DOI: 10.1111/cgf.14805

COMPUTER GRAPHICS *forum*
Volume 42 (2023), number 6, e14805

Visually Abstracting Event Sequences as Double Trees Enriched with Category-Based Comparison

Cedric Krause,[1] Shivam Agarwal,[1] Michael Burch[2] and Fabian Beck[1]

[1]University of Bamberg, Bamberg, Germany
fabian.beck@uni-bamberg.de
[2]University of Applied Sciences of the Grisons, Chur, Switzerland

Abstract
Event sequence visualization aids analysts in many domains to better understand and infer new insights from event data. Analysing behaviour before or after a certain event of interest is a common task in many scenarios. In this paper, we introduce, formally define, and position double trees *as a domain-agnostic tree visualization approach for this task. The visualization shows the sequences that led to the event of interest as a tree on the left, and those that followed on the right. Moreover, our approach enables users to create selections based on event attributes to interactively compare the events and sequences along colour-coded categories. We integrate the double tree and category-based comparison into a user interface for event sequence analysis. In three application examples, we show a diverse set of scenarios, covering short and long time spans, non-spatial and spatial events, human and artificial actors, to demonstrate the general applicability of the approach.*

Keywords: Visualization, Information Visualization

CCS Concepts: • Human-centred computing → Visualization techniques; Information visualization

1. Introduction

An event in data science and visualization corresponds to a particular happening, incidence or effect within a temporal sequence. Events are typically classified through event types. In many scenarios, the user has previous knowledge about events that are particularly important to answer certain analysis questions. Analysing biographies of historical figures, key private and professional events (*e.g*. marriage, academic degree) can act as important references and make different biographies comparable. For a soccer analyst, it might be important to look at what happened before a shot was taken, to steer the team's tactics. The event of interest can be considered as an anchor, and the analysis around it can yield insights into what led to the event or how it impacted the follow-up. For this purpose, we define and discuss *double trees* as a data structure focused around an anchor that abstracts preceding subsequences in a tree on the left, and succeeding subsequences in a tree on the right.

To counterbalance the data aggregation through the double trees, a group of visual comparison analysis tasks [PS16, GGJ*21] can re-introduce context and reveal relevant deviations and commonalities in the event sequences. For example, comparing researchers' careers by their nationality can give a better understanding if and how this attribute impacted the career progression. It is possible to compare event sequence data at various levels of granularity. Many techniques for the comparison of individual sequences (*e.g*. Refs. [WS09, GFL*20]) and mined patterns (*e.g*. Refs. [CXR18, PW14]) have been proposed. Prior research focusing on the comparison of sequence collections (*e.g*. Refs. [MDM*15, ZLD*15]), however, is rare and does not extend beyond two collections. Moreover, in event sequence comparison, information about absolute time and temporal gaps between events get lost. We aim to address these challenges by linking these observations about categories back to the attributes on the event or sequence level, while hinting at time differences between consecutive events.

We have developed an approach to analyse what happened before and after an event of interest, by using the double tree structure with an interactively defined anchor. It supports the comparison of up to around ten colour-coded categories based on a variety of event attributes. In Figure 1, the lives of physics Nobel Prize laureates are modelled as sequences of important events and visualized with our approach, aligned at the doctorate degree, which can be considered a major step in an academic's career. The example discerns the

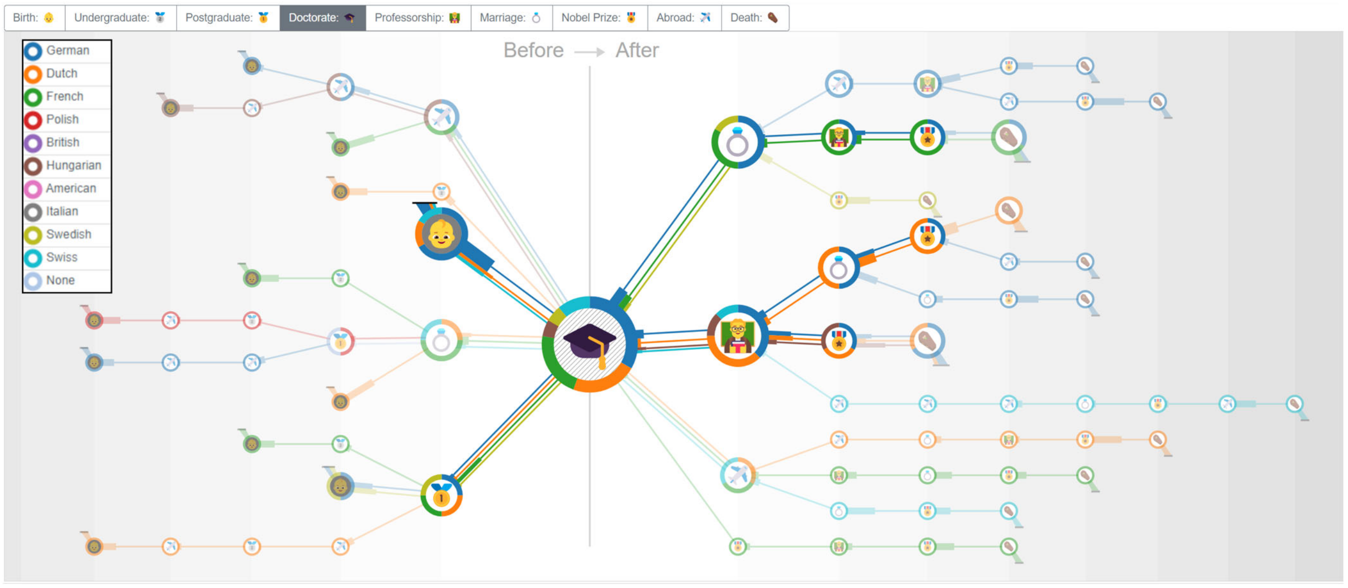

Figure 1: *Double tree visualization displaying life events of Physics Nobel Prize laureates. Subsequences before and after a user-selected anchor (here, the doctorate degree) are aggregated as trees. Colours support a comparison based on event attributes (here, nationalities).*

nationality of the scientists—as the colour-coded categories—to investigate regional differences. Our approach also shows time gaps between consecutive events as small bar charts on the edges, split by colour in case a comparison is activated. We complement this visualization with a detail view for inspecting individual sequences (Figure 5). Interactive selections allow for an iterative exploration and comparison of relevant subsets of event sequences.

In three application examples, we demonstrate how our novel approach can help answer common questions in event analytics. We look at biographical events for physics Nobel Prize laureates from the early 20th century, we compare in-game actions from the soccer World Cup final in 2018, and we analyse AI train routing models that challenged classical operations research techniques.

The paper, hence, introduces a new approach that leverages the double tree structure to aggregate event sequences around a central event of interest (anchor). It shows time differences between events and is enriched with visual event sequence comparison, where comparison categories stem from interactively selectable attributes of the events. The main contributions of this work include

- a formal definition and visual design of structuring event sequence collections as double trees,
- a formal definition and visual design of a category-based comparison technique for multivariate event sequences within the double trees,
- a prototypical implementation of this approach called *DTVis*, integrated into a user interface that adds functionality for filtering, selection and investigation of individual sequences, and
- application examples from diverse domains to exemplify the general applicability of the approach.

The implementation of *DTVis* is available on GitHub, and supplemental material contains a video showing its main features.

2. Related Work

Our approach tackles two of the tasks identified by Plaisant and Shneiderman [PS16]—the analysis of subsequences before and after an event of interest and comparing sets of sequences. In addressing these tasks, our approach shares similarities with other event sequence visualizations. Moreover, we find related approaches in visual comparison strategies for hierarchical data.

2.1. Event sequence visualization

Guo *et al.* [GGJ*21] have published a survey on visual analysis of event sequence data. Our approach fits into the following categories: *(1) data scale: subsequences and sequence collection*; *(2) automated sequence analysis: none*; *(3) visual representation: hierarchy-based* and *(4) interaction techniques: querying, alignment, emphasis and aggregation*. Out of the analysis tasks, our work concerns *explicit summarization* and the *comparison of sequence collections*.

Hierarchy-based event sequence visualizations have already been used for summarization. *Lifeflow* [WGP*11] displays the sequences hierarchically as icicle plots. It is also possible to align the sequences on one event type which leads to icicle plots before and after the alignment point, which corresponds to the same double tree data model that we use. Culy *et al.* [CL10] have also used the double tree structure in their work on words in context. However, their approach is limited to one level of depth on either side of the alignment point and only fans out the next level for selected children. *AcitiviTree* [VJC09] applied the same concept, but extended it to consider entire subsequences as alignment points, instead of single event types. Even though the aforementioned approaches make use of the same double tree data structure, they do not discuss it as a generalizable technique and, for instance, lack a formal definition and discussion of design alternatives. We discuss in depth a generalized, interactively adaptable version of the approach and, in addition, extend it with comparison capabilities regarding interactively defined categories based on event attributes.

Other techniques use a hierarchical visualization for event sequences, like *CoreFlow* [LKD*17] and *MAQUI* [LLMB18], and operate on large datasets with potentially thousands of event sequences. However, they follow a different approach and identify frequent patterns. Whereas this increases scalability, it comes at the cost of losing information about infrequent patterns and outliers, which would still be visible in our approach.

Regarding the comparison task, the survey by Guo *et al.* [GGJ*21] attests that there are only few event sequence comparison techniques at all [WS09, MDM*15, ZLD*15, QBW*20, GFL*20], none of which model the data hierarchically. The interaction technique of alignment is only used in one of these papers [WS09], but comparison is limited to individual sequences instead of aggregations. Our approach complements previous research by investigating comparison around an alignment point and considering aggregations of sequences. Most closely related might be approaches considering the comparison of sequence collections. *CoCo* [MDM*15] compares two collections guided by statistical metrics. The authors provide basic statistical summary metrics for the collections (*e.g.* difference in the number of sequences) to provide an overview and introduce different metrics on event sequences, temporal distributions and attributes. *MatrixWave* [ZLD*15] uses a different approach, where steps in the event sequences are represented by transition matrices. Multiple matrices are then concatenated in a zigzag line and encode differences between two categories in a colour gradient in the nodes. Our proposed approach is different from the aforementioned ones, as it can compare up to 10 collections instead of just two. Other event sequence comparison techniques (*e.g.* Refs. [WS09, GFL*20]) differ more fundamentally from our approach, since they do not compare collections of event sequences, but patterns or individual sequences. In our approach, we do not consider mined patterns; the individual sequence comparison is only a minor part in our approach, supported by the sequence list which is related to time-line-based comparison techniques (*e.g.* Wongsuphasawat and Shneiderman [WS09]). In addition to comparing multiple collections of sequences, our approach allows a more fine-grained comparison as it can consider different event-level categories within each event sequence, not only one category per event sequence.

For analysis of temporal differences and absolute time in event sequences, there are different variations. Some approaches (*e.g.* Du *et al.* [DPSS16]) extract the temporal information about events from the sequences themselves and focus on whether certain event types typically occur sooner or later. Other approaches focus on the time gaps between consecutive events, although the visual encodings vary between these techniques. While *MAQUI* [LLMB18] uses the placement on the x-axis to convey this information, other approaches (*e.g.* Refs. [WG12, GS14]) encode the time gaps into equidistant links by splitting them into a temporal and a connection part of the link. Our approach uses a similar representation of time differences between events. We further incorporate the visual encoding of time in both, the individual sequence list and the double tree. Moreover, the comparison aspect is extended to the temporal analysis explicitly, unlike in the existing techniques.

2.2. Visual tree and stream comparison

In this work, we compare event sequences within a hierarchical tree data structure and, hence, our approach relates to visual tree comparison methods. A survey by Graham and Kennedy [GK10] proposed a taxonomy based on the number of trees to compare. Our approach falls under the multiple tree comparison ($n > 2$) category of the taxonomy. Among the limited existing approaches enabling comparison of multiple hierarchies, juxtaposition is commonly used instead of a consolidated, single representation of a tree structure like in our approach. For instance, *BarcodeTree* [LZD*20] juxtaposes the condensed bar code tree representations, Beck *et al.* [BMW16] use a juxtaposed icicle plot representation of hierarchies and *TreeJuxtaposer* [MGT*03] is a *focus+context* approach to compare juxtaposed trees represented in a dendrogram layout. Another general difference of our approach to these tree comparison methods is that we can also compare event-specific categories (*i.e.* different event categories in one sequence).

Our aggregated tree representation of event sequences uses colours for comparison. Visually similar to *BaobabView* [EW11], we split the edges, use the width to encode the number of transitions between two event nodes, and colour them based on their category. However, unlike showing decision trees, our approach models and visualizes the aligned event sequences as two trees with a common root node. Although not showing trees but directed acyclic graphs, *Set Streams* [AB20] shares similarity with our approach but considers sequences of potentially overlapping set memberships, whereas events can only belong into one category. Their approach is limited to comparing two collections of sequences, but also works with colour-coded categories. Sankey-based event sequence visualizations (*e.g.* Refs. [PW14, CWM16]) enable comparison using colour-coded ribbons. While they focus on comparing the aggregated sequences as-is, our approach centres on enabling a condensed comparison of subsequences around a selected anchor event type.

3. Double Tree

Event sequences are, by definition, ordered temporally. Many scenarios include analysis tasks where the user focuses the analysis on a specific event type of interest and investigates the behaviour building up to it or following from it [PS16]. With double trees, we align all event sequences on this specific event type as the anchor—making it a natural focal point in the visualization as well—and aggregate what happened before and after into branches, without losing track of time differences between consecutive events.

The double tree aggregates sequences to enhance the visual scalability. When designing them, we considered different aggregation techniques. Most notably, we rejected directed acyclic graphs as an alternative. Directed acyclic graphs lose information about the provenance of their nodes, as it is not possible to unambiguously trace back its path to the root (which is the focus of our analysis). Interactions in a double tree are easier to implement and use; clicking a node can select unambiguously the branch up to the root. In a directed acyclic graph, there might be multiple paths to the root, and selecting all of them could be unintentional. Moreover, the construction of directed acyclic graphs requires additional parameters, which might vary between scenarios.

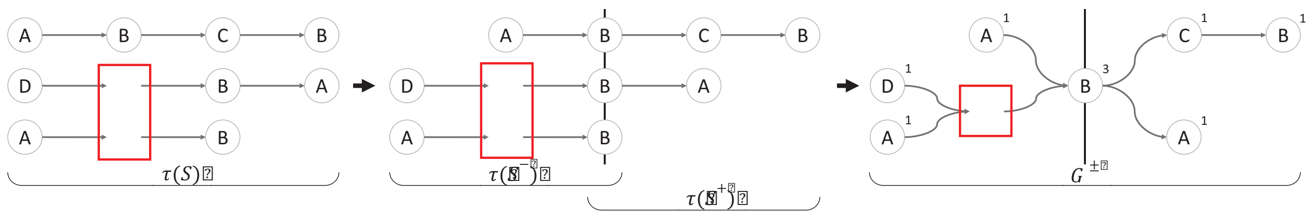

Figure 2: *From a collection of event sequences, we derive sequences of event types $\tau(S)$, align all sequences at the anchor (here,* B*) and create postfix and prefix trees from the subsequences $\tau(S^-)$ and $\tau(S^+)$. Merging the trees at their roots results in the double tree $G^\pm$ with weights $w(v)$ shown at the top right of each node. The red rectangles indicate in this example the two events of type* C *(before* B*), finally merged in the same tree node.*

3.1. Formal definition

The basic data structures that are subject to our analysis are ordered sequences of events. Our dataset is a collection of event sequences $C := (S_1, \ldots, S_n)$, with $S_i := (e_i^{(1)}, e_i^{(2)}, \ldots)$ where n is the number of sequences in the dataset, and the number of events in any sequence can vary. Each event $e \in E$ has a particular event type $\tau(e) \in T$. In our model, we always focus on one event type $\tau_c \in T$ as the anchor and consider all sequences that contain it. If there are multiple events of the anchor's event type in as sequence, the first event will act as the anchor event e_i^* in this sequence S_i. All sequences that contain the anchor will then be cut in two pieces, with S_i^- consisting of the subsequence up to and S_i^+ consisting of the subsequence starting from the anchor event.

From these subsequences, we construct two trees of event types: a postfix tree for the sequences before the anchor and a prefix tree for those after. The process is displayed visually in Figure 2. To this end, from a sequence of events S, we derive a sequence of event types $\tau(S) = (\tau(e^{(1)}), \tau(e^{(2)}), \ldots)$, cut at the anchor event into sequences $\tau(S^-)$ and $\tau(S^+)$. We use these sequences of event types to construct the postfix tree for all $\tau(S_i^-)$ and prefix tree for all $\tau(S_i^+)$. Formally, we can model these trees as directed graphs G^- (postfix tree) and G^+ (prefix tree). Both have the same root v_r, and child vertices are identified by their subsequence up to the root.

Combining the postfix and prefix tree at both of their roots results in what we consider a *double tree* $G^\pm = (G^-, G^+)$. With the postfix tree on the left and the prefix tree on the right, time is read from left to right in the double tree.

Each vertex v in these trees represents a set of events of the same type that are at the start (if v is in G^-) or the end (if v is in G^+) of a subsequence to the root (red rectangles in Figure 2). We can assign a weight $w(v) = |v|$ to each vertex v counting the number of events it represents. Directed edges $d_{v,v}$ $:= \{(e, e\ ) \in v \times v\ | \exists i$ with $S_i = (\ldots, e, e\ , \ldots)\}$ are sets of tuples of consecutive events that are contained in the sets v and v , respectively. The weight of an edge is simply $w(d) = |d|$.

Moreover, each event has a timestamp $t(e)$ that determines its position in the sequence. The time difference between two events is defined as $\delta(e, e\ ) = t(e\ ) - t(e) > 0$ in a sequence $S_i = (\ldots, e, e\ , \ldots)$. Along similar lines, we define the time difference $\delta(v, v\ )$ between two consecutive vertices as the average time difference between all consecutive events that are contained in v and v , respectively, and occurred in the same original sequence.

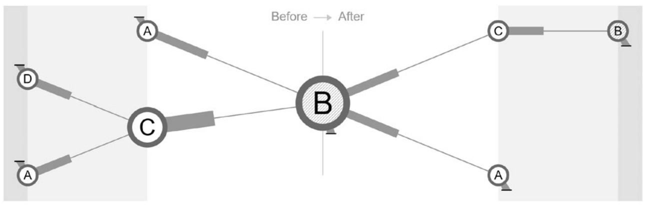

Figure 3: *Double tree, without active categories, with the same data as in Figure 2. Time differences between all events are equal.*

3.2. Layout and visual encoding

We visualize the double tree data structure $G^\pm$ described above as a node-link diagram. We first explain its base layout and main visual encodings as shown in Figure 3 without any highlighting of categories and selections.

The tree layout is based on the tidy tree algorithm [RT81] with adjustments to the position of the root node v_r to fit the trees on both sides. The single node at the centre is the merged root and represents the currently selected anchor τ_c. A vertical line separates the double tree into events that occurred before the anchor (left, G^-) and after (right, G^+). Sequences that do not contain an event of the anchor's event type are not displayed.

Sibling nodes in trees—at least when being drawn—have an order (here, vertically arranged from top to bottom). To reflect this order in the vertical arrangement of the tree branches, sequences that appear early in the collection tend to be placed towards the top of the double tree visualization. Figure 3 shows this arrangement, as the first exemplary event sequence `A,B,C,B` is connected by the topmost links. However, sequences later in the dataset might extend an already existing branch at the top of the double tree. Wherever the new and existing sequences diverge, the existing sequence will be placed above the new one.

Each node v represents a set of individual events that share the same subsequence to the root node v_r. We encode the number of represented events $w(v)$ in the nodes' area, constrained with a minimum and maximum to improve legibility. The way we merge

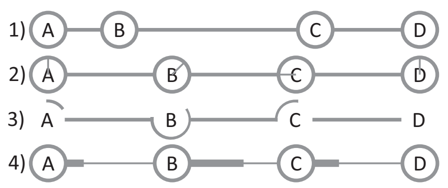

Figure 4: *Design alternatives to encode temporal extension between events: (1) position on the* x*-axis, (2) clock metaphor, (3) ring segments, (4) link length (selected alternative).*

sequences leads to trees where nodes closer to the centre represent more events. To account for this, we do not use equal distances for each horizontal layer. Instead, nodes close to the centre occupy more space. We compound this design choice by using a shaded background that is darker on the outer sides depending on the depth of trees, guiding the users' attention towards the lighter, higher-contrast centre of the visualization.

The common type of the events represented by each node is mapped to a visual identifier. If possible, we use application-specific icons so that the corresponding event types of the nodes in the tree can be quickly understood without sacrificing a lot of screen space. If no icons can properly represent the event types, a single character or short string (*e.g.* the initials of the event type names) can be used as a fallback. In both cases, this encoding is also explained in a legend at the top of the view (Figure 1).

The root node v_r marks the first event of the selected event type in a sequence as the anchor τ_c. A node of the same event type might re-appear in the tree G^+ on the right, which could be of specific relevance for analysis. To allow quick retrieval of these events with the selected event type, we mark all nodes of that type and apply a hatch pattern to their otherwise unfilled inner area (*e.g.* nodes `B` in Figure 3). All other event types can also occur multiple times on both sides of the double tree. To quickly identify all such cases, the user can hover a node—including the root node—which highlights all other nodes with the same event type.

The links $d_{v,v}$ in our tree connect the nodes and also encode two attributes of these transitions. First, the width encodes the weight $w(d)$ of the link. Hence, subsequences that occur in many event sequences have thicker links (Figure 3). Second, the average time differences $\delta(v, v\ )$ between the source node and the target node are encoded. By default, absolute time is displayed (*i.e.* longer link means more absolute time differences), but can be changed to relative time (*i.e.* longer link means longer share of the durations of the sequences). We have considered four design alternatives to convey these time differences within the links (Figure 4). Alternative *(1)* arranges the nodes along the *x*-axis according to their exact time differences, which has a big impact on the overall double tree layout and leads to issues when comparing the lengths (number of nodes) of different branches. The clock metaphor of alternative *(2)* causes occlusion with the event identifier. Moreover, it is problematic to interpret when the time difference is absolute, such that the pointers of most sequences would not complete a full circle, just like alternative *(3)* where the ring segments would not complete a full circle. All three of the aforementioned alternatives also suffer from difficulties in splitting the time difference into multiple categories, as it is necessary for category-based comparison. Hence, we decided to integrate alternative *(4)* into our visualization in which the time difference is encoded in the length of the link, and the link is then connected to the target node as a thin line. The time differences encoded like this hint at absolute time, but it remains an open challenge to make timestamps of specific events readable and comparable within the double tree.

The beginning of an event sequence is explicitly indicated by an incoming link, top-left of the corresponding node representing the first event. Conversely, when the link goes to the bottom right of the node, the event sequence ends with the event (*e.g.* Figure 3).

4. Category-Based Comparison

The double tree visualization allows us to investigate collections of event sequences around the anchor. Using the visualization with real data, we considered it natural to compare the sequence progressions of different collections, *e.g.* how different soccer teams are building up towards shots. Instead of creating multiple double trees side-by-side, we decided to leverage tree comparison techniques and integrated the comparison aspect into the double tree itself. We generalize this to select the comparison categories interactively, based on the attributes of the events.

Event attributes can be categorical (*e.g.* the person who executed the event), or numerical (*e.g.* the score or rating of the event). In some cases, these attributes are inherited from the sequence of events it belongs to (*e.g.* the whole event sequence is performed by the same person). To process and visualize event-level and sequence-level attributes in the same way, we assign the sequence level attribute to all events within that sequence. We utilize these attributes to define event categories that we can compare within the double tree.

4.1. Formal definition of categories

These event categories define a partition P of the events E, from which the individual event sequences are assembled, yielding a family of sets $P = \{E_1, E_2, \ldots, E_m\}$ with $\bigcup_i E_i = E$ and $E_i \cap E_j = \emptyset$ for $i = j$. This way $v \cap E_i$ is the set of events represented by vertex v that fall into the category of E_i. This partition can be used to subdivide the weight of nodes of the double tree $G^{\pm}$ into weights $w(v \cap E_1), w(v \cap E_2), \ldots, w(v \cap E_m)$ with $\sum_i w(v \cap E_i) = w(v)$. Each weight $w(v \cap E_i)$ summarizes the number of events from category E_i at the position in the sequence that corresponds to the position of v in the double tree.

Each directed edge in the graph is also subdivided into the categories. Edge $d_{v \cap E_i, v}$ represents the tuples of consecutive events that are contained in v the given partition E_i and v . Analogously, we can define the time difference of the categories for each link such that $\delta(v \cap E_i, v\ )$ yields the time difference from the source to the target when only considering the category E_i at the source.

In contrast to the event types, which are fixed and define the structure of the double tree, we use such event categories and resulting event partitions as a transient subdivision, interactively defined on

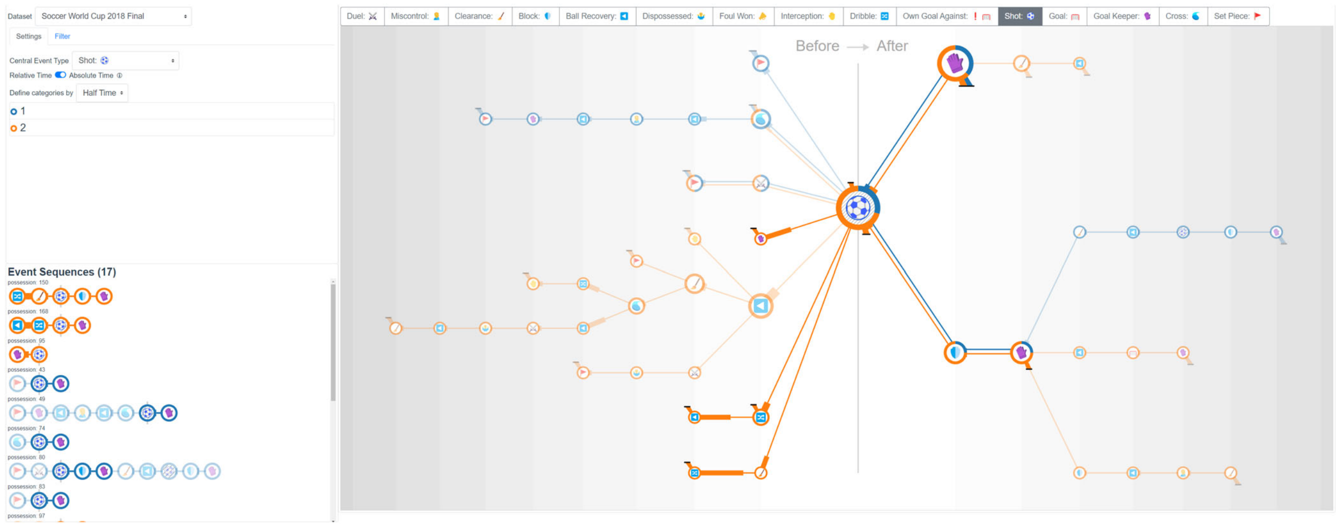
Dataset: Soccer World Cup 2018 Final
Settings Filter
Central Event Type: Shot
Relative Time / Absolute Time
Define categories by: Half Time
1
2
Event Sequences (17)

Figure 5: *The full interface with controls and raw event sequences on the left and the main tree visualization on the right. The data shows event sequences from the soccer World Cup final 2018 that led to shots.*

different attributes of the events. Changing the definition of the categories does not alter the structure of the double tree, but only the subdivision of weights of the nodes.

4.2. Adaptable visual comparison

We allow for the interactive choice of an attribute of the events that groups events into partitions. The user can select this attribute from which the categories are derived. This process is straightforward for categorical attributes. If the selected attribute is numerical, however, the partition can be derived from user-defined data ranges, which results in two categories *Over* and *Under or equal*. The temporal attribute that is used in the links can also be selected as a numerical attribute. Each category maps to a distinct colour as shown, for instance, in Figure 5, for two and in Figure 1, for 10 categories. All visual elements representing a category can be hovered, which will highlight all other visual elements representing the same category (Figure 10, middle).

The relative sizes of these categories of events are visualized as ring fractions of the respective colours around the nodes in the double tree. The angles of the fractions correspond to the weight of the categories $w(v \cap E_i)$ in relation to the total weight of the node $w(v)$. The links also split up, where the colour is determined by the source node in the specific sequence, since we read all links from left to right. The widths of these coloured links display the weights of the links $w(v \cap E_i)$ given the category represented by the same colour. The lengths of the links are also adjusted to show the average time differences for the respective category $\delta(v \cap E_i, v\ )$, which allows for their comparison across the selected categories for each link.

5. DTVis

We implemented the design discussed above in a tool called *DTVis*. The single-page web application is written in Typescript and uses the Vue.js framework and D3.js [BOH11]. The application is heavily modularized into components, to increase reusability and extensibility. Transformers from the file structures used in our datasets to the internal data model of *DTVis* exist and can be taken as a template to add new transformers for other data sources. All source code and data are publicly available on GitHub at https://github.com/vis-uni-bamberg/event-sequence-double-trees.

Within *DTVis*, we integrate the double tree visualization enriched with category-based comparison into a user interface (Figure 5). A control panel on the top left allows determining the event type of the anchor and defining the categories for comparison of events and sequences. The user can also select whether the time difference encoded in the length of the links should be absolute or relative to the duration of the sequence they occur in. The individual sequences contained in the double tree are shown in the sequence list at the bottom left. Ensuring consistency, the time differences and categorical information encoded in the links are the same as in the double tree visualization, and hovering the sequence in the list will highlight the respective branches of the double tree (Figure 9).

Depending on the dataset, the double tree might get complex, but further filtering of event sequences can help to focus on a specific aspect and simplify. We incorporate multiple interaction techniques to set the focus of the analysis and to inspect details of event sequences. Through a query builder, the user can provide a subsequence of event types, including wildcards in between. Sequences have to match the query to be displayed, which can greatly reduce the size of the double tree.

In the double tree visualization, the user can highlight branches by clicking on the nodes. The highlighting is propagated to the list of sequences, where the corresponding subsequences are also highlighted (Figure 5). Multiple branches can be highlighted on the same side of the double tree. In this case, no single event sequence can satisfy all of them. Hence, the individual sequences are highlighted if they match any of the mutually exclusive branches. It is also possible to highlight branches on both sides of the double tree. The individual sequences in the sequence list are sorted by the degree to which they align with the highlighted branches in the double tree. At the top are those sequences that contain highlighted branches on both sides, followed by those matched only on the left and by those matched only on the right.

6. Application Examples

To demonstrate our approach, we apply it to three domains, namely, biographical data, soccer event data and routing data from simulated trains. This set of scenarios is diverse regarding the length of sequences and time spans, contains spatial and non-spacial events, and events from human and artificial actors. Hence, the scenarios highlight the general applicability of our approach.

6.1. Biographies of physics nobel prize laureates

For our first application example, we manually collected biographical data from public sources about winners of the Nobel Prize in physics between 1901 and 1921. Each scientist represents one event sequence to a total of 25. We chose nationality and age of the person at the time of the event as attributes. Both of them are on the event level. The event types we collected range from their birth, over academic degrees, the Nobel Prize, to their death; they can be seen in the legend in Figure 1.

The prize: When selecting the Nobel Prize 🎖 as the anchor, we observe that most of the winners did not have any events between winning the Nobel Prize and their death ⚰, even though the link shows that the average time between those events was rather large. Those scientists who had events that followed were comparatively young when receiving their Nobel Prize. Two of them only started their first professorship 🧑‍🏫 after being awarded the Nobel Prize. Selecting the professorship event type to the right of the root (Figure 6), we see in the sequence list that one of them is Marie Curie, who, as a woman, was denied university admission in her home country, and only took over the professorship of her husband after his death. The other person to receive his first professorship after his Nobel Prize is Lawrence Bragg, who was 25 years old when he was distinguished with the award together with his father.

The doctorate degree: To investigate when scientists have received a doctorate degree 🎓, we select it as the anchor (Figure 1). If we define the categories by nationality of the person at the time of the event, we quickly see that only people from six different countries received a doctorates degree at all, even though the dataset contains people of 10 countries. Most prominently, no *British* laureate acquired a doctorate degree, which might be surprising since they are tied with *Germany* for most Nobel Prizes won in the selected time range. The historical reason is that the PhD degree was only adopted in 1917 in the United Kingdom.

The first professorship: If we are more interested in the career paths of the scientists, we can select the first professorship 🧑‍🏫 as the anchor (Figure 7). We observe that multiple scientists went abroad ✈ as the event before gaining their professorship and highlight that node. Through the short time differences shown in the links, we can see that these five scientists received their professorship shortly after

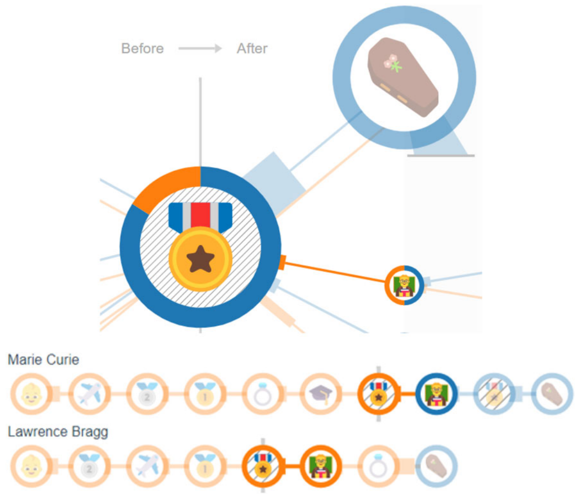

Figure 6: *Marie Curie and Lawrence Bragg have not been professors before winning the Nobel Prize. Orange means the persons were 35 years or younger, while blue means they were older.*

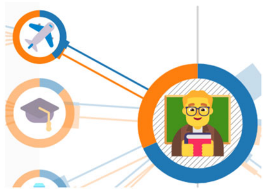

Figure 7: *Out of all 22 scientists who received a professorship, five of them went abroad shortly before their appointment.*

emigration, indicating that this might have been a career accelerator or the reason to move abroad. We can define categories by the age and interact with the slider to select the threshold for the *Over* and *Under or equal* categories, which confirms that between 0 and 2 years laid between the events.

6.2. Key actions in soccer matches

The next domain concerns event data from a soccer match. We look at the final of the FIFA World Cup 2018 between *France* and *Croatia*. The data are taken from Statsbomb [Sta] and contains about 3500 events like fouls, shots or dribbles. To reduce the size of the dataset, we excluded several event types that distorted the double tree due to their frequency. The two most notable event types we removed are passes and carries, which are the most common event types in the data. Furthermore, we have promoted some subtypes to regular event types where meaningful (*e.g.* main type: shot, subtype: goal). The sequences in this dataset are the different ball possessions, which change whenever the opposite team acquires the ball or the ball goes out of play. Some attributes we collected are on the event level (*e.g.* players) and others are on the sequence level (*e.g.* halftime).

Set pieces: In the match, there have been a combined 25 corners and free kicks. These events always start a new event sequence;

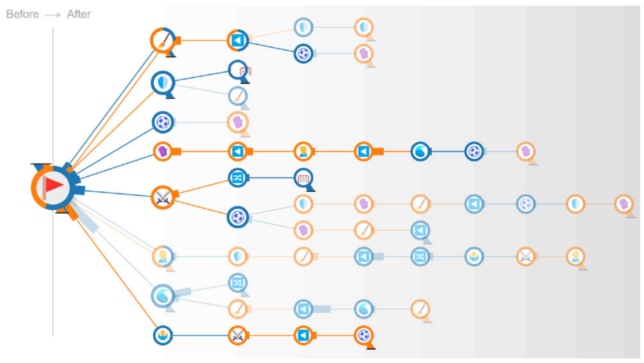

Figure 8: *Sequences from the soccer World Cup 2018 final that started with a set piece (corners or free kicks). All branches that led to a shot (missed), goal or own goal are highlighted. Events are orange for* France *and blue for* Croatia*.*

hence, we see that the double tree in Figure 8 only has branches on the right side. From the ring fraction of the root node, we see that *France* had almost as many events of the set piece event type as *Croatia*. This is information we might typically see in any basic scoreboard of the match. However, if we look more specifically into the sequences and check which of these sequences ended in a shot, goal or own goal !, we see that, of the original 25 sequences, only nine remain (which are highlighted in Figure 8). Looking in more detail, we observe that *Croatia* has also scored an own goal in one of the highlighted sequences. After the set piece from *France*, a failed block attempt, led to the own goal, which we can also draw from the visualization. While analysing sequences from one game and one type of attack only can be anecdotal, it demonstrates the additional information our approach provides over widely used statistical match summaries.

Shots in each halftime: Our visualization can quickly show us that the offensiveness of the teams was entirely different in the *first half* compared to the *second half* of the match. Figure 5 displays the sequences that led to shots from either team, and the coloured categories indicate whether the sequences happened in the first or second half of the match. While the subsequences before the shot in the *second half* show events from regular play, almost all the shots in the *first half* resulted from a set piece. Investigating the links (absolute time) on the left side of the double tree, we also notice that the sequences from the *first half* occurred in quick succession—as we would expect for a set piece. The sequences from the *second half* have much longer links (*e.g.* those selected in Figure 5), indicating larger time differences between the events, and were likely resulting from a slower, more controlled setup of attacks.

6.3. Routing of simulated trains

Flatland [MNL*20] is a virtual environment to promote research addressing the generic vehicle re-scheduling problem [LMB07]. Each map consists of railway tracks and trains. The trains originate near stations, can only move forward on tracks, wait at their position, might freeze for a few time steps due to random malfunctions, and end once they reach their destination station. The goal is to schedule all trains to reach their destination in the shortest time. The top submissions of the competition organized for the *NeurIPS 2020* conference [LSS*21] were an operations research (OR) technique, followed by three solutions based on reinforcement learning (RL). We selected `Level 20 Map 3` from the competition dataset [The21], with 14 stations and 98 trains to schedule. Using an existing visual analytics tool [AWWB22], we defined 13 regions-of-interest based on their importance in the structure of the rail network (*R1–R13* in Figure 9, left; labelled *1–13* in all visualizations). *R6* (a single-cell region) is an important junction of tracks from many directions, *R9* and *R10* are parallel tracks for trains transitioning between *R8* and *R6*. The sequences in this dataset are the individual trains for each scheduling approach, and the events are traversals of marked regions. Attributes are the approach that scheduled this train (sequence level) and the time step of the region's traversal (event level).

Waiting on tracks: With the links in the double tree, we investigate the elapsed time for trains going from one region to another. Anchoring the double tree on *R13*, we focus on the links from *R12*, on the bottom left, up to the root (Figure 9, right). The widths of the coloured links suggest that *OR_old_driver* scheduled a higher number of trains between these two regions. The time difference encoded in the red links shows that the trains, on average, took more time steps (absolute time) between the two regions than those of other scheduling techniques. Although regions *R12* and *R13* are nearby, there are many parallel tracks between them. The trains might have waited on the parallel tracks to allow oncoming trains, which could have led to a higher time to reach *R13*.

Revisiting a region: Trains visiting a region more than once can indicate inefficient planning. To explore re-visits, we focus on region *R13*, set it as the anchor and select its repeated occurrence in the tree on the right. In Figure 9 (right), we see two branches with repeated visits to the region ($R13 \rightarrow R4 \rightarrow R13$ and $R13 \rightarrow R12 \rightarrow R13$). Several trains, scheduled by all techniques except *RL_netcetera*, visited the region *R13* more than once (absence of orange coloured ring segment in the *R13* nodes on the right side of the double tree). Investigating further to understand the revisits, we select regions *R3* (on the left) and *R12* (on the right) to order the sequence list, as shown in Figure 10 (right). Hovering over the sequence $R3 \rightarrow R4 \rightarrow R13 \rightarrow R12 \rightarrow R13 \rightarrow R4 \rightarrow R3$ in the list, the path gets highlighted in the double tree (Figure 9, right). The sequence implies a train moving east in *R3* passed through *R4* and *R13*, was looped around using structural layout in *R12* to change the direction of movement, revisiting *R13* and finally *R3*, but now moving west. Since the trains cannot move backwards, the cyclic manoeuvre might have been necessary to change the direction of a train. *OR_old_driver* exhibited this specific manoeuvre twice (first two rows in Figure 10, right). The next two sequences in the list show a similar behaviour by trains of *RL_marmot* that circle from region *R12* over *R13* and *R4* to *R11*, while trains could have moved directly from *R12* to *R11* by moving west.

Uni-directional usage of parallel tracks: Next, we compare the usage of two parallel tracks, *R9* and *R10* between regions *R8* and *R6*. Out of the two connected regions, we choose *R6* as the anchor and select all occurrences of *R8* on both sides of the double tree (see, Figure 10, middle). Hovering over the red coloured ring segment of *OR_old_driver*, we infer that it scheduled all trains from *R8* going to *R6* only via *R10* (no red coloured ring segment in $R8 \rightarrow R9 \rightarrow R6$;

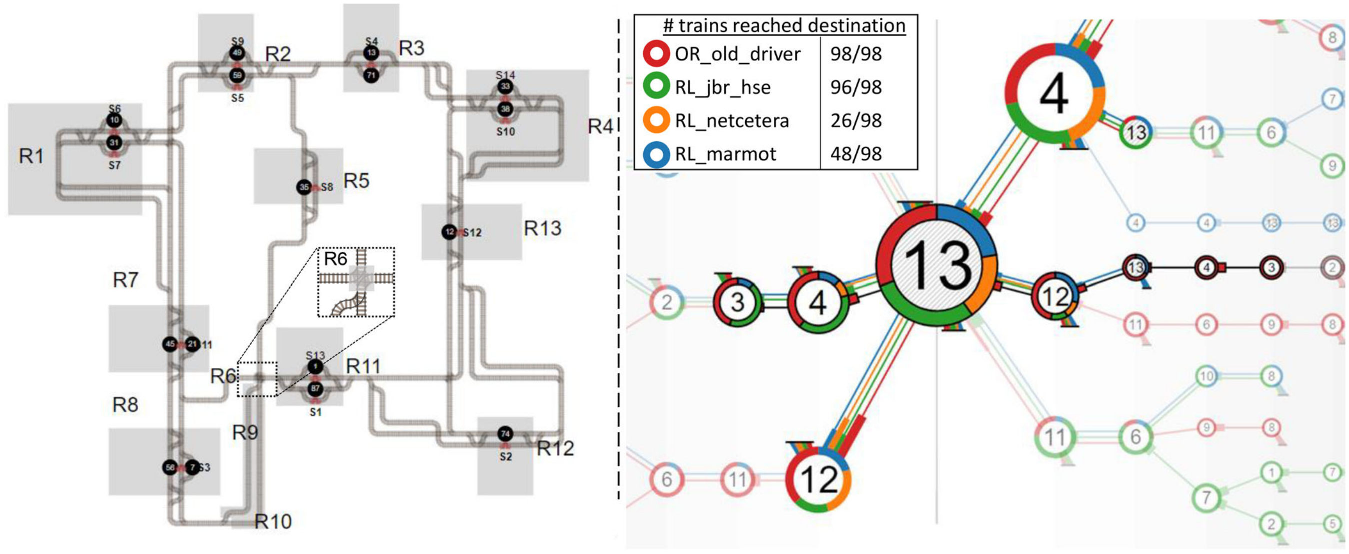

Figure 9: *Flatland map with regions-of-interest* R1–R13 *in grey and trains as black circles (left). Filtered and selected event sequences to show repeated visits in* R13 *from* R3 *and* R12*, comparing the performance of the four scheduling techniques (right).*

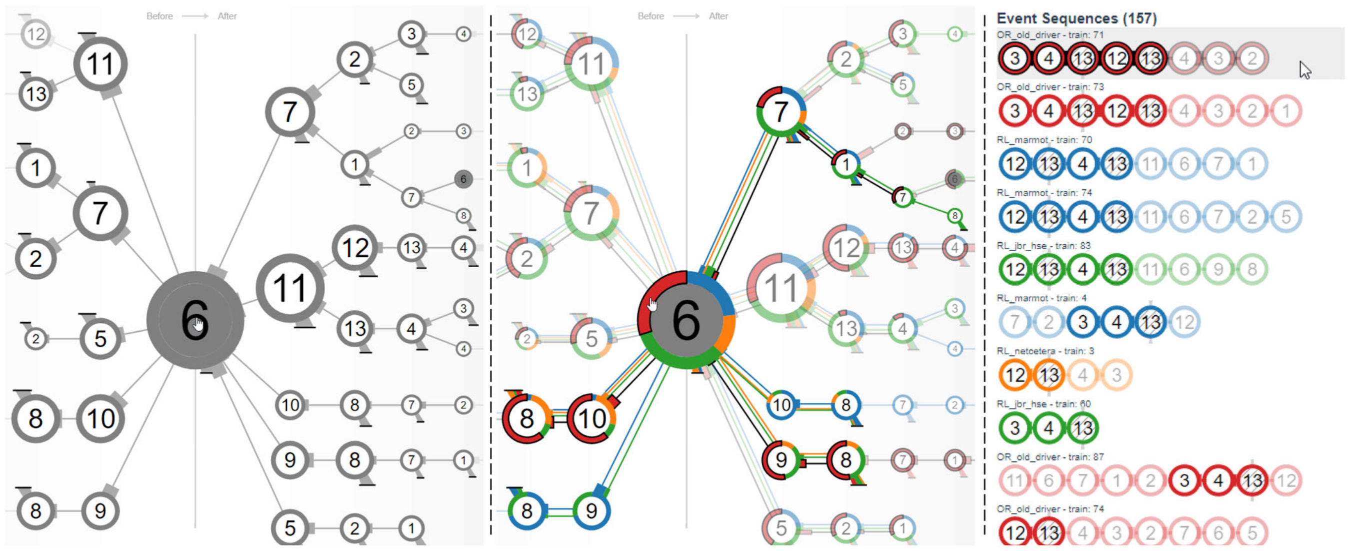

Figure 10: *Sequences showing transition of trains through the region* R6 *in Flatland (left). Comparing the sequences based on scheduling techniques (red is highlighted with a black border because it is hovered) and focusing on the movement of trains between two regions* R8 *and* R6 *(middle). The list of sequences re-visiting region* R13 *and hovering over the first row (right).*

left side of the double tree). Additionally, *OR_old_driver* scheduled all trains from *R6* to *R8* to transit via *R9*, suggesting exclusive usage of parallel tracks for opposite directions to avoid head-on collisions. *RL_marmot* also exhibits the strategy, while *RL_jbr_hse* moves trains on the tracks from both directions (green colour in all nodes of *R9* and *R10* between *R6* and *R8*), and *RL_netcetera* uses only one of the two tracks (*R10*) exclusively.

Busy junction: To investigate routes around the junction in *R6*, we select it as the anchor in the double tree. From Figure 10 (left), we observe many links on both sides of *R6*. It indicates that trains are coming in and going out in all four directions—east (*R7*), north (*R5*), west (*R11*) and south (*R9* and *R10*). Hovering the node of the anchor (*R6*), we see its repeated occurrence in the sequence *R6* → *R7* → *R1* → *R7* → *R6* highlighted on the right side of the double tree. Repeated visits by trains to an important junction could indicate inefficient planning, as revisiting junction points could stall the traffic. We also observe the orange outgoing edge at the bottom right of the node of region *R6*, indicating that trains scheduled by *RL_netcetera* stayed on top of the junction or on connecting tracks towards other regions. This insight indicates possible reasons behind the bad performance of *RL_netcetera* (only 26 of 98 trains reached their destination).

7. Limitations and Open Research Questions

Our approach allows focusing analysis on a user-defined anchor and comparing event sequences based on categories. As shown in the application examples, this already supports various relevant use cases and provides meaningful insights. We tried to keep the data model simple so that the approach stays broadly applicable and the design of the visualization will not become overly complex. However, some assumptions regarding the data also restrict us in supporting certain analyses. Moreover, the application examples also revealed other limitations of the approach. We discuss these limiting factors and translate them to open research questions that can inspire future research.

How to better handle numeric attributes, without transforming them into discrete and unordered categories? Numeric attributes of the events can be used in our approach to define categories for comparison. This converts the often continuous numeric scale into a discrete one and ignores the inherent order of the resulting discrete categories. This is acceptable often (*e.g.* focusing on the last 15 min of a soccer match), but can be considered a limitation of our approach in other cases. Among the authors, we have discussed different designs for visualizing such numeric data in addition to categories, but possible solutions would make the diagram significantly more complex and difficult to interpret. Showing one additional numeric value per node is not an issue, however, in our case, we would need to show a distribution of numeric values, split according to the interactively defined categories to be compatible with the category-based comparison. Displaying time differences is a step in the direction of visualizing numeric attributes. However, while the temporal differences fit on the links as it describes a characteristic of the transition, other numerical attributes would require different encodings.

How to support the visualization and comparison of multiple categorical attributes simultaneously? The interactive definition of categories allows for quickly switching between different aspects of comparison. In the application examples, we already experienced that, in some cases, it would also be interesting to see the categorical information about two or more different aspects at the same time. For instance, it might be relevant to analyse a soccer match for by categories of *team* and *first/second half* to identify, for instance, that one team adopted the strategy of the other team in the second half. If both categories are binary, like in this example, we could translate them into a new categorical variable with four cases. However, this approach does not scale to handle many more variables and categories, as the number of possible combinations grows quickly. Hand-picking interesting combinations and grouping everything else in an *Other* category could potentially help with this issue. Generally, the limiting factor here is the use of colours, which restricts the comparison to about 10 categories. While this is a substantial gain over the maximum of two categories as in prior research, support for even higher numbers of categories would require other channels and means to express category memberships.

How to simplify and aggregate the tree to support longer sequences of events and larger sets of sequences? Currently, our approach is limited to event sequences consisting of up to about 30 events. If trying to show more, the horizontal space for each layer would become too small. A straightforward solution would be to cut non-branching sequences, making them explorable on demand, or implementing horizontal panning where the view always starts at the anchor, which would be the focus of most analysis tasks. Integrating pattern mining techniques and visualizing the patterns would reduce the length of sequences, however, would come at the cost of significantly decreased interpretability. If not only the lengths of the event sequences grow but also more and more distinct sequences shall be considered, the complexity of the tree structure itself might also become too high. We have added the query builder to partially tackle this challenge, but it would need further capabilities for vastly greater numbers of distinct sequences. On the other hand, we could allow optional alternatives in the sequences to better aggregate the sequences but would transform the tree data structure into a directed acyclic graph, which leads to the issues we discussed in Section 3. Introducing focus-and-context techniques (*e.g.* a data lens [TGK*17]) might further improve the visual scalability. However, if aiming at analysis of event sequences at a large scale, an additional overview visualization becomes necessary, potentially with the double trees acting as an intermediate representation for subsets of event sequences.

8. Conclusion

We proposed double trees as a visualization approach for abstracting collections of event sequences focused around a user-selected event of interest. This interactively set focus of analysis makes the approach versatile regarding the studied analysis questions and, at the same time, reduces the complexity of the visual representations to an interpretable higher-level abstraction. We extended the double trees with interactive category-based comparison of colour-coded categories of events and event sequences. We assume that the events in the sequences carry type information and can be further categorized based on other attributes and meta-data. As demonstrated in our application examples, such data are relevant in different use cases across various domains.

Acknowledgements

This work is partly funded by *Mercator Research Center Ruhr* (*MERCUR*; project: “Vergleichende Analyse dynamischer Netzwerkstrukturen im Zusammenspiel statistischer und visueller Methoden”).

Open access funding enabled and organized by Projekt DEAL.

References

[AB20] Agarwal S., Beck F.: Set Streams: Visual exploration of dynamic overlapping sets. *Computer Graphics Forum 39*, 3 (2020), 383–391.

[AWWB22] Agarwal S., Wallner G., Watson J., Beck F.: Spatio-temporal analysis of multi-agent scheduling behaviors on fixed-track networks. In *Proceedings of the IEEE Pacific Visualization Symposium* (2022), pp. 21–30. https://doi.org/10.1109/PacificVis53943.2022.00011

[BMW16] BECK F., MELCHER J., WEISKOPF D.: Identifying modularization patterns by visual comparison of multiple hierarchies. In *Proceedings of the 2016 IEEE 24th International Conference on Program Comprehension* (2016), IEEE, pp. 1–10. https://doi.org/10.1109/ICPC.2016.7503712

[BOH11] BOSTOCK M., OGIEVETSKY V., HEER J.: D3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics 17*, (2011), 2301–2309.

[CL10] CULY C., LYDING V.: Double Tree: An advanced KWIC visualization for expert users. In *Proceedings of the 14th International Conference Information Visualisation* (2010), pp. 98–103. https://doi.org/10.1109/IV.2010.24

[CWM16] CHOU J.-K., WANG Y., MA K.-L.: Privacy preserving event sequence data visualization using a Sankey diagram-like representation. In *Proceedings of the SIGGRAPH ASIA 2016 Symposium on Visualization* (2016), ACM. https://doi.org/10.1145/3002151.3002153

[CXR18] CHEN Y., XU P., REN L.: Sequence synopsis: Optimize visual summary of temporal event data. *IEEE Transactions on Visualization and Computer Graphics 24* (2018), 45–55.

[DPSS16] DU F., PLAISANT C., SPRING N., SHNEIDERMAN B.: EventAction: Visual analytics for temporal event sequence recommendation. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology* (2016), pp. 61–70. https://doi.org/10.1109/VAST.2016.7883512

[EW11] ELZEN S. V. D., WIJK J. J. V.: Baobab-View: Interactive construction and analysis of decision trees. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology* (2011), pp. 151–160. https://doi.org/10.1109/VAST.2011.6102453

[GFL*20] GUO R., FUJIWARA T., LI Y., LIMA K. M., SEN S., TRAN N. K., MA K. L.: Comparative visual analytics for assessing medical records with sequence embedding. *Visual Informatics 4* (2020), 72–85.

[GGJ*21] GUO Y., GUO S., JIN Z., KAUL S., GOTZ D., CAO N.: Survey on visual analysis of event sequence data. *IEEE Transactions on Visualization and Computer Graphics* (2021). https://doi.org/10.1109/TVCG.2021.3100413

[GK10] GRAHAM M., KENNEDY J.: A survey of multiple tree visualisation. *Information Visualization 9*, 4 (2010), 235–252.

[GS14] GOTZ D., STAVROPOULOS H.: Decisionflow: Visual analytics for high-dimensional temporal event sequence data. *IEEE Transactions on Visualization and Computer Graphics 20* (2014), 1783–1792.

[LKD*17] LIU Z., KERR B., DONTCHEVA M., GROVER J., HOFFMAN M., WILSON A.: CoreFlow: Extracting and visualizing branching patterns from event sequences. *Computer Graphics Forum 36*, 3 (2017), 527–538.

[LLMB18] LAW P.-M., LIU Z., MALIK S., BASOLE R. C.: MAQUI: Interweaving queries and pattern mining for recursive event sequence exploration. *IEEE Transactions on Visualization and Computer Graphics 25* (2018), 396–406.

[LMB07] LI J.-Q., MIRCHANDANI P. B., BORENSTEIN D.: The vehicle rescheduling problem: Model and algorithms. *Networks 50*, 3 (2007), 211–229.

[LSS*21] LAURENT F., SCHNEIDER M., SCHELLER C., WATSON J., LI J., CHEN Z., ZHENG Y., CHAN S.-H., MAKHNEV K., SVIDCHENKO O., EGOROV V., IVANOV D., SHPILMAN A., SPIROVSKA E., TANEVSKI O., NIKOV A., GRUNDER R., GALEVSKI D., MITROVSKI J., SARTORETTI G., LUO Z., DAMANI M., BHATTACHARYA N., AGARWAL S., EGLI A., NYGREN E., MOHANTY S.: Flatland competition 2020: MAPF and MARL for efficient train coordination on a grid world. In *NeurIPS 2020 Competition and Demonstration Track, Proceedings of Machine Learning Research* (2021), vol. *133*, pp. 275–301. https://proceedings.mlr.press/v133/laurent21a.html

[LZD*20] LI G., ZHANG Y., DONG Y., LIANG J., ZHANG J., WANG J., MCGUFFIN M. J., YUAN X.: BarcodeTree: Scalable comparison of multiple hierarchies. *IEEE Transactions on Visualization and Computer Graphics 26*, 1 (2020), 1022–1032.

[MDM*15] MALIK S., DU F., MONROE M., ONUKWUGHA E., PLAISANT C., SHNEIDERMAN B.: Cohort comparison of event sequences with balanced integration of visual analytics and statistics. In *Proceedings of the International Conference on Intelligent User Interfaces* (2015), pp. 38–49. https://doi.org/10.1145/2678025.2701407

[MGT*03] MUNZNER T., GUIMBRETÌERE F., TASIRAN S., ZHANG L., ZHOU Y.: TreeJuxtaposer: Scalable tree comparison using focus+context with guaranteed visibility. In *Proceedings of the ACM SIGGRAPH* (2003). https://doi.org/10.1145/1201775

[MNL*20] MOHANTY S., NYGREN E., LAURENT F., SCHNEIDER M., SCHELLER C., BHATTACHARYA N., WATSON J., EGLI A., EICHENBERGER C., BAUMBERGER C., VIENKEN G., STURM I., SARTORETTI G., SPIGLER G.: Flatland-RL: Multi-agent reinforcement learning on trains. http://arxiv.org/abs/2012.05893 (2020)

[PS16] PLAISANT C., SHNEIDERMAN B.: The diversity of data and tasks in event analytics. In *Proceedings of the IEEE VIS 2016 Workshop on Temporal & Sequential Event Analysis* (2016). http://eventevent.github.io/papers/EVENT_2016_paper_13.pdf

[PW14] PERER A., WANG F.: Frequence: Interactive mining and visualization of temporal frequent event sequences. In *Proceedings of the International Conference on Intelligent User Interfaces* (2014), pp. 153–162. https://doi.org/10.1145/2557500.2557508

[QBW*20] QI J., BLOEMEN V., WANG S., WIJK J. V., WETERING H. V. D.: STBins: Visual tracking and comparison of multiple data sequences using temporal binning. *IEEE Transactions on Visualization and Computer Graphics 26* (2020), 1054–1063.

[RT81] REINGOLD E. M., TILFORD J. S.: Tidier drawings of trees. *IEEE Transactions on Software Engineering SE-7* (1981), 223–228. https://doi.org/10.1109/TSE.1981.234519

[Sta] Statsbomb.com: Statsbomb Academy Free Data. https://statsbomb.com/academy/#data. Accessed: 2021-11-29.

[TGK*17] TOMINSKI C., GLADISCH S., KISTER U., DACHSELT R., SCHUMANN H.: Interactive lenses for visualization: An extended survey. *Computer Graphics Forum* (2017), 173–200. https://doi.org/10.1111/cgf.12871

[The21] The Flatland Community: The dataset of recorded episodes from Flatland 2020 NeurIPS Competition winners. https://www.aicrowd.com/challenges/flatland/dataset_files (2021). Accessed: March 2021.

[VJC09] VROTSOU K., JOHANSSON J., COOPER M.: ActiviTree: Interactive visual exploration of sequences in event-based data using graph similarity. *IEEE Transactions on Visualization and Computer Graphics 15* (2009), 945–952.

[WG12] WONGSUPHASAWAT K., GOTZ D.: Exploring flow, factors, and outcomes of temporal event sequences with the outflow visualization. *IEEE Transactions on Visualization and Computer Graphics* (2012), 2659–2668. https://doi.org/10.1109/TVCG.2012.225

[WGP*11] WONGSUPHASAWAT K., GÓMEZ J. A. G., PLAISANT C., WANG T. D., BEN S., TAIEB-MAIMON M.: LifeFlow: Visualizing an overview of event sequences. In *Proceedings of the Conference on Human Factors in Computing Systems* (2011), pp. 1747–1756. https://doi.org/10.1145/1978942.1979196

[WS09] WONGSUPHASAWAT K., SHNEIDERMAN B.: Finding comparable temporal categorical records: A similarity measure with an interactive visualization. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology* (2009), pp. 27–34. https://doi.org/10.1109/VAST.2009.5332595

[ZLD*15] ZHAO J., LIU Z., DONTCHEVA M., HERTZMANN A., WILSON A.: MatrixWave: Visual comparison of event sequence data. In *Proceedings of the Conference on Human Factors in Computing Systems* (2015), pp. 259–268. https://doi.org/10.1145/2702123.2702419

Supporting Information

Additional supporting information may be found online in the Supporting Information section at the end of the article.

Video S1