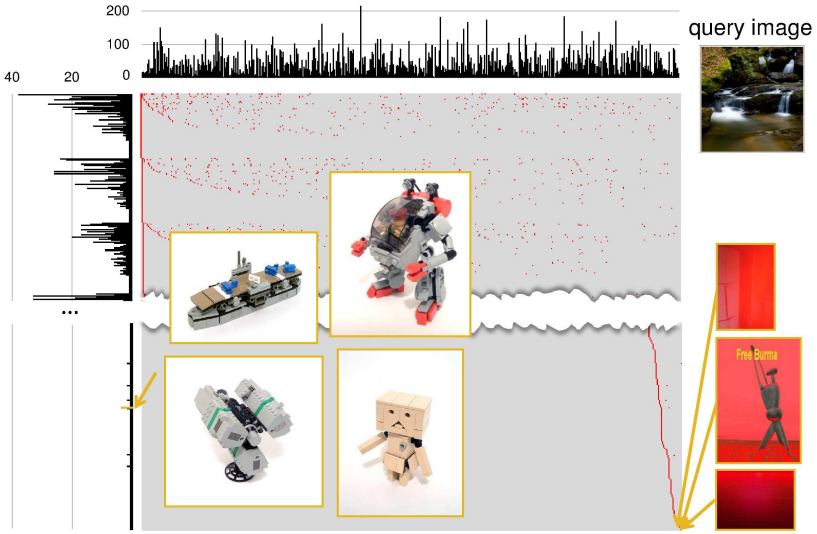


Figure 6.6. — Analyzing the bottom part of the resource ranking.

XML retrieval and blog site search. Distributed text IR research has also influenced techniques for XML retrieval [Larson, 2007] and blog feed search [Elsas et al., 2008]. In blog feed search, a blog feed can be perceived as a single collection and individual blog posts (i.e. feed entries) are interpreted as documents in order to retrieve similar blogs according to a given information need. In passage retrieval such as XML retrieval, different sections, subsections, etc. can be grouped together and be perceived as a resource (for references see Lalmas [2009]). Also here, techniques presented in this thesis can be applied if for example content-based multimedia search is to be supported.

Chapter 7.

Conclusion and Outlook

This thesis addresses the need for efficient search techniques assuming the metric space model. Here, the dissimilarity between two data objects is measured as a distance between them. The distance measure is a distance metric and thus obeys the metric postulates. Arbitrary distance metrics can be applied and thus the metric space model provides a flexible indexing paradigm in times of a growing diversity in data types in different application contexts [Novák, 2008, *p.* 5] and an ever increasing amount of media items in the WWW, in companies, and on private devices.

After a general introduction, this thesis describes the foundations of metric space indexing which provide the theoretical background and the basis for the design and the analysis of MAMs (see chapter 2). The thesis addresses both centralized and distributed MAMs and outlines existing approaches in both fields in a structured way (see chapter 3). Two MAMs are developed in this thesis—IF4MI and RS4MI. IF4MI belongs to the group of centralized MAMs (see chapter 4). On the other hand, RS4MI as a distributed MAM is primarily targeted to search scenarios where data objects are physically distributed (see chapter 5). However, RS4MI is by no means restricted to a certain type of P2P IR system or distributed IR in general. An overview of various application fields for our distributed resource description and selection techniques RS4MI including HFS and UFS is also given in this thesis (see chapter 6). There, we also name application fields for centralized MAMs such as IF4MI.

Addressing the major thesis objectives outlined in section 1.3 on page 10, we can conclude the following. This thesis shows how to improve the resource description and selection technique from Eisenhardt et al. [2006] which enhances earlier work from Müller et al. [2005a]. More fine-grained resource descriptions called HFS and UFS lead to an improved resource selection performance. Increased space requirements

of the resource descriptions are encountered by the use of compression techniques.

Besides these improvements, we devise resource description and selection techniques for precise search and thus the general RS4MI approach is no longer tied to approximate query processing. RS4MI is applicable in arbitrary metric spaces. It can outperform existing approaches which are based on local clustering. Different RS4MI summary types can be used and combined in a single distributed IR system and when appropriate, small peers can be represented exactly, that is, transfer index data in an unsummarized way.

Furthermore, we show how RS4MI can positively influence future research in different application fields even apart from P2P IR and classic distributed IR. As one concrete example, we adapt ideas from RS4MI and develop the centralized MAM entitled IF4MI. If IF4MI is for example applied by the resources as their choice of a local document index structure, information already maintained by IF4MI can directly be used as a resource description without any need for a new computation of an RS4MI summary. As a second example, the resource description and selection techniques are applied in the field of visual analytics for the analysis of large sets of image collections. Here, the applicability is by no means restricted to the image media type.

There are multiple technical improvements to both approaches IF4MI and RS4MI. As for every other MAM, to name only a subset of improvements, batch processing of queries, improved pivot selection strategies, indexing the pivots themselves, and the support of additional query types are some potential areas of future research. These technical improvements are frequently addressed in the field of centralized MAMs and additional requirements can thus be derived from features of alternative MAMs (see e.g. Chávez et al. [2001b], Hetland [2009b], and Zezula et al. [2006]).

More particularly, there is also a variety of future work more closely related to our resource description and selection techniques. One direction of future research can address the idea of index swapping presented in Eisenhardt et al. [2006] and Eisenhardt et al. [2008] where the resources try to sharpen their resource descriptions by transferring index data within the system. More focused resource descriptions can lead to better resource selection performance. This technique reflects the trend towards hybrid indexing scenarios where techniques from local and global indexing schemes are combined.

For the query processing techniques presented in this thesis, several design alternatives are possible. With regard to precise k-NN query processing as an extension of range query processing, we state one particular algorithm. Several variations and alternatives are possible: for its general design, for estimating the search radius, for measuring the amount of overlap between the query and the cluster balls and thus for determining mechanisms on how to best rank the resources (see e.g. Amato [2002, *ch.* 4] and Amato et al. [2003]), for allowing different resource description types to be used in a particular system, to name only a few.

Varying the space partitioning techniques on top of RS4MI is another design alternative. Ideas from PBI including the permutation-based space partitioning can be applied. Algorithms for approximate and precise search can be designed and compared with the approaches developed in this thesis.

An interesting field of future work regarding distributed RS4MI query processing is also the development of an index structure for the resource descriptions to speed-up the resource selection process. It seems natural to extend and adapt IF4MI in this regard.

Multiple research questions also arise with respect to the development of the resource selection interface for visual analytics. It should be analyzed how to best visualize particular resource descriptions. If a huge amount of clusters is applied, using one pixel per cluster and resource might not be possible because of a limited screen resolution. In addition, it is difficult to visualize a huge amount of resources at a time. Thus, aggregation and grouping mechanisms should be investigated which go in hand with the resource selection algorithms.

With the upcoming demand for more flexible indexing paradigms than addressed by traditional SAMs and MAMs [Skopal and Bustos, 2011], it is also interesting to analyze if and how techniques from IF4MI and RS4MI can be helpful for the design of non-metric access methods.

Furthermore, we look forward to the application of IF4MI and RS4MI in the application fields discussed in chapter 6. Therefore, an open source implementation of IF4MI within the widely used inverted file library Apache Lucene[63] can be an interesting option for the dissemination of the IF4MI approach, inherently providing capabilities for the computation of RS4MI resource descriptions.

[63] see *http://lucene.apache.org/*, last visit: 12.10.2014

Appendix A.

Photo License Information

License information of the photos printed in chapter 6 are listed in the following. The photos are taken from the MIRFLICKR 25000 image collection (*http://press.liacs.nl/mirflickr/*, last visit: 23.6.2014). We name the title of the photo (in italics), followed by the MIRFLICKR image number in brackets, Flickr user information (last visit of URLs: 23.6.2014), and creative commons (CC) license information in version 2.0; for details about the licenses see *http://creativecommons.org/licenses/* (last visit: 23.6.2014).

Figure 6.2 and figure 6.3:

Waterfall Odyssey 2008 (no. 14) by Joe Plocki (*https://www.flickr.com/people/turbojoe/*) – CC BY-NC

Figure 6.4:

query image:

Waterfall Odyssey 2008 (no. 14) by Joe Plocki (*https://www.flickr.com/people/turbojoe/*) – CC BY-NC

result images (line by line from top left to bottom right):

Drops (no. 17992) by mm7163 – CC BY-NC

AUGURI Best Wishes (no. 8826) by Federico Soffici – CC BY-NC-SA

Fergal Martin jumps and scores (no. 15083) by Kenneth Barrett (*https://www.flickr.com/people/griangrafanna*) – CC BY-NC

Yellow Steel (no. 7350) by Chris Smart (*https://www.flickr.com/people/sigma*) – CC BY-NC-ND

Tannery Falls II (no. 12472) by James Marvin Phelps (*https://www.flickr.com/people/mandj98*) – CC BY

Home (no. 24095) by Claudia
(*https://www.flickr.com/people/querida79*) – CC BY-NC-SA

Meet me in the middle, (no. 1986) by John S. Ra
(*https://www.flickr.com/people/racreations*) – CC BY-ND

Maiko off the clock (no. 17363) by Chris Gladis
(*https://www.flickr.com/people/MShades*) – CC BY

vidas ajeNas (no. 1569) by Sun Say – CC BY-ND

Figure 6.6:

query image:

Waterfall Odyssey 2008 (no. 14) by Joe Plocki
(*https://www.flickr.com/people/turbojoe/*) – CC BY-NC

bottom left group (line by line from top left to bottom right):

Escort Carrier (no. 14354) by Soren – CC BY-NC

Zuck (no. 6281) by Soren – CC BY-NC

Type 90 Missile Carrier (no. 3950) by Soren – CC BY-NC

Danboard (no. 5357) by Soren – CC BY-NC

bottom right group (from top to bottom):

smeg (no. 18601) by Aliceson
(*https://www.flickr.com/people/aliceson*) – CC BY-NC-ND

Free Burma (no. 13000) by Alex Schraufstetter
(*https://www.flickr.com/people/dyzzy*) – CC BY-NC-SA

Red Light Room (no. 21828) by Quim Gil
(*https://www.flickr.com/people/qgil*) – CC BY-SA

List of Abbreviations

AESA	Approximating and Eliminating Search Algorithm
BoVW	bag of visual words
CAN	Content-Addressable Network
CBIR	content-based image retrieval
CEDD	Color and Edge Directivity Descriptor
CLARA	Clustering LARge Applications
CLARANS	Clustering Large Applications based upon RANdomized Search
CPU	central processing unit
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DHT	distributed hashtable
EMD	earth mover's distance
FAMES	FAst MEdoid Selection
GNAT	Geometric Near-neighbor Access Tree
GPU	graphics processing unit
GROUP	Gossip-based peer-to-peeR cOmmUnity building Protocol
HFS	Highly Fine-grained Summaries
IF4MI	Inverted File for Metric Indexing and search
IR	information retrieval
k-NN	k-nearest neighbor
LAESA	Linear Approximating and Eliminating Search Algorithm
LC	List of Clusters
LM	language model

LSH	locality-sensitive hashing
LSI	latent semantic indexing
MAM	metric access method
MCAN	Metric Content-Addressable Network
MON	Metric Overlay Network
MRoute	Multimedia Routing Index
MSN	Metric Social Network
NN	nearest neighbor
P2P	peer-to-peer
PAM	Partitioning Around Medoids
PBI	permutation-based indexes
PCA	principal component analysis
PDMS	peer data management system
PP	pivot permutation
PREGO	P2P REcommender system based on Gossip Overlays
PtoAM	Ptolemaic access method
RI	routing index
RS4MI	Resource description and Selection for Metric Indexing and search
SAM	spatial access method
SON	Semantic Overlay Network
SRI	Semantic Routing Index
SSS	Sparse Spatial Selection
UFS	Ultra Fine-grained Summaries
WWW	World Wide Web
XML	Extensible Markup Language

List of Symbols

$\mathbb{N}$	the set of natural numbers (p.25)
$\mathbb{R}$	the set of real numbers (p.8)
$\mathcal{M}$	a metric space (p.16)
$\mathcal{V}$	a vector space (p.9)
$\mathbb{H}$	a subspace of a metric space described as a hyper-ring; i.e. a metric shell (p.34)
$\mathbb{Q}$	the query ball (p.9)
$\mathbb{S}$	a subspace of a metric space (p.24)
$\mathbb{U}$	the universe/domain of feature objects (p.8)
A	a set of arbitrary elements (p.21)
B	a set of arbitrary elements (p.21)
C	the set of reference objects c_i with $C = \{c_i \mid i = 1, \ldots, m\}$ (p.26)
K	the result set of a k-NN query (p.9)
L_P	list of peers/resources determining the order in which they are queried (p.132)
L_x	list of cluster IDs sorted by increasing $dist(c_i, x)$ for $c_i \in C$ when assuming a Voronoi-like partitioning; synonym: pivot permutation (p.27)
O	the database as the set of database objects (p.8)
O_a	the database of a resource/peer p_a (p.60)
P	the set of resources/peers p_a with $P = \{p_a \mid a = 1, \ldots, n\}$ (p.60)
R^{in}	a $m \times m$ matrix with $r^{\text{in}}_{i,j}$ radii (p.33)

R^{out}	a $m \times m$ matrix with $r_{i,j}^{\text{out}}$ radii (p.33)
S	a $\delta \times \delta$ similarity matrix with $s_{i,j}$ values (p.18)
a	running variable with $a \in \mathbb{N}^+$; used in different contexts (p.60)
b	running variable with $b \in \mathbb{N}^+$; used in different contexts (p.73)
c	a cluster center; synonyms: reference object, pivot, anchor (p.24)
$[c]$	a cluster with center c arising from Voronoi-like partitioning (p.25)
$[c]_{r^{\text{out}}}$	a metric ball defined by its center c and a (covering) radius r^{out} (p.30)
$\check{d}_i$	a lower bound distance of the distance $dist(q,o)$ calculated as $\check{d}_i = \lvert dist(q,c_i) - dist(c_i,o)\rvert$ with $1 \leq i \leq m$ (p.47)
$\hat{d}_i$	an upper bound distance of the distance $dist(q,o)$ calculated as $\hat{d}_i = dist(q,c_i) + dist(c_i,o)$ with $1 \leq i \leq m$ (p.133)
$d^{\max}$	the maximum possible distance value between two feature objects if the distance metric has an upper bound (p.135)
e	the number of "words" by which a database or query object is represented in case of Sznajder et al. [2008] (p.56)
i	running variable with $i \in \mathbb{N}^+$; used in different contexts (p.18)
j	running variable with $j \in \mathbb{N}^+$; used in different contexts (p.18)
k	used in three different contexts: (1) the number of NNs to be retrieved in a k-NN query, (2) the desired number of clusters in k-means or k-medoids clustering, (3) the number of dimensions of a k-d-tree; the meaning becomes clear from the context where k is used (p.9)

l	the number of reference objects determining a cluster in permutation-based partitioning with $l \in \mathbb{N}^+$ and $1 \leq l \leq m$ (p.28)
m	the number of reference objects; sometimes indices such as in case of m_{if} or m_{pm} are used to distinguish the application context where reference objects are applied (p.26)
m'	the number of reference objects; usually $m' \ll m$ (p.52)
n	the number of peers/resources (p.106)
n_{pip}	the number of peers/resources queried in parallel (p.132)
o	a database object $o \in O$ (p.9)
$\vec{o}$	a database object represented as a δ-dimensional feature vector; $\vec{o}[i]$ denotes a vector component i with $1 \leq i \leq \delta$ (p.18)
p_a	a peer/resource $p_a \in P$ (p.60)
q	a query object $q \in \mathbb{U}$ (p.8)
$\vec{q}$	a query object represented as a δ-dimensional feature vector; $\vec{q}[i]$ denotes a vector component i with $1 \leq i \leq \delta$ (p.18)
r	the search radius $r \in \mathbb{R}^+$ (p.9)
$r_{i,j}^{\text{in}}$	the inner radius of a metric shell ($r_{i,j}^{\text{in}} \in \mathbb{R}_0^+$); a single index i in case of r_i^{in} denotes the ID of the shell center, an index pair i,j denotes the minimum distance of database objects from cluster $[c_i]$ to center c_j $(1 \leq i, j \leq m)$ (p.30)
$r_{i,j}^{\text{out}}$	the (covering) radius of a metric ball or the outer radius of a metric shell ($r_{i,j}^{\text{out}} \in \mathbb{R}_0^+$); a single index i in case of r_i^{out} denotes the ID of the ball or shell center, an index pair i,j denotes the maximum distance of database objects from cluster $[c_i]$ to center c_j $(1 \leq i, j \leq m)$ (p.30)

$s_{i,j}$	a similarity value capturing the similarity between two feature vector dimensions i and j ($1 \leq i, j \leq \delta$) (p.18)
t	a (term) weight in a posting of an inverted file; e.g. determined by $L_x(c_i)$ (p.54)
w	a feature object $w \in \mathbb{U}$ (p.17)
x	a feature object $x \in \mathbb{U}$ (p.9)
$\vec{x}$	a feature object represented as a δ-dimensional feature vector; $\vec{x}[i]$ denotes a vector component i with $1 \leq i \leq \delta$ (p.9)
y	a feature object $y \in \mathbb{U}$ (p.9)
$\vec{y}$	a feature object represented as a δ-dimensional feature vector; $\vec{y}[i]$ denotes a vector component i with $1 \leq i \leq \delta$ (p.9)
z	a feature object $z \in \mathbb{U}$ (p.16)
$closest(x, C)$	a function determining the ID(s) of the cluster center(s) $c_i \in C$ with minimum distance $dist(c_i, x)$ to feature object x (p.27)
$dist(x, y)$	a distance measure between two feature objects x and y (p.8)
$dist'(\vec{x}, \vec{y})$	a distance measure in a mapped vector space (p.50)
$f(r, r_i^{\text{out}}, dist(q, c_i))$	a function for estimating the number of relevant database objects in a metric ball in case of a range query based on the search radius r, the radius of the metric ball r_i^{out}, and the distance $dist(q, c_i)$ between the query object and the ball center as discussed in Berretti et al. [2002a, *p.* 199] (p.136)
$knn(q, k)$	a k-NN query which determines the k closest database objects to the query object q (p.9)
$L_x(c_i)$	a function determining the position/rank $t \in \mathbb{N}_0^+$ of the index i in L_x; if i is not contained in the list, zero is returned (p.54)

$map(x)$	a function mapping x from a metric space into a vector space, the representation in the mapped vector space is denoted as $\vec{x}$ (p.50)
$mindist(x, [c]_{r^{\text{out}}})$	the minimum distance between a feature object x and a metric ball $[c]_{r^{\text{out}}}$ as defined in Ciaccia et al. [1997, *p.* 429] (p.134)
$oid(o)$	a function determining the object ID of a database object $o \in O$ (p.79)
$prox([c_i]_{r_i^{\text{out}}}, [c_j]_{r_j^{\text{out}}})$	the proximity of two metric balls as defined in Amato [2002, *p.* 72] and Amato et al. [2003, *p.* 197] (p.135)
$range(q, r)$	a range query which determines all database objects $o \in O$ with $dist(q, o) \leq r$ (p.9)
$sim(x, y)$	a similarity measure between two feature objects x and y (p.8)
α	parameter of the Minkowski distance with $\alpha \in \mathbb{N}^+$ (p.18)
$\beta_{\text{b}}^{\text{M}}$	the block size of an M-tree or PM-tree (p.67)
β_{d}	the storage space needed for a single distance value (p.91)
β_{idx}	the storage space needed for a particular index; $idx \in \{\text{IF}, \text{MI}_{\text{mod}}, \text{M-tree}\}$ (p.90)
β_{o}	the storage space needed for a single database object (p.91)
$\beta_{\text{rd}}^{\text{avg}}$	the average space requirement of a set of resource descriptions (p.105)
$\tilde{\beta}_{\text{rd}}^{\text{avg}}$	the desired average space requirement of a set of resource descriptions (p.129)
μ	the arithmetic mean (p.37)
σ^2	the variance (p.37)
δ	the representational dimension(ality) of the feature space (p.17)

θ	a distance threshold for computing resource descriptions with the help of an M-tree as proposed in Berretti et al. [2002a,b, 2004] (p.67)
ι	separation constant of the Metric iDistance (see Novák [2008, *p.* 59f.]) (p.51)
ξ	a distance threshold (p.29)
ρ	the intrinsic dimension(ality) of a dataset as defined in Chávez et al. [2001b, *p.* 303] (p.37)
τ	the number of rows of the grid summaries for geographic IR with $\tau \in \mathbb{N}^+$ (p.109)
ψ	a distance threshold (p.18)

List of Figures

1.1. Using the image similarity search of Google[64] on 16.5.2013 (*http://images.google.com/*). Images in an initial search result can be used as query images for the search of visually similar images. 3
1.2. Our scenario of a distributed RS4MI system also using IF4MI for the local query processing (resources shaded gray). 11
1.3. Visualization of the thesis structure additionally indicating where the main thesis objectives are particularly addressed. 14

2.1. Hyperplane partitioning schemes. For visualization purposes—here and in the remainder of this thesis—a two-dimensional Euclidean space is assumed and the cluster borders are drawn as solid black lines. Subspaces $\mathbb{S}_i$ ($i \in \mathbb{N}^+$) are in some cases of generalized hyperplane partitioning denoted $[c_i]$ emphasizing the use of reference objects c_i as cluster centers. 25
2.2. Visualization of the lower-bound distance $(\mathit{dist}(q, c_1) - \mathit{dist}(q, c_2))/2$ on $\mathit{dist}(q, o)$ in case of generalized hyperplane partitioning (adapted from Hjaltason and Samet [2003a, *p.* 540]). 26
2.3. Outline of the permutation-based space partitioning with $m = 4$ and $l = 2$ (adapted from Novák and Batko [2009]). Cluster $[c_{1,4}]$ and cluster $[c_{4,1}]$ do not exist. 28
2.4. Parameterized generalized hyperplane partitioning with $\xi = -\mathit{dist}(c_1, c_2)/3$; adapted from Lokoč and Skopal [2010, *p.* 131]. 29
2.5. Ball partitioning schemes; figures adapted from Zezula et al. [2006, *p.* 20] 31
2.6. Applying the pruning rule based on the double-pivot distance constraint in case of a range query $\mathit{range}(q, r)$ where cluster $[c_1]$ can be successfully pruned. 32
2.7. Successful application of the pruning rule based on the range-pivot distance constraint in case of a range query $\mathit{range}(q, r)$. Cluster $[c_1]$ can be pruned because $\mathit{dist}(q, c_1) - r > r_1^{\text{out}}$ and thus the query ball does not intersect the ball with radius r_1^{out} around c_1. 33

2.8. Successful application of the pruning rule based on the range-pivot distance constraint in case of a range query *range*(q, r). Cluster $[c_1]$ can be pruned because *dist*$(q, c_1) + r < r_1^{\text{in}}$ and thus the query ball lies fully inside the ball with radius r_1^{in} around c_1 where—in cluster $[c_1]$—no database objects are lying. 34

2.9. A pruning example showing the usefulness of information stored in the matrices R^{in} and R^{out}; $r_{1,2}^{\text{in}}$ and $r_{1,2}^{\text{out}}$ are used for restricting the region of possible database objects in cluster $[c_1]$ to the two dark gray shaded intersection areas of $\mathbb{H}_{1,1}$ and $\mathbb{H}_{1,2}$. Since the query ball $\mathbb{Q}$ does not intersect any of these regions, cluster $[c_1]$ does not contain any database objects relevant to the query. 34

2.10. Applying the object-pivot distance constraint (visualization adapted from Hetland [2009b, *p.* 207]): On the one hand, *dist*$(q, c) -$ *dist*$(c, o) > r$ holds for database objects inside the inner white ball around center c which has radius *dist*$(q, c) - r$. On the other hand, *dist*(c, o) - *dist*$(q, c) > r$ holds for database objects outside the outer ball around c with radius *dist*$(q, c) + r$. Thus, $|$*dist*$(q, c) -$ *dist*$(c, o)| \leq r$ holds for database objects which lie inside the gray shaded shell (including the borders) containing the query ball $\mathbb{Q}$. These objects cannot be pruned from search based on c. By using additional reference objects, the region of possible database objects within the search radius can be further restricted through intersections of multiple shells. 36

2.11. Two exemplary distance histograms (figure adapted from Chávez et al. [2001b, *p.* 302]). According to the definition given in formula 2.20 on page 37, the histogram on the right reflects a higher intrinsic dimensionality than the histogram on the left. 38

3.1. Sketch of a basic k-means clustering algorithm adapted from Han and Kamber [2006, *p.* 403]. 43

3.2. Mapping technique of the Metric iDistance with $m = 4$, under the assumption of a normalized metric space and thus $\iota = 1$; adapted from Novák [2008, *p.* 60]. 51

3.3. Search example for Amato and Savino [2008] and Gennaro et al. [2010]. Note that database objects are included in the postings instead of object IDs for visualization purposes. 55

3.4. Different types of distributed MAMs; classes GG, LG, and LL adopted from Gil-Costa et al. [2009, *sect.* 4.2]. 58

3.5. Visualization of distributed metric space indexing schemes. Peers $p_a\,(1 \leq a \leq |P|)$ maintain documents $o_{j,a}\,(1 \leq j \leq |O_a|)$. Clusters are visualized as circles indicating metric balls, filled black circles denote the particular data region. The notation $\rightarrow p_a$ in combination with the same background color denotes the "responsibility" of a peer p_a. 60
3.6. Different P2P network architectures according to Lu [2007]; visualization inspired by Lu [2007, figure 2.1 on *p.* 7]. Squares □ denote peers; those with large black borders ◘ are super-peers. Circles ○ visualize resource descriptions. The same fill colors indicate a similar "thematic" focus of peers and/or resource descriptions. 63

4.1. Conceptual outline of the IF4MI structure. Database objects are included in the postings instead of object IDs for visualization purposes. 78
4.2. An algorithm for k-NN queries with IF4MI. 79
4.3. Necessary distance computations and pruned feature objects (via object pruning and cluster pruning) for IF($m_{\mathsf{if}}, m'_{\mathsf{if}}$) [in %]. IF($[m_{\mathsf{if}}], m'_{\mathsf{if}}$)—see the rightmost bar in each group—is a parameter setting with no cluster pruning where all posting lists are processed to prune objects based on pivot filtering. 83
4.4. Number of postings per posting list for IF(1024,*) and $|O| = 100\,000$ postings in total. The placeholder $*$ for m'_{if} indicates that these measurements are independent of m'_{if}. 85
4.5. Fraction of distance computations compared to a sequential scan when varying m_{if} and m'_{if} for IF($m_{\mathsf{if}}, m'_{\mathsf{if}}$). 86
4.6. Pruned and unpruned database objects for IF(m_{if},40). 87
4.7. Distance computations compared to a sequential scan for PM(m_{pm}, m') with a block size of 4096 bytes and for IF(m_{if}, m'). 89
4.8. Influence of different block sizes $\beta_{\mathsf{b}}^{\mathsf{M}}$ on the number of distance computations in relation to a sequential scan for the tree-based approaches. 90
4.9. Pruned and unpruned database objects for IF(347,40) and MI$_{\mathsf{mod}}$(40,3) on the database with $100\,000$ objects when performing 20-NN queries. 94
4.10. Necessary pivot filtering steps per successfully pruned database object for different pivot filtering heuristics in case of IF(1024,1024). 96

5.1. Distribution of peer sizes, that is, the number of database objects maintained per peer. 103

5.2. Fraction of queried peers to retrieve the top-20 image feature objects, i.e. the 20-NNs. 104
5.3. Box plot of summary sizes (zipped). 106
5.4. Fraction of queried peers for retrieving the top-20 images with UFS(e)(256) vs. UFS(e). 108
5.5. Visualizing summary creation for geographic IR. Four images are geotagged in this example, indicated as **x**. 109
5.6. Number of images per peer for the Geoflickr collection. 111
5.7. Geographic distribution of image locations for Geoflickr. 111
5.8. Resource selection performance of HFS, UFS, BB and GRID. . 113
5.9. Selected strategies for choosing the reference points for UFSe when analyzing 20 runs with the 200 queries each. 115
5.10. M-tree-based local clustering (left) with special treatment of single node trees (right). 119
5.11. Results of RS4MI for summaries with space complexity $\mathcal{O}(m)$ (left) and $\mathcal{O}(m^2)$ (right). 127
5.12. A conceptual algorithm for precise RS4MI k-NN queries. 132

6.1. Overview of our visual analytics interface. 145
6.2. Selecting a center as a query object after ranking the resources by estimated peer size. 147
6.3. The resource ranking in case of UFS. 148
6.4. An exemplary query result. 149
6.5. Visualizing a second type of resource descriptions. 149
6.6. Analyzing the bottom part of the resource ranking. 150

List of Tables

4.1. Number of pruned clusters of IF4MI for different values of m_{if}. . 87
4.2. Measured memory requirements of IF4MI for a database with $|O| = 100\,000$ database objects. 91
4.3. Memory requirements for M-tree node entries. 92
4.4. Comparing the space partitioning of IF4MI and the M-Index on a database with $100\,000$ objects in case of 20-NN queries. . . . 93
4.5. Skipped postings for IF(m_{if},*) in case of textual filter queries. . 98

5.1. Average summary sizes in bytes (zipped). 105
5.2. Resource selection performance and average summary sizes for the HFS, UFS, BB, and GRID approaches. 113
5.3. Statistics of the *result size* and the number of *peers* administering result documents for the 200 range queries with a search radius of $r = 0.5$. 117
5.4. Results for local k-medoids clustering with an automatic determination of the number of clusters k. 123
5.5. Results for local k-medoids clustering with all peers using the same global k. 124
5.6. Average summary sizes for RS4MI$_{\text{xxx11}}$ with $m = 64$. 129
5.7. Results when solely applying pivot filtering. 129
5.8. Results for hybrid RS4MI summaries. Table cells show the fraction of queried peers (top) and the average summary size (bottom). The number of peers applying pivot filtering is given in brackets ($10\,601$ peers in total). 130
5.9. Comparing different approaches with results averaged over ten runs with the 200 queries each. 131
5.10. Fraction of peers visited [in %] for retrieving a fraction of the 20-NNs in case of RS4MI$_{00100}$ with $m = 8192$ when using the Hellinger distance. 137
5.11. Fraction of peers visited [in %] for retrieving a fraction of the 20-NNs in case of RS4MI$_{00100}$ with $m = 8192$ when using the Euclidean distance. 138

5.12. Fraction of queried peers [in $\%$] for retrieving a fraction of the 20-NNs in case of RS4MI$_{00100}$ with $m = 8192$ averaged over ten runs. 138

References

Aberer, K., Klemm, F., Luu, T., Podnar, I., and Rajman, M. [2005]. *Building a peer-to-peer full-text Web search engine with highly discriminative keys*. Technical Report LSIR-REPORT-2005-011. Lausanne, Switzerland: Swiss Federal Institute of Technology (EPFL) (cit. on pp. 61 sq.).

Amato, G. [2002]. Approximate similarity search in metric spaces. PhD thesis. Computer Science Department, University of Dortmund, Germany (cit. on pp. 135 sq., 153, 163).

Amato, G., Rabitti, F., Savino, P., and Zezula, P. [Apr. 2003]. Region Proximity in Metric Spaces and Its Use for Approximate Similarity Search. *ACM Transactions on Information Systems*, vol. 21, no. 2, pp. 192–227 (cit. on pp. 135, 153, 163).

Amato, G. and Savino, P. [2008]. Approximate Similarity Search in Metric Spaces using Inverted Files. In: *Proc. of the 3rd Intl. Conf. on Scalable Information Systems*. Vico Equense, Italy: ICST, 28:1–28:10 (cit. on pp. 27, 39, 54 sqq.).

Arantes, A. S., Vieira, M. R., Traina, A. J., and Traina Jr., C. [2003]. The Fractal Dimension Making Similarity Queries More Efficient. In: *Proc. of the 2nd SIGKDD Workshop on Fractals, Power Laws and Other Next Generation Data Mining Tools*. Washington, DC, USA: ACM, pp. 12–17 (cit. on p. 37).

Arguello, J., Diaz, F., Callan, J., and Crespo, J.-F. [2009]. Sources of Evidence for Vertical Selection. In: *Proc. of the 32nd Intl. SIGIR Conf. on Research and Development in Information Retrieval*. Boston, MA, USA: ACM, pp. 315–322 (cit. on p. 144).

Aspnes, J. and Shah, G. [Nov. 2007]. Skip Graphs. *ACM Transactions on Algorithms*, vol. 3, no. 4, 37:1–37:25 (cit. on p. 59).

Athitsos, V., Alon, J., Sclaroff, S., and Kollios, G. [Jan. 2008]. BoostMap: An Embedding Method for Efficient Nearest Neighbor Retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 1, pp. 89–104 (cit. on p. 51).

Balog, K., Soboroff, I., Thomas, P., Bailey, P., Craswell, N., and de Vries, A. [2009]. Overview of the TREC 2008 Enterprise Track. In: *The 17th Text Retrieval Conf. Proceedings (TREC 2008)*. Gaithersburg, MD, USA: NIST (cit. on p. 142).

Balog, K., Fang, Y., de Rijke, M., Serdyukov, P., and Si, L. [July 2012]. Expertise Retrieval. *Foundations and Trends in Information Retrieval*, vol. 6, no. 2-3, pp. 127–256 (cit. on p. 142).

Banaei-Kashani, F., Chen, C.-C., and Shahabi, C. [2004]. WSPDS: Web Services Peer-to-Peer Discovery Service. In: *Proc. of the Intl. Conf. on Internet Computing*. Las Vegas, NV, USA: CSREA Press, pp. 733–743 (cit. on p. 149).

Baraglia, R., Dazzi, P., Mordacchini, M., Ricci, L., and Alessi, L. [2011]. GROUP: A Gossip Based Building Community Protocol. In: *Proc. of the 11th Intl. Conf. on Smart Spaces and Next Generation Wired/Wireless Networking*. St. Petersburg, Russia: Springer LNCS 6869, pp. 496–507 (cit. on p. 70).

Baraglia, R., Mordacchini, M., Dazzi, P., and Ricci, L. [2010]. A P2P REcommender System based on Gossip Overlays (PREGO). In: *Proc. of the 10th Intl. Conf. on Computer and Information Technology*. Bradford, West Yorkshire, UK: IEEE, pp. 83–90 (cit. on p. 70).

Barioni, M. C. N., Kaster, D. d., Razente, H. L., Traina, A. J. M., and Traina Jr., C. [2011]. Querying Multimedia Data by Similarity in Relational DBMS. In: *Advanced Database Query Systems: Techniques, Applications and Technologies*. Ed. by L. Yan and Z. Ma. Hershey, PA, USA: IGI Global. Chap. 14 (cit. on p. 22).

Barioni, M. C. N., Razente, H. L., Traina, A. J. M., and Traina Jr., C. [Mar. 2008]. Accelerating k-medoid-based algorithms through metric access methods. *Journal of Systems and Software*, vol. 81, no. 3, pp. 343–355 (cit. on p. 41).

Barrientos, R. J., Gómez, J. I., Tenllado, C., Matias, M. P., and Marin, M. [2012]. Range Query Processing in a Multi-GPU Environment. In: *Proc. of the 10th Intl. Symp. on Parallel and Distributed Processing with Applications*. Leganés, Spain: IEEE, pp. 419–426 (cit. on p. 59).

Bayer, R. and McCreight, E. M. [1972]. Organization and Maintenance of Large Ordered Indices. *Acta Informatica*, vol. 1, no. 3, pp. 173–189 (cit. on pp. 46 sq.).

Beckmann, N., Kriegel, H.-P., Schneider, R., and Seeger, B. [1990]. The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. In: *Proc. of the Intl. SIGMOD Conf. on Management of Data*. Atlantic City, NJ, USA: ACM, pp. 322–331 (cit. on p. 46).

Beecks, C. [2013]. Distance-based Similarity Models for Content-based Multimedia Retrieval. PhD thesis. Fakultät für Mathematik, Informatik und Naturwissenschaften, RWTH Aachen University (cit. on p. 17).

Beecks, C., Kirchhoff, S., and Seidl, T. [2013]. Signature Matching Distance for Content-based Image Retrieval. In: *Proc. of the Intl. Conf. on Multimedia Retrieval*. Dallas, TX, USA: ACM, pp. 41–48 (cit. on p. 17).

Begelman, G., Keller, P., and Smadja, F. [2006]. Automated Tag Clustering: Improving search and exploration in the tag space. In: *Proc. of the Workshop on Collaborative Web Tagging*. Edinburgh, UK: *http://www.ra.ethz.ch/cdstore/www2006/www.rawsugar.com/www2006/20.pdf* (last visit: 1.10.2014) (cit. on p. 4).

Belkin, N. J., Kantor, P., Fox, E. A., and Shaw, J. A. [1995]. Combining the Evidence of Multiple Query Representations for Information Retrieval. *Information Processing and Management*, vol. 31 (3), pp. 431–448 (cit. on p. 5).

Bender, M., Michel, S., Triantafillou, P., Weikum, G., and Zimmer, C. [2005a]. MINERVA: Collaborative P2P Search. In: *Proc. of the 31st Intl. Conf. on Very Large Data Bases*. Trondheim, Norway: VLDB Endowment, pp. 1263–1266 (cit. on p. 68).

Bender, M., Michel, S., Weikum, G., and Zimmer, C. [2005b]. The MINERVA Project: Database Selection in the Context of P2P Search. In: *11. Fachtagung Datenbanksysteme für Business, Technologie und Web*. Karlsruhe, Germany: GI, pp. 125–144 (cit. on pp. 7, 10, 68).

Bentley, J. L. [Sept. 1975]. Multidimensional Binary Search Trees Used for Associative Searching. *Communications of the ACM*, vol. 18 (9), pp. 509–517 (cit. on p. 115).

Berretti, S., Del Bimbo, A., and Pala, P. [2002a]. Extraction of Resource Descriptors for Distributed Content Based Image Retrieval. In: *Proc. of the 16th Intl. Conf. on Pattern Recognition*. Vol. 3. Quebec, Canada: IEEE, pp. 995–998 (cit. on pp. 42, 44, 47 sq., 67, 75 sq., 116 sq., 162, 164).

Berretti, S., Del Bimbo, A., and Pala, P. [2002b]. Using indexing structures for resource descriptors extraction from distributed image repositories. In: *Proc. of the Intl. Conf. on Multimedia and Expo*. Vol. 2. Lausanne, Switzerland: IEEE, pp. 197–200 (cit. on pp. 42, 44, 47 sq., 67, 75 sq., 116 sq., 136, 164).

Berretti, S., Del Bimbo, A., and Pala, P. [Dec. 2004]. Merging Results for Distributed Content Based Image Retrieval. *Multimedia Tools and Applications*, vol. 24 (3), pp. 215–232 (cit. on pp. 6, 42, 44, 47 sq., 67, 75 sq., 116 sq., 164).

Berretti, S., Del Bimbo, A., and Pala, P. [2007]. Graph Edit Distance for Active Graph Matching in Content Based Retrieval Applications. *The Open Artificial Intelligence Journal*, vol. 1, pp. 1–11 (cit. on p. 20).

Bille, P. [June 2005]. A Survey on Tree Edit Distance and Related Problems. *Theoretical Computer Science*, vol. 337 (1-3), pp. 217–239 (cit. on p. 19).

Blank, D., El Allali, S., Müller, W., and Henrich, A. [2007]. Sample-based Creation of Peer Summaries for Efficient Similarity Search in Scalable Peer-to-Peer Networks. In: *Proc. of the Intl. SIGMM Workshop on*

Multimedia Information Retrieval. Augsburg, Germany: ACM, pp. 143–152 (cit. on pp. 73, 101 sq., 125).

Blank, D. and Henrich, A. [2009]. Summarizing Georeferenced Photo Collections for Image Retrieval in P2P Networks. In: *Proc. of the Workshop on Geographic Information on the Internet*. Toulouse, France: *http://www.uni-bamberg.de/fileadmin/uni/fakultaeten/wiai_lehrstuehle/medieninformatik/Dateien/Publikationen/2009/blank2009summarizing.pdf* (last visit: 27.9.2014) (cit. on pp. 101, 108).

Blank, D. and Henrich, A. [2010a]. Binary Histograms for Resource Selection in Peer-to-Peer Media Retrieval. In: *Proc. of LWA Workshop – Lernen, Wissen, Adaptivität*. Kassel, Germany: *http://www.kde.cs.uni-kassel.de/conf/lwa10/papers/ir4.pdf* (last visit: 27.9.2014), pp. 183–190 (cit. on pp. 73, 101 sq.).

Blank, D. and Henrich, A. [2010b]. Description and Selection of Media Archives for Geographic Nearest Neighbor Queries in P2P Networks. In: *Proc. of the Information Access for Personal Media Archives Workshop*. Milton Keynes, UK: *http://doras.dcu.ie/15373/* (last visit: 27.9.2014), pp. 22–29 (cit. on pp. 101, 106, 108, 110, 112, 115).

Blank, D. and Henrich, A. [2012a]. 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*. Ed. by L. Díaz, C. Granell, and J. Huerta. Hershey, PA, USA: IGI global (cit. on pp. 101, 108, 110, 112, 115).

Blank, D. and Henrich, A. [2012b]. Inverted File-based Indexing for Efficient Multimedia Information Retrieval in Metric Spaces. In: *Proc. of the 27th Annual Symp. on Applied Computing*. Riva del Garda, Italy: ACM, pp. 900–905 (cit. on p. 77).

Blank, D. and Henrich, A. [2013a]. Inverted File-Based General Metric Space Indexing for Quality-Aware Similarity Search in Information Retrieval. In: *Quality Issues in the Management of Web Information*. Ed. by G. Pasi, G. Bordogna, and L. C. Jain. Vol. 50. Intelligent Systems Reference Library. Springer Berlin Heidelberg. Chap. 2, pp. 5–34 (cit. on pp. 47, 77).

Blank, D. and Henrich, A. [2013b]. Resource Description and Selection for Range Query Processing in General Metric Spaces. In: *15. Fachtagung Datenbanksysteme für Business, Technologie und Web*. Magdeburg, Germany: GI, pp. 93–112 (cit. on pp. 73, 101).

Bloom, B. H. [1970]. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Communications of the ACM*, vol. 13 (7), pp. 422–426 (cit. on p. 7).

Bockting, S. and Hiemstra, D. [2009]. Collection Selection with Highly Discriminative Keys. In: *Proc. of the 7th Intl. Workshop on Large-Scale*

Distributed Systems for Information Retrieval. Boston, MA, USA: *http://lsdsir09.isti.cnr.it/lsdsir09-1.pdf* (last visit: 27.9.2014) (cit. on pp. 4, 7).

Boldi, P. and Vigna, S. [2005]. Compressed Perfect Embedded Skip Lists for Quick Inverted-Index Lookups. In: *Proc. of the 12th Intl. Conf. on String Processing and Information Retrieval*. Buenos Aires, Argentina: Springer LNCS 3772, pp. 25–28 (cit. on p. 99).

Bolettieri, P., Esuli, A., Falchi, F., Lucchese, C., Perego, R., Piccioli, T., and Rabitti, F. [2009]. CoPhIR: a Test Collection for Content-Based Image Retrieval. CoRR, abs/0905.4627v2, *http://arxiv.org/abs/0905.4627v2* (last visit: 27.9.2014) (cit. on pp. 3, 81).

Bratsberg, S. E. and Hetland, M. L. [Sept. 2012]. Dynamic optimization of queries in pivot-based indexing. *Multimedia Tools and Applications*, vol. 60 (2), pp. 261–275 (cit. on p. 39).

Brin, S. [1995]. Near Neighbor Search in Large Metric Spaces. In: *Proc. of the 21st Intl. Conf on Very Large Data Bases*. Zurich, Switzerland: Morgan Kaufmann, pp. 574–584 (cit. on pp. 26, 48).

Brisaboa, N., Pedreira, O., Seco, D., Solar, R., and Uribe, R. [2008]. Clustering-Based Similarity Search in Metric Spaces with Sparse Spatial Centers. In: *Proc. of the 34th Conf. on Current Trends in Theory and Practice of Computer Science*. Nový Smokovec, Slovakia: Springer LNCS 4910, pp. 186–197 (cit. on p. 49).

Bryant, V. [1985]. *Metric Spaces: Iteration and Application*. Cambridge University Press (cit. on p. 16).

Bustos, B., Keim, D., Saupe, D., and Schreck, T. [July 2007]. Content-Based 3D Object Retrieval. *IEEE Computer Graphics and Applications*, vol. 27 (4), pp. 22–27 (cit. on pp. 2, 142).

Büttcher, S., Clarke, C. L. A., and Cormack, G. V. [2010]. *Information Retrieval: Implementing and Evaluating Search Engines*. MIT Press (cit. on pp. 77, 97).

Callan, J. [2000]. Distributed Information Retrieval. In: *Advances in Information Retrieval*. Ed. by W. B. Croft. Kluwer Academic Publishers, pp. 127–150 (cit. on p. 6).

Callan, J. and Connell, M. [Apr. 2001]. Query-based sampling of text databases. *ACM Transactions on Information Systems*, vol. 19 (2), pp. 97–130 (cit. on p. 6).

Celebi, M. E., Kingravi, H. A., and Vela, P. A. [Jan. 2013]. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications*, vol. 40 (1), pp. 200–210 (cit. on p. 43).

Celik, C. [2006]. New Approaches to Similarity Searching in Metric Spaces. PhD thesis. University of Maryland, College Park, MD, USA (cit. on pp. 39 sq., 47, 78, 81, 96).

Celik, C. [2008]. Effective Use of Space for Pivot-Based Metric Indexing Structures. In: *Proc. of the 1st Intl. Workshop on Similarity Search and Applications*. Cancun, Mexico: IEEE, pp. 113–120 (cit. on pp. 38, 40).

Chakrabarti, S., van den Berg, M., and Dom, B. [1999]. Focused Crawling: A new approach to topic-specific Web resource discovery. In: *Proc. of the 8th Intl. World Wide Web Conference*. Toronto, Canada: Elsevier North-Holland, Inc., pp. 1623–1640 (cit. on p. 142).

Chandrasekaran, K., Gauch, S., Lakkaraju, P., and Luong, H. P. [2008]. Concept-Based Document Recommendations for CiteSeer Authors. In: *Proc. of the 5th Intl. Conf. on Adaptive Hypermedia and Adaptive Web-Based Systems*. Hannover, Germany: Springer LNCS 5149, pp. 83–92 (cit. on p. 19).

Chang, W., Sheikholeslami, G., Zhang, A., and Syeda-Mahmood, T. F. [1997]. Efficient Resource Selection in Distributed Visual Information Systems. In: *Proc. of the 5th Intl. Conf. on Multimedia*. Seattle, WA, USA: ACM, pp. 203–213 (cit. on pp. 62, 67).

Chang, W. and Zhang, A. [1997]. Metadata for Distributed Visual Database Access. In: *Proc. of the 2nd Metadata Conf.* Silver Spring, MD, USA: IEEE (cit. on p. 62).

Chatzichristofis, S. A. and Boutalis, Y. S. [2008]. CEDD: Color and Edge Directivity Descriptor: A Compact Descriptor for Image Indexing and Retrieval. In: *Proc. of the 6th Intl. Conf. on Computer Vision Systems*. Santorini, Greece: Springer LNCS 5008, pp. 312–322 (cit. on pp. 116, 146).

Chatzichristofis, S. A., Zagoris, K., Boutalis, Y. S., and Papamarkos, N. [Mar. 2010]. Accurate Image Retrieval Based on Compact Composite Descriptors and Relevance Feedback Information. *Intl. Journal of Pattern Recognition and Artificial Intelligence*, vol. 24, no. 2, pp. 207–244 (cit. on p. 116).

Chávez, E., Figueroa, K., and Navarro, G. [Sept. 2008]. Effective Proximity Retrieval by Ordering Permutations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30 (9), pp. 1647–1658 (cit. on p. 27).

Chávez, E., Marroquín, J. L., and Navarro, G. [June 2001a]. Fixed Queries Array: A Fast and Economical Data Structure for Proximity Searching. *Multimedia Tools and Applications*, vol. 14 (2), pp. 113–135 (cit. on p. 46).

Chávez, E. and Navarro, G. [July 2005a]. A compact space decomposition for effective metric indexing. *Pattern Recognition Letters*, vol. 26 (9), pp. 1363–1376 (cit. on pp. 42, 49 sq.).

Chávez, E. and Navarro, G. [2005b]. Metric Databases. In: *Encyclopedia of Database Technologies and Applications*. Ed. by L. C. Rivero, J. H.

Doorn, and V. E. Ferraggine. IGI global. Chap. 62, pp. 367–372 (cit. on p. 37).

Chávez, E., Navarro, G., Baeza-Yates, R., and Marroquín, J. L. [Sept. 2001b]. Searching in Metric Spaces. *ACM Computing Surveys*, vol. 33 (3), pp. 273–321 (cit. on pp. 2, 8, 12, 16, 26, 36 sqq., 41, 45 sq., 142, 152, 164).

Ciaccia, P. and Patella, M. [1998]. Bulk Loading the M-tree. In: *Proc. of the 9th Australasian Database Conference*. Perth, Australia: Springer, pp. 15–26 (cit. on p. 48).

Ciaccia, P., Patella, M., and Zezula, P. [1997]. M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In: *Proc. of the 23rd Intl. Conf. on Very Large Data Bases*. Athens, Greece: Morgan Kaufmann, pp. 426–435 (cit. on pp. 12, 42, 47 sq., 134, 163).

Clark, D. [Jan. 2001]. Face-to-Face with Peer-to-Peer Networking. *Computer*, vol. 34 (1), pp. 18–21 (cit. on p. 4).

Clarkson, K. L. [2006]. Nearest-Neighbor Searching and Metric Space Dimensions. In: *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. Ed. by G. Shakhnarovich, T. Darrell, and P. Indyk. MIT Press. Chap. 2 (cit. on p. 16).

Comer, D. [June 1979]. Ubiquitous B-Tree. *ACM Computing Surveys*, vol. 11, no. 2, pp. 121–137 (cit. on pp. 46, 52, 78, 93).

Connor, R., Simeoni, F., Iakovos, M., and Moss, R. [June 2011a]. A bounded distance metric for comparing tree structure. *Information Systems*, vol. 36 (4), pp. 748–764 (cit. on p. 19).

Connor, R., Simeoni, F., Iakovos, M., and Moss, R. [2011b]. Towards a Universal Information Distance for Structured Data. In: *Proc. of the 4th Intl. Conf. on Similarity Search and Applications*. Lipari Island, Italy: ACM, pp. 69–77 (cit. on p. 23).

Crespo, A. and Garcia-Molina, H. [2002]. Routing Indices For Peer-to-Peer Systems. In: *Proc. of the 22nd Intl. Conf. on Distributed Computing Systems*. Vienna, Austria: IEEE, pp. 23–32 (cit. on p. 66).

Crespo, A. and Garcia-Molina, H. [2005]. Semantic Overlay Networks for P2P Systems. In: *Proc. of the 3rd Intl. Workshop on Agents and Peer-to-Peer Computing*. New York, NY, USA: Springer LNCS 3601, pp. 1–13 (cit. on p. 64).

Croft, B., Metzler, D., and Strohman, T. [2010]. *Search Engines: Information Retrieval in Practice*. Upper Saddle River, New Jersey, USA: Pearson, International Edition (cit. on pp. 19, 54).

Cuenca-Acuna, F. M., Peery, C., Martin, R. P., and Nguyen, T. D. [2003]. PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities. In: *Proc. of the 12th Intl. Symp. on High Performance Distributed Computing*. Seattle, WA, USA: IEEE, pp. 236–246 (cit. on pp. 5, 7, 61, 64, 66).

Datta, R., Joshi, D., Li, J., and Wang, J. Z. [Apr. 2008]. Image Retrieval: Ideas, Influences, and Trends of the New Age. *ACM Computing Surveys*, vol. 40, no. 2, 5:1–5:60 (cit. on p. 4).

De Amorim, R. C. [2013]. An Empirical Evaluation of Different Initializations on the Number of K-Means Iterations. In: *Proc. of the 11th Mexican Intl. Conf. on Artificial Intelligence*. San Luis Potosí, Mexico: Springer LNCS 7629, pp. 15–26 (cit. on p. 43).

Deselaers, T., Keysers, D., and Ney, H. [Apr. 2008]. Features for image retrieval: an experimental comparison. *Information Retrieval*, vol. 11, no. 2, pp. 77–107 (cit. on p. 22).

Deza, M. M. and Deza, E. [2009]. *Encyclopedia of Distances*. Springer Berlin Heidelberg (cit. on pp. 16 sq., 19 sqq.).

Do, M. N. and Vetterli, M. [Feb. 2002]. Wavelet-Based Texture Retrieval Using Generalized Gaussian Density and Kullback–Leibler Distance. *IEEE Transactions on Image Processing*, vol. 11 (2), pp. 146–158 (cit. on p. 22).

Dohnal, V., Gennaro, C., Savino, P., and Zezula, P. [Sept. 2003]. D-Index: Distance Searching Index for Metric Data Sets. *Multimedia Tools and Applications*, vol. 21 (1), pp. 9–33 (cit. on p. 52).

Dohnal, V. and Sedmidubský, J. [2009]. Query Routing Mechanisms in Self-Organizing Search Systems. In: *Proc. of the 2nd Intl. Workshop on Similarity Search and Applications*. Prague, Czech Republic: IEEE, pp. 132–139 (cit. on p. 72).

Dolin, R., Agrawal, D., El Abbadi, A., and Dillon, L. K. [1997]. Pharos: A Scalable Distributed Architecture for Locating Heterogeneous Information Sources. In: *Proc. of the 6th Intl. Conf. on Information and Knowledge Management*. Las Vegas, NV, USA: ACM, pp. 348–355 (cit. on p. 109).

Doulkeridis, C., Vlachou, A., Kotidis, Y., and Vazirgiannis, M. [2007]. Peer-to-Peer Similarity Search in Metric Spaces. In: *Proc. of the 33rd Intl. Conf. on Very Large Data Bases*. Vienna, Austria: VLDB Endowment, pp. 986–997 (cit. on pp. 42, 66, 73 sq., 121).

Doulkeridis, C., Vlachou, A., Kotidis, Y., and Vazirgiannis, M. [Dec. 2009a]. Efficient range query processing in metric spaces over highly distributed data. *Distributed and Parallel Databases*, vol. 26 (2-3), pp. 155–180 (cit. on p. 74).

Doulkeridis, C., Vlachou, A., Nørvåg, K., Kotidis, Y., and Vazirgiannis, M. [2009b]. Multidimensional Routing Indices for Efficient Distributed Query Processing. In: *Proc. of the 18th Intl. Conf. on Information and Knowledge Management*. Hong Kong, China: ACM, pp. 1489–1492 (cit. on p. 65).

Doulkeridis, C., Vlachou, A., Nørvåg, K., Kotidis, Y., and Vazirgiannis, M. [Mar. 2010a]. Efficient Search based on Content Similarity over Self-

Organizing P2P Networks. *Peer-to-Peer Networking and Applications*, vol. 3 (1), pp. 67–79 (cit. on p. 74).

Doulkeridis, C., Vlachou, A., Nørvåg, K., and Vazirgiannis, M. [2010b]. Distributed Semantic Overlay Networks. In: *Handbook of Peer-to-Peer Networking*. Ed. by X. Shen, H. Yu, J. Buford, and M. Akon. Part IV. Springer Science+Business Media, pp. 463–494 (cit. on p. 64).

Eisenhardt, M., Müller, W., Blank, D., El Allali, S., and Henrich, A. [June 2008]. Clustering-Based, Load Balanced Source Selection for CBIR in P2P Networks. *Intl. Journal of Semantic Computing*, vol. 2 (2), pp. 235–252 (cit. on pp. 73, 152).

Eisenhardt, M., Müller, W., Henrich, A., Blank, D., and El Allali, S. [2006]. Clustering-Based Source Selection for Efficient Image Retrieval in Peer-to-Peer Networks. In: *Proc. of the 8th Intl. Symp. on Multimedia*. San Diego, CA, USA: IEEE, pp. 823–830 (cit. on pp. III, 8, 10, 13, 27, 72 sq., 81, 95, 101 sqq., 107, 133 sq., 136, 139, 146, 151 sq.).

El Allali, S., Blank, D., Müller, W., and Henrich, A. [2008]. Image Data Source Selection Using Gaussian Mixture Models. In: *Adaptive Multimedia Retrieval: Retrieval, User, and Semantics: 5th Intl. Workshop*. Paris, France: Springer LNCS 4918, pp. 170–181 (cit. on pp. 42, 62, 121).

Elahi, B. M., Römer, K., Ostermaier, B., Fahrmair, M., and Kellerer, W. [2009]. Sensor Ranking: A Primitive for Efficient Content-based Sensor Search. In: *Proc. of the 8th Intl. Conf. on Information Processing in Sensor Networks*. San Francisco, CA, USA: ACM/IEEE, pp. 217–228 (cit. on p. 143).

Elsas, J. L., Arguello, J., Callan, J., and Carbonell, J. G. [2008]. Retrieval and Feedback Models for Blog Feed Search. In: *Proc. of the 31st Intl. SIGIR Conf. on Research and Development in Information Retrieval*. Singapore: ACM, pp. 347–354 (cit. on p. 150).

Endres, D. M. and Schindelin, J. E. [July 2003]. A New Metric for Probability Distributions. *IEEE Transactions on Information Theory*, vol. 49 (7), pp. 1858–1860 (cit. on p. 23).

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. [1996]. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: *Proc. of the 2nd Intl. Conf. on Knowledge Discovery and Data Mining*. Portland, OR, USA: AAAI, pp. 226–231 (cit. on p. 44).

Ester, M., Kriegel, H.-P., Sander, J., Wimmer, M., and Xu, X. [1998]. Incremental Clustering for Mining in a Data Warehousing Environment. In: *Proc. of the 24rd Intl. Conf. on Very Large Data Bases*. New York City, USA: Morgan Kaufmann, pp. 323–333 (cit. on p. 44).

Esuli, A. [2009]. MiPai: using the PP-Index to build an efficient and scalable similarity search system. In: *Proc. of the 2nd Intl. Workshop on Simi-*

larity Search and Applications. Prague, Czech Republic: IEEE, pp. 146–148 (cit. on pp. 27, 53).

Fagin, R., Kumar, R., and Sivakumar, D. [2003]. Comparing top k lists. In: *Proc. of the 14th Symp. on Discrete Algorithms*. Baltimore, MD, USA: ACM-SIAM, pp. 28–36 (cit. on p. 53).

Falchi, F., Gennaro, C., and Zezula, P. [2007]. A Content-Addressable Network for Similarity Search in Metric Spaces. In: *Proc. of the 2005/2006 Intl. Workshop on Databases, Information Systems, and Peer-to-Peer Computing*. Trondheim, Norway: Springer LNCS 4125, pp. 98–110 (cit. on p. 59).

Figueroa Mora, K. and Paredes, R. [2010]. Finding Good Permutants for Proximity Searching in Metric Spaces. In: *Proc. of the Intl. Conf. on Information Security and Artificial Intelligence*. Chengdu, China: IEEE, pp. 320–323 (cit. on p. 39).

Figueroa, K., Chávez, E., Navarro, G., and Paredes, R. [Dec. 2009]. Speeding up Spatial Approximation Search in Metric Spaces. *ACM Journal of Experimental Algorithmics*, vol. 14, no. 3.6, 3.6:1–3.6:21 (cit. on p. 46).

Figueroa, K. and Fredriksson, K. [2007]. Simple Space-Time Trade-Offs for AESA. In: *Proc. of the 6th Intl. Workshop on Experimental Algorithms*. Rome, Italy: Springer LNCS 4525, pp. 229–241 (cit. on pp. 46 sq.).

Frakes, W. B. [1992]. Introduction to Information Storage and Retrieval Systems. In: *Information Retrieval: Data Structures & Algorithms*. Ed. by W. B. Frakes and R. Baeza-Yates. Prentice-Hall. Chap. 1 (cit. on p. 5).

Fredriksson, K. [Jan. 2007]. Engineering Efficient Metric Indexes. *Pattern Recognition Letters*, vol. 28 (1), pp. 75–84 (cit. on p. 46).

Gaede, V. and Günther, O. [June 1998]. Multidimensional Access Methods. *ACM Computing Surveys*, vol. 30 (2), pp. 170–231 (cit. on p. 9).

Ganti, V., Ramakrishnan, R., Gehrke, J., Powell, A., and French, J. [1999]. Clustering Large Datasets in Arbitrary Metric Spaces. In: *Proc. of the 15th Intl. Conf. on Data Engineering*. Sydney, Australia: IEEE, pp. 502–511 (cit. on p. 41).

Garcés-Erice, L., Biersack, E. W., Ross, K. W., Felber, P. A., and Urvoy-Keller, G. [Dec. 2003]. Hierarchical Peer-To-Peer Systems. *Parallel Processing Letters*, vol. 13 (4), pp. 643–657 (cit. on p. 65).

Gennaro, C., Amato, G., Bolettieri, P., and Savino, P. [2010]. An Approach to Content-Based Image Retrieval Based on the Lucene Search Engine Library. In: *Proc. of the 14th European Conf. on Research and Advanced Technology for Digital Libraries*. Glasgow, UK: Springer LNCS 6273, pp. 55–66 (cit. on pp. 55 sq.).

Gennaro, C., Lenzi, R., Mandreoli, F., Martoglia, R., Mordacchini, M., Penzo, W., and Sassatelli, S. [Apr. 2011]. A unified multimedia and semantic

perspective for data retrieval in the semantic web. *Information Systems*, vol. 36 (2), pp. 174–191 (cit. on p. 71).

Gennaro, C., Mordacchini, M., Orlando, S., and Rabitti, F. [2008]. Processing Complex Similarity Queries in Peer-to-Peer Networks. In: *Proc. of the 23rd Intl. Symp. on Applied Computing*. Fortaleza, Ceará, Brazil: ACM, pp. 473–478 (cit. on pp. 66, 70 sq.).

Gennaro, C., Savino, P., and Zezula, P. [2001]. Similarity Search in Metric Databases through Hashing. In: *Proc. of the 3rd Intl. Workshop on Multimedia Information Retrieval*. Ottawa, Canada: ACM, pp. 1–5 (cit. on p. 52).

Gil-Costa, V., Marin, M., and Reyes, N. [Mar. 2009]. Parallel query processing on distributed clustering indexes. *Journal of Discrete Algorithms*, vol. 7 (1), pp. 3–17 (cit. on pp. 57 sq., 61).

Gravano, L. and Garcia-Molina, H. [1995]. Generalizing GlOSS to Vector-Space Databases and Broker Hierarchies. In: *Proc. of the 21th Intl. Conf. on Very Large Data Bases*. Zürich, Switzerland: Morgan Kaufmann, pp. 78–89 (cit. on p. 65).

Green, J. A. [1988]. *Sets and Groups: A First Course in Algebra*. Routledge and Kegan Paul (cit. on p. 26).

Guttman, A. [1984]. R-Trees: A Dynamic Index Structure for Spatial Searching. In: *Proc. of the Intl. SIGMOD Conf. on Management of Data*. Boston, MA, USA: ACM, pp. 47–57 (cit. on pp. 46 sq., 115, 141).

Guy, I., Jacovi, M., Perer, A., Ronen, I., and Uziel, E. [2010]. Same Places, Same Things, Same People? Mining User Similarity on Social Media. In: *Proc. of the Intl. Conf. on Computer Supported Cooperative Work*. Savannah, GA, USA: ACM, pp. 41–50 (cit. on p. 143).

Halevy, A. Y., Ives, Z. G., Suciu, D., and Tatarinov, I. [2003]. Schema Mediation in Peer Data Management Systems. In: *Proc. of the 19th Intl. Conf. on Data Engineering*. Bangalore, India: IEEE, pp. 505–516 (cit. on p. 71).

Hamming, R. W. [Apr. 1950]. Error Detecting and Error Correcting Codes. *The Bell System Technical Journal*, vol. 29 (2), pp. 147–160 (cit. on p. 20).

Han, J. and Kamber, M. [2006]. *Data Mining: Concepts and Techniques*. 2nd ed. San Francisco, CA, USA: Morgan Kaufmann (cit. on pp. 42 sqq.).

Han, P., Xie, B., Yang, F., and Shen, R. [Aug. 2004]. A scalable P2P recommender system based on distributed collaborative filtering. *Expert Systems with Applications*, vol. 27 (2), pp. 203–210 (cit. on p. 4).

Hansen, V. L. [1999]. *Fundamental Concepts in Modern Analysis*. World Scientific (cit. on p. 9).

Hariharan, R., Hore, B., and Mehrotra, S. [2008]. Discovering GIS Sources on the Web using Summaries. In: *Proc. of the 8th Joint Conf. on Digital Libraries*. Pittsburgh, PA, USA: ACM/IEEE, pp. 94–103 (cit. on p. 5).

Hattori, Y. [2003]. Metric Spaces. In: *Encyclopedia of General Topology*. Ed. by K. P. Hart, J. Nagata, and J. E. Vaughan. Elsevier. Chap. e-1, pp. 235–238 (cit. on pp. 16 sq.).

Henrich, A. [2008]. *Information Retrieval 1: Grundlagen, Modelle und Anwendungen*. Version 1.2, *http://www.uni-bamberg.de/minf/ir1_buch/* (last visit: 1.10.2014). Universität Bamberg, Lehrstuhl für Medieninformatik (cit. on p. 5).

Henrich, A. and Blank, D. [2012]. *Summarizing data collections by their spatial, temporal, textual and image footprint: Techniques for source selection and beyond*. Keynote Talk at the DFG SPP 1335 Text Workshop on Scalable Visual Analytics (held by: A. Henrich). Leipzig, Germany. November 15, 2012. Slides online: *https://www.uni-bamberg.de/fileadmin/uni/fakultaeten/wiai_lehrstuehle/medieninformatik/Dateien/Publikationen/2012/Henrich-Leipzig-2012.pdf* (last visit: 8.10.2014) (cit. on p. 144).

Hetland, M. L. [2009a]. Ptolemaic Indexing. CoRR, abs/0911.4384, *http://arxiv.org/abs/0911.4384* (last visit: 1.10.2014) (cit. on p. 17).

Hetland, M. L. [2009b]. The Basic Principles of Metric Indexing. In: *Swarm Intelligence for Multi-objective Problems in Data Mining*. Ed. by C. A. C. Coello, S. Dehuri, and S. Ghosh. Springer. Chap. 9, pp. 199–232 (cit. on pp. 16, 25, 30, 32, 36 sq., 45 sqq., 49 sq., 152).

Hetland, M. L., Skopal, T., Lokoč, J., and Beecks, C. [Oct. 2013]. Ptolemaic access methods: Challenging the reign of the metric space model. *Information Systems*, vol. 38 (7), pp. 989–1006 (cit. on p. 17).

Heymann, P., Koutrika, G., and Garcia-Molina, H. [Nov. 2007]. Fighting Spam on Social Web Sites: A Survey of Approaches and Future Challenges. *IEEE Internet Computing*, vol. 11 (6), pp. 36–45 (cit. on p. 4).

Hjaltason, G. R. and Samet, H. [2000]. *Contractive Embedding Methods for Similarity Searching in Metric Spaces*. Tech. rep. CAR-TR-938, CS-TR-4102, IRI-97-12715. Computer Science Department, Center for Automation Research, Institute for Advanced Computer Studies, University of Maryland (cit. on p. 50).

Hjaltason, G. R. and Samet, H. [Dec. 2003a]. Index-Driven Similarity Search in Metric Spaces. *ACM Transactions on Database Systems*, vol. 28 (4), pp. 517–580 (cit. on pp. 16, 24 sqq., 44).

Hjaltason, G. R. and Samet, H. [May 2003b]. Properties of Embedding Methods for Similarity Searching in Metric Spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25 (5), pp. 530–549 (cit. on pp. 50 sq.).

Hu, R., Rüger, S., Song, D., Liu, H., and Huang, Z. [2008]. Dissimilarity measures for content-based image retrieval. In: *Proc. of the Intl. Conf. on Multimedia and Expo*. Hannover, Germany: IEEE, pp. 1365–1368 (cit. on pp. 18, 20 sqq.).

Hu, X., Chiueh, T.-c., and Shin, K. G. [2009]. Large-Scale Malware Indexing Using Function-Call Graphs. In: *Proc. of the 16th Intl. Conf. on Computer and Communications Security*. Chicago, IL, USA: ACM, pp. 611–620 (cit. on pp. 2, 20, 142).

Huiskes, M. J. and Lew, M. S. [2008]. The MIR Flickr Retrieval Evaluation. In: *Proc. of the 1st Intl. Conf. on Multimedia Information Retrieval*. Vancouver, Canada: ACM, pp. 39–43 (cit. on p. 146).

Ilyas, I. F., Beskales, G., and Soliman, M. A. [Oct. 2008]. A Survey of Top-k Query Processing Techniques in Relational Database Systems. *ACM Computing Surveys*, vol. 40 (4), 11:1–11:58 (cit. on p. 5).

Jagadish, H. V., Ooi, B. C., Tan, K.-L., Yu, C., and Zhang, R. [June 2005]. iDistance: An Adaptive B^+-Tree Based Indexing Method for Nearest Neighbor Search. *ACM Transactions on Database Systems*, vol. 30 (2), pp. 364–397 (cit. on pp. 51, 68, 73).

Julesz, B. [Mar. 1981]. Textons, the elements of texture perception, and their interactions. *Nature*, vol. 290 (5802), pp. 91–97 (cit. on p. 17).

Kaufman, L. and Rousseeuw, P. J. [1990]. *Finding Groups in Data: An Introduction to Cluster Analysis*. New York: John Wiley (cit. on p. 43).

Keim, D., Andrienko, G., Fekete, J.-D., Görg, C., Kohlhammer, J., and Melançon, G. [2008]. Visual Analytics: Definition, Process, and Challenges. In: *Information Visualization: Human-Centered Issues and Perspectives*. Ed. by A. Kerren, J. T. Stasko, J.-D. Fekete, and C. North. Springer LNCS 4950, pp. 154–175 (cit. on p. 144).

Kim, D.-H. and Chung, C.-W. [Mar. 2003]. Collection fusion using Bayesian estimation of a linear regression model in image databases on the Web. *Information Processing and Management*, vol. 39 (2), pp. 267–285 (cit. on p. 62).

Kim, D.-H., Lee, S.-L., and Chung, C.-W. [Nov. 2002]. Heterogeneous image database selection on the Web. *The Journal of Systems and Software*, vol. 64 (2), pp. 131–149 (cit. on p. 62).

Kirk, P. [2003]. *Gnutella*. *http://rfc-gnutella.sourceforge.net/* (last visit: 1.10.2014) (cit. on p. 58).

Kokare, M., Chatterji, B. N., and Biswas, P. K. [2003]. Comparison of Similarity Metrics for Texture Image Retrieval. In: *Proc. of the Conf. on Convergent Technologies for the Asia-Pacific Region*. Vol. 2. Taj Residency, Bangalore: IEEE, pp. 571–575 (cit. on p. 22).

Kriegel, H.-P., Kröger, P., Kunath, P., and Renz, M. [2007]. Generalizing the Optimality of Multi-step k-Nearest Neighbor Query Processing. In: *Proc. of the 10th Intl. Conf. on Advances in Spatial and Temporal*

Databases. Boston, MA, USA: Springer LNCS 4605, pp. 75–92 (cit. on p. 8).

Kufer, S., Blank, D., and Henrich, A. [2012]. Techniken der Ressourcenbeschreibung und -auswahl für das geographische Information Retrieval. In: *Proc. of LWA Workshop – Lernen, Wissen, Adaptivität*. Dortmund, Germany https://www.uni-bamberg.de/fileadmin/uni/fakultaeten/wiai_lehrstuehle/medieninformatik/Dateien/Publikationen/2012/kufer_2012_techniken.pdf (last visit: 1.10.2014) (cit. on p. 115).

Kufer, S., Blank, D., and Henrich, A. [2013]. Using Hybrid Techniques for Resource Description and Selection in the Context of Distributed Geographic Information Retrieval. In: *Proc. of the 13th Intl. Symp. on Advances in Spatial and Temporal Databases*. Munich, Germany: Springer LNCS 8098, pp. 330–347 (cit. on p. 115).

Kufer, S. and Henrich, A. [2014]. Hybrid Quantized Resource Descriptions for Geospatial Source Selection. In: *Proc. of the 4th Intl. Workshop on Location and the Web*. Shanghai, China: ACM, to appear (cit. on p. 115).

Kullback, S. and Leibler, R. A. [Mar. 1951]. On Information and Sufficiency. *Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86 (cit. on p. 22).

Kunze, M. and Weske, M. [2011]. Metric Trees for Efficient Similarity Search in Large Process Model Repositories. *Business Process Management Workshops*. Springer Lecture Notes in Business Information Processing, vol. 66, pp. 535–546 (cit. on pp. 2, 20, 47, 142).

Lalmas, M. [2009]. *XML Retrieval*. Synthesis Lectures on Information Concepts, Retrieval and Services. Morgan & Claypool Publishers (cit. on p. 150).

Lalmas, M. [2011]. Aggregated Search. In: *Advanced Topics in Information Retrieval*. Ed. by M. Melucci and R. Baeza-Yates. Vol. 33. The Information Retrieval Series. Springer. Chap. 5, pp. 109–123 (cit. on p. 144).

Larson, R. R. [2007]. Probabilistic Retrieval Approaches for Thorough and Heterogeneous XML retrieval. In: *Comparative Evaluation of XML Information Retrieval Systems*. Ed. by N. Fuhr, M. Lalmas, and A. Trotman. Springer LNCS 4518, pp. 318–330 (cit. on p. 150).

Lee, J. [2006]. A Graph-Based Approach for Modeling and Indexing Video Data. In: *Proc. of the 8th Intl. Symp. on Multimedia*. San Diego, CA, USA: IEEE, pp. 348–355 (cit. on p. 20).

Lerner, F. A. [2009]. *The Story of Libraries: From the Invention of Writing to the Computer Age*. 2nd ed. New York, London: Continuum International Publishing Group Inc. (cit. on p. 1).

Levenshtein, V. I. [Feb. 1966]. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707–710 (cit. on p. 19).

Li, H. and Wolters, L. [2006]. An Investigation of Grid Performance Predictions Through Statistical Learning. In: *Proc. of the 1st Intl. Workshop on Tackling Computer System Problems with Machine Learning Techniques*. Saint-Malo, France: *http://research.microsoft.com/en-us/um/redmond/events/sysml/papers/sysml-Li.pdf* (last visit: 1.10.2014) (cit. on p. 142).

Linari, A. and Patella, M. [2007]. Metric Overlay Networks: Processing Similarity Queries in P2P Databases. In: *Proc. of the 5th Intl. Workshop on Databases, Information Systems and Peer-to-Peer Computing*. Vienna, Austria (cit. on pp. 69 sq.).

Linari, A. and Weikum, G. [2006]. Efficient Peer-to-Peer Semantic Overlay Networks based on Statistical Language Models. In: *Proc. of the Intl. Workshop on Information Retrieval in Peer-to-Peer Networks*. Arlington, VA, USA: ACM, pp. 9–16 (cit. on p. 70).

Liu, H., Song, D., Rüger, S., Hu, R., and Uren, V. [2008]. Comparing Dissimilarity Measures for Content-Based Image Retrieval. In: *4th Asia Infomation Retrieval Symp., Revised Selected Papers*. Harbin, China: Springer LNCS 4993, pp. 44–50 (cit. on pp. 20 sqq., 116).

Liu, T., Rosenberg, C., and Rowley, H. A. [2007]. Clustering Billions of Images with Large Scale Nearest Neighbor Search. In: *Proc. of the 8th Workshop on Applications of Computer Vision*. Austin, TX, USA: IEEE, pp. 28–33 (cit. on p. 59).

Lodi, S., Penzo, W., Mandreoli, F., Martoglia, R., and Sassatelli, S. [2008]. Semantic Peer, Here are the Neighbors You Want! In: *Proc. of the 11th Intl. Conf. on Extending Database Technology: Advances in Database Technology*. Nantes, France: ACM, pp. 26–37 (cit. on pp. 23, 71).

Lokoč, J., Hetland, M. L., Skopal, T., and Beecks, C. [2011]. Ptolemaic Indexing of the Signature Quadratic Form Distance. In: *Proc. of the 4th Intl. Conf. on Similarity Search and Applications*. Lipari Island, Italy: ACM, pp. 9–16 (cit. on p. 18).

Lokoč, J., Moško, J., Čech, P., and Skopal, T. [July 2014]. On indexing metric spaces using cut-regions. *Information Systems*, vol. 43, pp. 1–19 (cit. on pp. 78, 80).

Lokoč, J. and Skopal, T. [2010]. On Applications of Parameterized Hyperplane Partitioning. In: *Proc. of the 3rd Intl. Conf. on Similarity Search and Applications*. Istanbul, Turkey: ACM, pp. 131–132 (cit. on p. 29).

Lozupone, C. and Knight, R. [Dec. 2005]. UniFrac: a New Phylogenetic Method for Comparing Microbial Communities. *Applied and Environmental Microbiology*, vol. 71, no. 12, pp. 8228–8235 (cit. on p. 23).

Lu, J. [2007]. Full-Text Federated Search in Peer-to-Peer Networks. CMU-LTI-07-003. PhD thesis. Language Technologies Institute, School of Computer Science, Carnegie Mellon University (cit. on pp. 57, 62 sq., 66).

Lupu, M., Li, J., Ooi, B. C., and Shi, S. [2007]. Clustering wavelets to speed-up data dissemination in structured P2P MANETs. In: *Proc. of the 23th Intl. Conf. on Data Engineering*. Istanbul, Turkey: IEEE, pp. 386–395 (cit. on pp. 63, 68, 143).

Lv, Q., Cao, P., Cohen, E., Li, K., and Shenker, S. [2002]. Search and Replication in Unstructured Peer-to-Peer Networks. In: *Proc. of the 16th Intl. Conf. on Supercomputing*. New York City, USA: ACM, pp. 84–95 (cit. on p. 58).

Mamou, J., Mass, Y., Shmueli-Scheuer, M., and Sznajder, B. [2009]. A Unified Inverted Index for an Efficient Image and Text Retrieval. In: *Proc. of the 32nd Intl. SIGIR Conf. on Research and Development in Information Retrieval*. Boston, MA, USA: ACM, pp. 814–815 (cit. on p. 56).

Mancini, V., Bustos, F., Gil-Costa, V., and Printista, A. M. [2012]. Data Partitioning Evaluation for Multimedia Systems in Hybrid Environments. In: *Proc. of the 7th Intl. Conf. on P2P, Parallel, Grid, Cloud and Internet Computing*. Victoria, Canada: IEEE, pp. 321–326 (cit. on pp. 57, 61).

Mandreoli, F., Martoglia, R., Sassatelli, S., and Penzo, W. [2006]. SRI: Exploiting Semantic Information for Effective Query Routing in a PDMS. In: *Proc. of the 8th Intl. Workshop on Web Information and Data Management*. Arlington, VA, USA: ACM, pp. 19–26 (cit. on p. 71).

Manjunath, B. S., Salembier, P., and Sikora, T., eds. [2002]. *Introduction to MPEG-7: Multimedia Content Description Interface*. Wiley & Sons (cit. on p. 82).

Manning, C. D., Raghavan, P., and Schütze, H. [2008]. *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press (cit. on pp. 21, 42).

Manning, C. D. and Schütze, H. [1999]. *Foundations of Statistical Natural Language Processing*. Cambridge, MA, USA: MIT Press (cit. on p. 22).

Mao, R., Miranker, W. L., and Miranker, D. P. [2010]. Dimension Reduction for Distance-Based Indexing. In: *Proc. of the 3rd Intl. Conf. on Similarity Search and Applications*. Istanbul, Turkey: ACM, pp. 25–32 (cit. on pp. 37 sq.).

Mardia, K. V., Kent, J. T., and Bibby, J. M. [1979]. *Multivariate Analysis*. London: Academic Press (cit. on p. 122).

Marin, M., Gil-Costa, V., and Hernandez, C. [2009]. Dynamic P2P Indexing and Search Based on Compact Clustering. In: *Proc. of the 2nd Intl. Workshop on Similarity Search and Applications*. Prague, Czech Republic: IEEE, pp. 124–131 (cit. on pp. 50, 74 sq.).

Mass, Y., Sagiv, Y., and Shmueli-Scheuer, M. [2011]. KMV-Peer: A Robust and Adaptive Peer-Selection Algorithm. In: *Proc. of the 4th Intl. Conf. on Web Search and Data Mining*. Hong Kong, China: ACM, pp. 157–166 (cit. on p. 7).

Micó, M. L., Oncina, J., and Vidal, E. [Jan. 1994]. A new version of the Nearest-Neighbour Approximating and Eliminating Search Algorithm (AESA) with linear preprocessing time and memory requirements. *Pattern Recognition Letters*, vol. 15 (1), pp. 9–17 (cit. on p. 46).

Miller, G. A. [Nov. 1995]. WordNet: A Lexical Database for English. *Communications of the ACM*, vol. 38 (11), pp. 39–41 (cit. on p. 23).

Milojicic, D. S., Kalogeraki, V., Lukose, R., Nagaraja, K., Pruyne, J., Richard, B., Rollins, S., and Xu, Z. [2002]. *Peer-to-Peer Computing*. Tech. rep. HPL-2002-57 (R.1). Palo Alto, CA, USA: HP Laboratories (cit. on p. 4).

Mohajer, M., Englmeier, K.-H., and Schmid, V. J. [2011]. A comparison of Gap statistic definitions with and without logarithm function. CoRR, abs/1103.4767, *http://arxiv.org/abs/1103.4767* (last visit: 2.10.2014) (cit. on p. 122).

Mohamed, H. and Marchand-Maillet, S. [2012]. Parallel Approaches to Permutation-Based Indexing Using Inverted Files. In: *Proc. of the 5th Intl. Conf. on Similarity Search and Applications*. Toronto, Canada: Springer LNCS 7404, pp. 148–161 (cit. on p. 54).

Muja, M. and Lowe, D. G. [2013]. Fast Matching of Binary Features. In: *Proc. of the 9th Intl. Conf. on Computer and Robot Vision 2012*. Toronto, Canada: IEEE, pp. 404–410 (cit. on pp. 20, 41).

Müller, H., Squire, D. M., Müller, W., and Pun, T. [1999]. Efficient access methods for content-based image retrieval with inverted files. *Proc. SPIE 3846, Multimedia Storage and Archiving Systems IV*, pp. 461–472 (cit. on p. 56).

Müller, W., Eisenhardt, M., and Henrich, A. [Oct. 2005a]. Fast retrieval of high-dimensional feature vectors in P2P networks using compact peer data summaries. *Multimedia Systems*, vol. 10 (6), pp. 464–474 (cit. on pp. III, 8, 10, 72, 81, 151).

Müller, W., Eisenhardt, M., and Henrich, A. [2005b]. Scalable Summary Based Retrieval in P2P Networks. In: *Proc. of the 14th Intl. Conf. on Information and Knowledge Management*. Bremen, Germany: ACM, pp. 586–593 (cit. on pp. 10, 64 sqq., 103).

Müller, W. and Henrich, A. [2003]. Fast Retrieval of High-Dimensional Feature Vectors in P2P Networks Using Compact Peer Data Summaries. In: *Proc. of the 5th Intl. Workshop on Multimedia Information Retrieval*. Berkeley, CA, USA: ACM, pp. 79–86 (cit. on p. 72).

Müller, W., Sarshar, N., and Boykin, P. O. [2006]. Comparison of Image Similarity Queries in P2P Systems. In: *Proc. of the 6th Intl. Conf. on*

Peer-to-Peer Computing. Cambridge, UK: IEEE, pp. 98–105 (cit. on p. 4).

Myrvold, W. and Ruskey, F. [Sept. 2001]. Ranking and unranking permutations in linear time. *Information Processing Letters*, vol. 79 (6), pp. 281–284 (cit. on p. 80).

Naumann, F. and Herschel, M. [2010]. *An Introduction to Duplicate Detection*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers (cit. on p. 22).

Ng, R. T. and Han, J. [1994]. Efficient and Effective Clustering Methods for Spatial Data Mining. In: *Proc. of the 20th Intl. Conf. on Very Large Data Bases*. Santiago de Chile, Chile: Morgan Kaufmann, pp. 144–155 (cit. on p. 43).

Nievergelt, J., Hinterberger, H., and Sevcik, K. C. [Mar. 1984]. The Grid File: An Adaptable, Symmetric Multikey File Structure. *ACM Transactions on Database Systems*, vol. 9 (1), pp. 38–71 (cit. on p. 115).

Nistér, D. and Stewénius, H. [2006]. Scalable Recognition with a Vocabulary Tree. In: *Proc. of the Intl. Conf. on Computer Vision and Pattern Recognition – Volume 2*. New York, NY, USA: IEEE, pp. 2161–2168 (cit. on p. 41).

Nottelmann, H. and Fuhr, N. [2004]. Decision-theoretic resource selection for different data types in MIND. In: *Proc. of the 2003 Intl. Workshop on Distributed Information Retrieval*. Toronto, Canada: Springer LNCS 2924, pp. 43–57 (cit. on p. 62).

Nottelmann, H. and Fuhr, N. [2006]. Comparing Different Architectures for Query Routing in Peer-to-Peer Networks. In: *Proc. of the 28th European Conf. on Information Retrieval Research*. London, UK: Springer LNCS 3936, pp. 253–264 (cit. on p. 7).

Novák, D. [2008]. Similarity Search on a Very Large Scale. PhD thesis. Faculty of Informatics, Masaryk University Brno (cit. on pp. 2, 18, 25, 51, 151, 164).

Novák, D. and Batko, M. [2009]. Metric Index: An Efficient and Scalable Solution for Similarity Search. In: *Proc. of the 2nd Intl. Workshop on Similarity Search and Applications*. Prague, Czech Republic: IEEE, pp. 65–73 (cit. on pp. 12, 28, 51 sq.).

Novák, D., Batko, M., and Zezula, P. [June 2011]. Metric Index: An efficient and scalable solution for precise and approximate similarity search. *Information Systems*, vol. 36 (4), pp. 721–733 (cit. on pp. 12, 51 sq., 81, 93).

Novák, D., Batko, M., and Zezula, P. [Sept. 2012]. Large-scale similarity data management with distributed Metric Index. *Information Processing and Management*, vol. 48, no. 5, pp. 855–872 (cit. on p. 59).

Novák, D., Kyselak, M., and Zezula, P. [2010]. On Locality-sensitive Indexing in Generic Metric Spaces. In: *Proc. of the 3rd Intl. Conf. on Similarity*

Search and Applications. Istanbul, Turkey: ACM, pp. 59–66 (cit. on p. 52).

Novák, D. and Zezula, P. [2006]. M-Chord: A Scalable Distributed Similarity Search Structure. In: *Proc. of the 1st Intl. Conf. on Scalable Information Systems*. Hong Kong, China: ACM, article no. 19 (cit. on p. 59).

Novák, D. and Zezula, P. [Oct. 2013]. Performance Study of Independent Anchor Spaces for Similarity Searching. *The Computer Journal*. *http://comjnl.oxfordjournals.org/content/early/2013/10/08/comjnl.bxt114.abstract* (last visit: 2.10.2014) (cit. on pp. 12, 27 sq., 39).

Obrador, P., de Oliveira, R., and Oliver, N. [2010]. Supporting Personal Photo Storytelling for Social Albums. In: *Proc. of the Intl. Conf. on Multimedia*. Firenze, Italy: ACM, pp. 561–570 (cit. on p. 144).

Paltoglou, G., Salampasis, M., and Satratzemi, M. [2008]. Integral Based Source Selection for Uncooperative Distributed Information Retrieval Environments. In: *Proc. of the Workshop on Large-Scale Distributed Systems for Information Retrieval*. Napa Valley, CA, USA: ACM, pp. 67–74 (cit. on p. 7).

Papadopoulos, A. N. and Manolopoulos, Y. [Jan. 2001]. Distributed Processing of Similarity Queries. *Distributed Parallel Databases*, vol. 9 (1), pp. 67–92 (cit. on p. 57).

Papapetrou, O., Siberski, W., Balke, W.-T., and Nejdl, W. [2007]. DHTs over Peer Clusters for Distributed Information Retrieval. In: *Proc. of the 21st Intl. Conf. on Advanced Information Networking and Applications*. Niagara Falls, Canada: IEEE, pp. 84–93 (cit. on p. 63).

Paramita, M. L., Sanderson, M., and Clough, P. [2009]. Diversity in Photo Retrieval: Overview of the ImageCLEF Photo Task 2009. In: *Proc. of the 10th Intl. Cross-Language Evaluation Forum: Multimedia Experiments*. Corfu, Greece: Springer, pp. 45–59 (cit. on pp. 4, 97).

Patella, M. and Ciaccia, P. [Mar. 2009]. Approximate similarity search: A multi-faceted problem. *Journal of Discrete Algorithms*, vol. 7 (1), pp. 36–48 (cit. on pp. 6, 45, 53).

Paterlini, A. A., Nascimento, M. A., and Traina Jr., C. [June 2011]. Using Pivots to Speed-Up k-Medoids Clustering. *Journal of Information and Data Management*, vol. 2, no. 2, pp. 221–236 (cit. on pp. 43 sq.).

Pedreira, O. and Brisaboa, N. R. [2007]. Spatial Selection of Sparse Pivots for Similarity Search in Metric Spaces. In: *Proc. of the 33rd Conf. on Current Trends in Theory and Practice of Computer Science*. Harrachov, Czech Republic: Springer LNCS 4362, pp. 434–445 (cit. on p. 38).

Pele, O. and Werman, M. [2009]. Fast and Robust Earth Mover's Distances. In: *Proc. of the 12th Intl. Conf. on Computer Vision*. Kyoto, Japan: IEEE, pp. 460–467 (cit. on pp. 18 sq.).

Pele, O. and Werman, M. [2010]. The Quadratic-Chi Histogram Distance Family. In: *Proc. of the 11th European Conf. on Computer Vision: Part II*. Heraklion, Crete, Greece: Springer LNCS 6312, pp. 749–762 (cit. on p. 18).

Pestov, V. [2007]. Intrinsic dimension of a dataset: what properties does one expect? In: *Proc. of the Intl. Joint Conf. on Neural Networks*. Orlando, FL, USA: IEEE, pp. 2959–2964 (cit. on pp. 16, 36 sq.).

Picard, D., Revel, A., and Cord, M. [June 2012]. An Application of Swarm Intelligence to Distributed Image Retrieval. *Information Sciences*, vol. 192, pp. 71–81 (cit. on p. 58).

Popescu, A., Tsikrika, T., and Kludas, J. [2010]. Overview of the Wikipedia Retrieval Task at ImageCLEF 2010. In: *CLEF 2010 LABs and Workshops, Notebook Papers*. Padua, Italy: *www.clef-initiative.eu/documents/71612/86374/CLEF2010wn-ImageCLEF-PopescuEt2010.pdf* (last visit: 2.10.2014) (cit. on pp. 4, 97).

Preparata, F. P. and Shamos, M. I. [1985]. *Computational Geometry: An Introduction*. Springer New York (cit. on p. 115).

Qiu, C., Lu, Y., Gao, P., Wang, J., and Lv, R. [2010]. Parallel M-tree Based on Declustering Metric Objects using K-medoids Clustering. In: *Proc. of the 9th Intl. Symp. on Distributed Computing and Applications to Business, Engineering and Science*. Hong Kong, China: IEEE, pp. 206–210 (cit. on p. 43).

Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Shenker, S. [2001]. A Scalable Content-Addressable Network. In: *Proc. of the Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communication*. San Diego, CA, USA: ACM, pp. 161–172 (cit. on p. 59).

Rousseeuw, P. J. [Nov. 1987]. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65 (cit. on p. 123).

Rubner, Y., Tomasi, C., and Guibas, L. J. [Nov. 2000]. The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision*, vol. 40 (2), pp. 99–121 (cit. on pp. 17, 19).

Rüger, S. [2010]. *Multimedia Information Retrieval*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers (cit. on p. 1).

Ruiz, G., Santoyo, F., Chávez, E., Figueroa, K., and Téllez, E. S. [2013]. Extreme Pivots for Faster Metric Indexes. In: *Proc. of the 6th Intl. Conf. on Similarity Search and Applications*. A Coruña, Spain: Springer LNCS 8199, pp. 115–126 (cit. on p. 39).

Russell, S. J. and Norvig, P. [2010]. *Artificial Intelligence: A Modern Approach*. 3rd ed. Pearson Education (cit. on p. 57).

Salzberg, B. and Tsotras, V. J. [June 1999]. Comparison of Access Methods for Time-Evolving Data. *ACM Computing Surveys*, vol. 31 (2), pp. 158–221 (cit. on p. 41).

Samet, H. [2006]. *Foundations of Multidimensional and Metric Data Structures*. San Francisco, CA, USA: Morgan Kaufmann (cit. on pp. 9, 32, 45, 115).

Saroiu, S., Gummadi, P. K., and Gribble, S. D. [2002]. A Measurement Study of Peer-to-Peer File Sharing Systems. In: *Proc. of the ACM/SPIE Conf. on Multimedia Computing and Networking*. San Jose, CA, USA: SPIE 4673, pp. 156–170 (cit. on pp. 103, 110).

Sedmidubský, J. [2010]. Self-organizing Similarity Search: The Social Network Approach. PhD thesis. Faculty of Informatics, Masaryk University Brno (cit. on pp. 58, 72).

Seeger, B. and Kriegel, H.-P. [1990]. The Buddy-Tree: An Efficient and Robust Access Method for Spatial Data Base Systems. In: *Proc. of the 16th Intl. Conf. on Very Large Data Bases*. Brisbane, Australia: Morgan Kaufmann, pp. 590–601 (cit. on p. 115).

Sen, S., Harper, F. M., LaPitz, A., and Riedl, J. [2007]. The Quest for Quality Tags. In: *Proc. of the Intl. Conf. on Supporting Group Work*. Sanibel Island, FL, USA: ACM, pp. 361–370 (cit. on p. 4).

Shannon, C. E. [July 1948]. A Mathematical Theory of Communication. *The Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656 (cit. on p. 19).

Shokouhi, M. and Si, L. [Jan. 2011]. Federated Search. *Foundations and Trends in Information Retrieval*, vol. 5 (1), pp. 1–102 (cit. on pp. 22, 62).

Sibson, R. [1973]. SLINK: An optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, vol. 16 (1), pp. 30–34 (cit. on p. 44).

Sinnott, R. [Aug. 1984]. Virtues of the Haversine. *Sky and Telescope*, vol. 68 (2), p. 159 (cit. on p. 109).

Sivic, J. and Zisserman, A. [2003]. Video Google: A Text Retrieval Approach to Object Matching in Videos. In: *Proc. of the 9th Intl. Conf. on Computer Vision – Volume 2*. Nice, France: IEEE, pp. 1470–1477 (cit. on p. 17).

Skala, M. [2009]. Counting distance permutations. *Journal of Discrete Algorithms*, vol. 7, no. 1, pp. 49–61 (cit. on pp. 12, 28).

Skopal, T. [2004]. Metric Indexing in Information Retrieval. PhD thesis. Faculty of Electrical Engineering and Computer Science, Department of Computer Science, VŠB – Technical University of Ostrava (cit. on p. 9).

Skopal, T. [Nov. 2007]. Unified Framework for Fast Exact and Approximate Search in Dissimilarity Spaces. *ACM Transactions on Database Systems*, vol. 32 (4), 29:1–29:47 (cit. on pp. 6, 21).

Skopal, T. [2010]. Where are you heading, metric access methods? A provocative survey. In: *Proc. of the 3rd Intl. Conf. on Similarity Search and Applications*. Istanbul, Turkey: ACM, pp. 13–21 (cit. on pp. 9, 82).

Skopal, T. and Bustos, B. [Oct. 2011]. On Nonmetric Similarity Search Problems in Complex Domains. *ACM Computing Surveys*, vol. 43 (4), 34:1–34:50 (cit. on pp. 8 sq., 16 sqq., 21, 153).

Skopal, T., Lokoč, J., and Bustos, B. [May 2012]. D-Cache: Universal Distance Cache for Metric Access Methods. *IEEE Transactions on Knowledge and Data Engineering*, vol. 24 (5), pp. 868–881 (cit. on p. 35).

Skopal, T. and Moravec, P. [2005]. Modified LSI Model for Efficient Search by Metric Access Methods. In: *Proc. of the 27th European Conf. on Information Retrieval*. Santiago de Compostela, Spain: Springer LNCS 3408, pp. 245–259 (cit. on pp. 21, 142).

Skopal, T., Pokorný, J., Krátký, M., and Snášel, V. [2003]. Revisiting M-Tree Building Principles. In: *Proc. of the 7th East European Conf. on Advances in Databases and Information Systems*. Dresden, Germany: Springer LNCS 2798, pp. 148–162 (cit. on pp. 48, 88).

Skopal, T., Pokorný, J., and Snášel, V. [2005]. Nearest Neighbours Search Using the PM-Tree. In: *Proc. of 10th Intl. Conf. on Database Systems for Advanced Applications*. Beijing, China: Springer LNCS 3453, pp. 803–815 (cit. on pp. 12, 48).

Smeulders, A. W. M., Worring, M., Santini, S., Gupta, A., and Jain, R. [Dec. 2000]. Content-Based Image Retrieval at the End of the Early Years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22 (12), pp. 1349–1380 (cit. on p. 8).

Smith, J. R. [1997]. Integrated Spatial and Feature Image Systems: Retrieval, Analysis and Compression. PhD thesis. Graduate School of Arts and Sciences, Columbia University, New York, NY, USA (cit. on p. 103).

Socorro, R., Micó, L., and Oncina, J. [Aug. 2011]. A fast pivot-based indexing algorithm for metric spaces. *Pattern Recognition Letters*, vol. 32 (11), pp. 1511–1516 (cit. on pp. 38, 46).

Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H. [2001]. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In: *Proc. of the Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communication*. San Diego, CA, USA: ACM, pp. 149–160 (cit. on p. 59).

Sznajder, B., Mamou, J., Mass, Y., and Shmueli-Scheuer, M. [2008]. Metric inverted – an efficient inverted indexing method for metric spaces. In: *Proc. of the 1st Efficiency Issues in Information Retrieval Workshop*. Glasgow, UK: *http://irlab.dc.fi.udc.es/ecir/sznajder.pdf* (last visit: 3.10.2014) (cit. on pp. 56, 160).

Tatarinov, I. and Halevy, A. [2004]. Efficient Query Reformulation in Peer Data Management Systems. In: *Proc. of the Intl. Conf. on the Management of Data*. Paris, France: ACM, pp. 539–550 (cit. on p. 71).

Téllez, E. S. and Chávez, E. [2010]. On Locality Sensitive Hashing in Metric Spaces. In: *Proc. of the 3rd Intl. Conf. on Similarity Search and Applications*. Istanbul, Turkey: ACM, pp. 67–74 (cit. on p. 53).

Téllez, E. S. and Chávez, E. [2012]. The List of Clusters Revisited. In: *Proc. of the 4th Mexican Conf. on Pattern Recognition*. Huatulco, Mexico: Springer LNCS 7329, pp. 187–196 (cit. on p. 50).

Téllez, E. S., Chávez, E., and Camarena-Ibarrola, A. [2009]. A Brief Index for Proximity Searching. In: *Proc. of the 14th Iberoamerican Conf. on Pattern Recognition*. Guadalajara, Jalisco, Mexico: Springer LNCS 5856, pp. 529–536 (cit. on p. 53).

Tene, O. [2008]. What Google Knows: Privacy and Internet Search Engines. *Utah Law Review*, vol. 2008, no. 4, pp. 1433–1492 (cit. on p. 4).

Tepper, M., Musé, P., Almansa, A., and Mejail, M. [2011]. *Boruvka Meets Nearest Neighbors*. Tech. rep. HAL: hal-00583120 (cit. on p. 49).

Thomas, P. and Hawking, D. [Oct. 2009]. Server selection methods in personal metasearch: a comparative empirical study. *Information Retrieval*, vol. 12 (5), pp. 581–604 (cit. on pp. 7, 143).

Tibshirani, R., Walther, G., and Hastie, T. [2001]. Estimating the number of clusters in a dataset via the gap statistic. *Journal of the Royal Statistical Society: Series B*, vol. 63 (2), pp. 411–423 (cit. on p. 122).

Tigelaar, A. S., Hiemstra, D., and Trieschnigg, D. [May 2012]. Peer-to-Peer Information Retrieval: An Overview. *ACM Transactions on Information Systems*, vol. 30 (2), 9:1–9:34 (cit. on p. 61).

Traina Jr., C., Filho, R. F. S., Traina, A. J. M., Vieira, M. R., and Faloutsos, C. [Oct. 2007]. The Omni-family of all-purpose access methods: a simple and effective way to make similarity search more efficient. *The International Journal on Very Large Data Bases*, vol. 16 (4), pp. 483–505 (cit. on pp. 12, 39, 46).

Traina Jr., C., Traina, A. J. M., and Faloutsos, C. [2000a]. Distance Exponent: A New Concept for Selectivity Estimation in Metric Trees (Extended Abstract). In: *Proc. of the 16th Intl. Conf. on Data Engineering*. San Diego, CA, USA: IEEE, p. 195 (cit. on p. 36).

Traina Jr., C., Traina, A. J. M., Seeger, B., and Faloutsos, C. [2000]. Slim-Trees: High Performance Metric Trees Minimizing Overlap Between Nodes. In: *Proc. of the 7th Intl. Conf. on Extending Database Technology: Advances in Database Technology*. Konstanz, Germany: Springer LNCS 1777, pp. 51–65 (cit. on p. 48).

Traina Jr., C., Traina, A. J. M., Wu, L., and Faloutsos, C. [2000b]. Fast Feature Selection Using Fractal Dimension. In: *Proc. of the 15th Brazilian*

Symp. on Databases. Joao Pessoa, Paraiba, Brazil: Brazilian Computer Society, pp. 158–171 (cit. on p. 37).

Tsikrika, T., Popescu, A., and Kludas, J. [2011]. Overview of the Wikipedia Retrieval Task at ImageCLEF 2011. In: *CLEF 2011 Labs and Workshops, Notebook Papers*. Amsterdam, The Netherlands: *http://www.clef-initiative.eu/documents/71612/86377/CLEF2011wn-ImageCLEF-TsikrikaEt2011.pdf* (last visit: 4.10.2014) (cit. on pp. 4, 97).

Uhlmann, J. K. [Nov. 1991]. Satisfying general proximity/similarity queries with metric trees. *Information Processing Letters*, vol. 40 (4), pp. 175–179 (cit. on pp. 24, 30).

Verma, K., Sivashanmugam, K., Sheth, A., Patil, A., Oundhakar, S., and Miller, J. [Jan. 2005]. METEOR-S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services. *Journal of Information Technology and Management*, vol. 6 (1), pp. 17–39 (cit. on p. 149).

Vidal, E. [Jan. 1994]. New formulation and improvements of the Nearest-Neighbour Approximating and Eliminating Search Algorithm (AESA). *Pattern Recognition Letters*, vol. 15 (1), pp. 1–7 (cit. on p. 45).

Vidal-Ruiz, E. [July 1986]. An algorithm for finding nearest neighbours in (approximately) constant average time. *Pattern Recognition Letters*, vol. 4 (3), pp. 145–157 (cit. on p. 45).

Vieira, M. R., Traina Jr., C., Chino, F. J. T., and Traina, A. J. M. [Feb. 2010]. DBM-Tree: A Dynamic Metric Access Method Sensitive to Local Density Data. *Journal of Information and Data Management*, vol. 1, no. 1, pp. 111–127 (cit. on p. 47).

Vincenty, T. [Apr. 1975]. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey Review*, vol. 23, no. 176, pp. 88–93 (cit. on p. 112).

Vlachou, A., Doulkeridis, C., and Kotidis, Y. [2010]. Peer-to-Peer Similarity Search Based on M-Tree Indexing. In: *Proc. of the 15th Intl. Conf. on Database Systems for Advanced Applications – Part II*. Tsukuba, Japan: Springer LNCS 5982, pp. 269–275 (cit. on pp. 75 sq.).

Vlachou, A., Doulkeridis, C., and Kotidis, Y. [2012a]. Metric-Based Similarity Search in Unstructured Peer-to-Peer Systems. *Transactions on Large-Scale Data- and Knowledge-Centered Systems*, vol. 5, pp. 28–48 (cit. on pp. 75 sq.).

Vlachou, A., Doulkeridis, C., and Nørvåg, K. [Aug. 2012b]. Distributed top-k query processing by exploiting skyline summaries. *Distributed and Parallel Databases*, vol. 30, no. 3-4, pp. 239–271 (cit. on pp. 10, 62).

Vleugels, J. and Veltkamp, R. C. [1999]. Efficient Image Retrieval through Vantage Objects. In: *Proc. of the 3rd Intl. Conf. on Visual Informa-*

tion and Information Systems. Amsterdam, The Netherlands: Springer LNCS 1614, pp. 575–585 (cit. on p. 12).

Vu, Q. H., Lupu, M., and Wu, S. [2009]. SiMPSON: Efficient Similarity Search in Metric Spaces over P2P Structured Overlay Networks. In: *Proc. of the 15th Intl. Euro-Par Conf. on Parallel Processing*. Delft, The Netherlands: Springer LNCS 5704, pp. 498–510 (cit. on pp. 63, 68 sq.).

Wojna, A. [Nov. 2002]. Center-Based Indexing in Vector and Metric Spaces. *Fundamenta Informaticae*, vol. 56 (3), pp. 285–310 (cit. on p. 33).

Wu, L., Hua, X.-S., Yu, N., Ma, W.-Y., and Li, S. [May 2012]. Flickr Distance: A Relationship Measure for Visual Concepts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 5, pp. 863–875 (cit. on p. 23).

Wu, M., Zhu, F., Lv, J., Jiang, T., and Ying, J. [2009]. Improve Semantic Web Services Discovery through Similarity Search in Metric Space. In: *Proc. of the 3rd Intl. Symp. on Theoretical Aspects of Software Engineering*. Tianjin, China: IEEE, pp. 297–298 (cit. on p. 148).

Xu, H. and Agrafiotis, D. K. [Nov. 2003]. Nearest Neighbor Search in General Metric Spaces Using a Tree Data Structure with a Simple Heuristic. *Journal of Chemical Information and Computer Sciences*, vol. 43 (6), pp. 1933–1941 (cit. on p. 22).

Xu, W. and Miranker, D. P. [May 2004]. A metric model of amino acid substitution. *Bioinformatics*, vol. 20 (8), pp. 1214–1221 (cit. on pp. 19, 142).

Yan, M. and Ye, K. [Dec. 2007]. Determining the Number of Clusters Using the Weighted Gap Statistic. *Biometrics*, vol. 63 (4), pp. 1031–1037 (cit. on p. 122).

Yang, B. and Garcia-Molina, H. [2003]. Designing a Super-Peer Network. In: *Proc. of the 19th Intl. Conf. on Data Engineering*. Bangalore, India: IEEE, pp. 49–60 (cit. on p. 65).

Yao, A. C.-C. [1978]. On Random 2-3 Trees. *Acta Informatica*, vol. 9 (2), pp. 159–170 (cit. on p. 91).

Yianilos, P. N. [July 1998]. *Excluded Middle Vantage Point Forests for Nearest Neighbor Search*. Tech. rep. Princeton, NJ, USA: NEC Research Institute (cit. on p. 31).

Zezula, P., Amato, G., Dohnal, V., and Batko, M. [2006]. *Similarity Search: The Metric Space Approach*. Secaucus, NJ, USA: Springer New York, Inc. (cit. on pp. 2, 8, 16 sq., 19 sq., 24, 30 sqq., 38, 45, 51, 142, 152).

Zezula, P., Savino, P., Amato, G., and Rabitti, F. [Dec. 1998]. Approximate similarity retrieval with M-trees. *The International Journal on Very Large Data Bases*, vol. 7 (4), pp. 275–293 (cit. on p. 6).

Zhai, C. [Nov. 2008]. Statistical Language Models for Information Retrieval A Critical Review. *Foundations and Trends in Information Retrieval*, vol. 2 (3), pp. 137–213 (cit. on p. 22).

Zhang, Z. and Nasraoui, O. [Apr. 2009]. Profile-Based Focused Crawling for Social Media-Sharing Websites. *EURASIP Journal on Image and Video Processing*, vol. 2009. Article ID 856037, pp. 1–13 (cit. on p. 142).

Zhou, J. and Sander, J. [2003]. Data Bubbles for Non-Vector Data: Speeding-up Hierarchical Clustering in Arbitrary Metric Spaces. In: *Proc. of the 29th Intl. Conf. on Very Large Data Bases*. VLDB Endowment. Berlin, Germany, pp. 452–463 (cit. on pp. 41, 44).

Zobel, J. and Moffat, A. [July 2006]. Inverted Files for Text Search Engines. *ACM Computing Surveys*, vol. 38, no. 2, 6:1–6:56 (cit. on pp. 77, 97).

University
of Bamberg
Press

Ständig zunehmende Datenmengen und eine immer größer werdende Vielfalt an Datentypen in verschiedenen Anwendungskontexten erfordern sowohl skalierbare als auch flexible Indexierungs- und Suchtechniken. Metrische Zugriffsstrukturen (MAMs: metric access methods) können diese Flexibilität bieten, weil sie lediglich unterstellen, dass die Distanz zwischen zwei Datenobjekten durch eine Distanzmetrik modelliert wird. Darüber hinaus lassen sich skalierbare Lösungen mit Hilfe verteilter MAMs entwickeln. Sowohl IF4MI als auch RS4MI, die beide in dieser Arbeit vorgestellt werden, stellen metrische Zugriffsstrukturen dar. IF4MI gehört zur Gruppe der zentralisierten MAMs. Diese Zugriffsstruktur basiert auf einer invertierten Liste und repräsentiert daher eine hybride Indexstruktur, die neben einer inhaltsbasierten Ähnlichkeitssuche in beliebigen metrischen Räumen direkt auch Möglichkeiten der Textsuche unterstützt. Im Gegensatz zu IF4MI handelt es sich bei RS4MI um eine verteilte MAM, die auf Techniken der Ressourcenbeschreibung und -auswahl beruht. Dabei sind die Datenobjekte physisch verteilt. RS4MI ist jedoch keineswegs auf die Anwendung in einem bestimmten verteilten Information-Retrieval-System beschränkt. Verschiedene Anwendungsfelder sind für die Techniken zur Ressourcenbeschreibung und -auswahl denkbar, zum Beispiel im Bereich der Visuellen Analyse. Dabei gehen Anwendungsmöglichkeiten weit über den für die Arbeit unterstellten Anwendungskontext der inhaltsbasierten Bildsuche hinaus.

eISBN: 978-3-86309-311-2

www.uni-bamberg.de/ubp