Algorithms for Consistent Dynamic Labeling of Maps With a Time-Slider Interface

Annika Bonerath, Anne Driemel, Jan-Henrik Haunert, Herman Haverkort, Elmar Langetepe, and Benjamin Niedermann

***Abstract*—User interfaces for inspecting spatio-temporal events often allow their users to filter the events by specifying a time window with a time slider. We consider the case that filtered events are visualized on a map using textual or iconic labels. However, to ensure a clear visualization, not all filtered events are annotated with a label. We present algorithms for setting up a data structure that encodes for every possible time window the set of displayed labels. Our algorithms ensure that the displayed labels never overlap and guarantee the stability of the labeling during certain basic interactions with the time slider. Assuming that the labels have different priorities (weights), we aim to maximize the weight of the displayed labels integrated over all possible time windows. As basic interactions, we consider moving the entire time window, symmetrically scaling it, and dragging one of its endpoints. We consider two stability requirements: (1) during a basic interaction, a label should appear and disappear at most once; (2) if a label is displayed for a time window Q, then it is also displayed for all the time windows contained in Q and that contain its timestamp. We prove that finding an optimal solution is NP-hard and propose efficient constant-factor approximation algorithms for unit-square and unit-disk labels, as well as a fast greedy heuristic for arbitrarily shaped labels. In experiments on real-world data, we compare the non-exact algorithms with an exact approach through integer linear programming.**

***Index Terms*—Map labeling, approximation algorithm, dynamic query interface, temporal consistency, time-window query.**

I. Introduction

MAP labeling is a standard technique for visualizing spatial data. It refers to annotating a map with text and icons. Recently research on map labeling has dealt with consistency constraints for interactive maps [1], [2], [3], [4], e.g., to avoid that users are distracted by flickering labels. As an open challenge, we address with this article the consistent labeling of interactive maps for the exploration of spatio-temporal events.

Received 16 September 2022; revised 15 November 2024; accepted 23 December 2024. Date of publication 8 January 2025; date of current version 5 September 2025. This work was supported in part by German Research Foundation under Germany's Excellence Strategy, under Grant EXC-2070 - 390732324 - PhenoRob, in part by Hausdorff Center for Mathematics, in part by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Projektnummer under Grant 459420781, and in part by the Open Access Publication Fund of the University of Bon. Recommended for acceptance by J. Seo. *(Corresponding author: Annika Bonerath.)*

Annika Bonerath, Anne Driemel, Jan-Henrik Haunert, Herman Haverkort, and Elmar Langetepe are with the University of Bonn, 53113 Bonn, Germany (e-mail: bonerath@igg.uni-bonn.de; haunert@igg.uni-bonn.de).

Benjamin Niedermann is with yWorks GmbH, 72070 Tübingen, Germany.

This article has supplementary downloadable material available at https://doi.org/10.1109/TVCG.2025.3527582, provided by the authors.

Digital Object Identifier 10.1109/TVCG.2025.3527582

Every spatio-temporal event corresponds to a location and a timestamp. Examples are natural phenomena (e.g., earthquakes, storms, or bird sightings) or cultural events (e.g., concerts). A typical task that a user wants to solve with the data is to search for an event that lies in a time window. Consider a singer who has several concerts at different locations and times. Then a typical task of a music fan is to find a concert in a certain month that has a good location. Beyond sets of spatio-temporal events, our method can be applied to other use-cases. For example, consider a tourist who searches for a hotel in a foreign city. Every hotel is associated with a location and a room fare (instead of a timestamp) and the tourist searches for hotels that lie in a desired price range. In the remainder of this paper, we consider spatio-temporal events but all our models and approaches can be used for similar data sets.

A common interface for the described task consists of a map displaying the events' location and possibly additional information as well as a filter tool to search for events in time windows. In Fig. 1, we show our exemplary interface that allows users to specify a time window and receive the visualization.

The events are displayed with annotations on the map that are either simple (e.g., in the use-case of tornadoes, we have used circular annotations showing the tornadoes' strengths) or more complex (e.g., diagrams, icons, or plots showing additional data). These annotations are placed at the locations of the events. Fig. 1 shows three maps of the United States with circular annotations displaying the occurrences of tornadoes in winter, spring, and summer. To have a clear visualization and avoid an occluded appearance of the map, only a selection of events is displayed. We call a selection of annotations where no two annotations overlap and whose timestamps lie in a time window a *labeling* of that time window.

Commonly, the filtering for a time window is implemented by time sliders [5]. In our interface, illustrated in Fig. 1, we introduce a *timeline* that consists of a time axis as well as a a time slider (purple rectangle) that represents the time window. We allow user interaction with a time slider that enables the following basic interactions:

1.) *panning:* continuous translation of the time window (see Fig. 2(a)),
2.+3.) *left-sided and right-sided scaling:* continuous change of the left or right boundary of the time window, respectively (see Fig. 2(b)),
4.) *uniform scaling:* continuous change of both boundaries of the time window in opposite directions, such that the

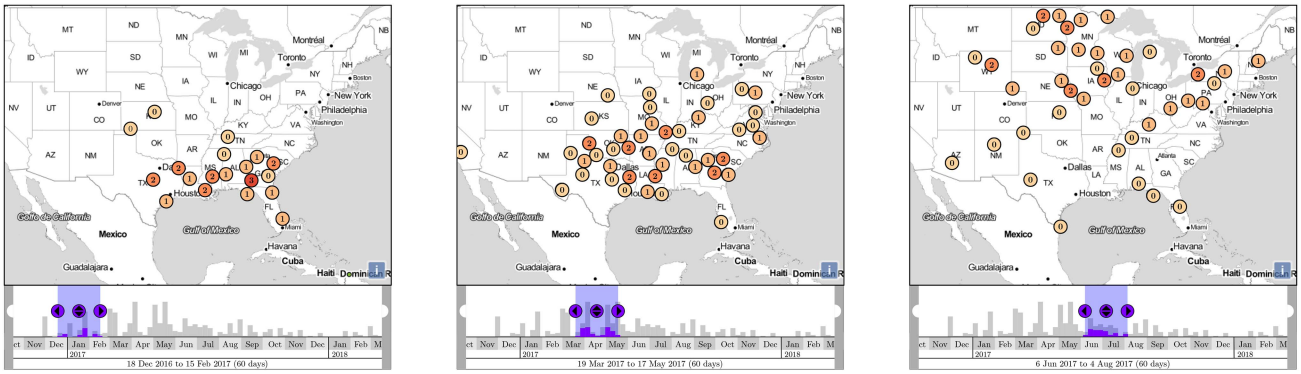


Fig. 1. Labelings of tornadoes in the year 2017 in the United States. The tornadoes particularly exist (left) in the southeast in winter, (middle) in the midwest in spring, and (right) in the north of the United States in summer. The colors and numbers indicate the strengths of the tornadoes. The labelings were computed with our approach.

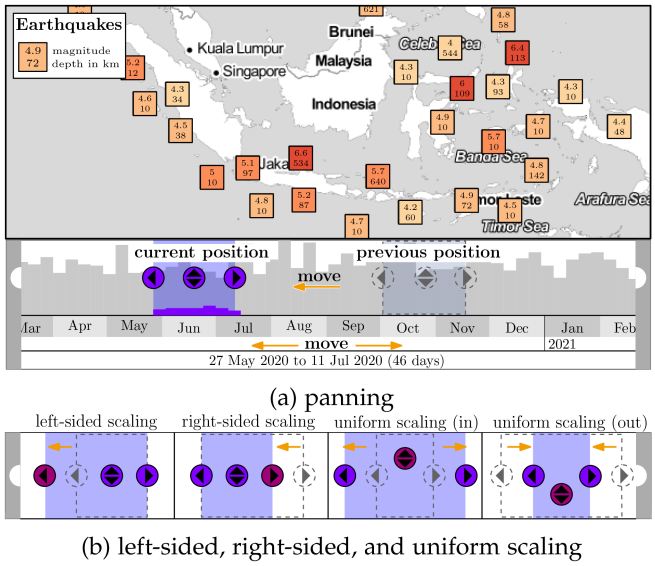


Fig. 2. Labeling of earthquakes in 2020 in Southeast Asia and basic interactions with the time slider. The labeling was computed with our approach. *Map tiles by Stamen Design, under CC BY 3.0. Map data by OpenStreetMap, under ODbL.*

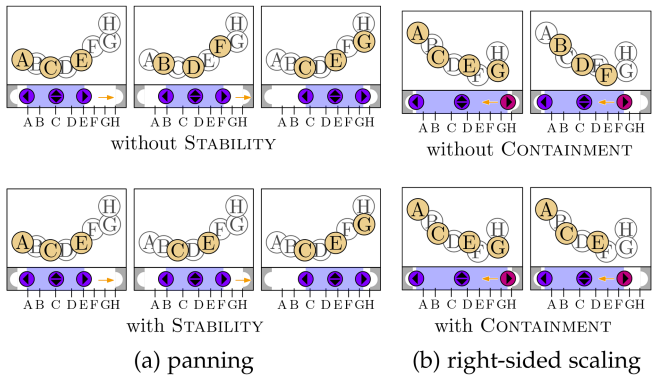


Fig. 3. Sequences of map frames for two basic interactions. The colored labels are displayed, while the white labels and the timestamps are shown only for illustration.

center of the time window remains the same (see Fig. 2(b)).

When the user performs a basic interaction, a sequence of map frames is generated and displayed where each map frame shows the labeling of one time window. These map frames form an animation of the map showing the occurrences of the events over time. In the following, we discuss our approach for selecting the labels for each time window.

At first, we aim for reproducibility, which means that the labeling of a time window should be the same no matter what kind of basic interaction has happened before.

Then, we would like to have a good selection of labels for every time window. In static map labeling, a typical strategy is to show a maximally large selection of overlap-free annotations [6] to obtain a high information density while preserving the clearness of the visualization. For our scenario, we assume that the events are of different priorities that are represented by weights. We want to maximize the sum of the weights of the displayed labels integrated over all time windows while, for every time window, no two displayed labels overlap (similar to [7]). This objective allows us, e.g., in the hotel use-case to display as many potential options to the user as possible.

However, by maximizing the sum of the weights of the displayed labels integrated over all time-window queries, the user may experience undesirable flickering effects from frame to frame. That means a single annotation may appear and disappear repeatedly. Unless additional requirements on the labelings are enforced, this can happen even within a single basic interaction. For example, in Fig. 3(a) (upper row), the labels A, C, and E appear and disappear multiple times although their timestamps lie in the time window for the entire sequence. As such flickering may distract the user, we require that the sequence of presented labelings is stable. In detail, we require that during one basic interaction, each label may appear and disappear only once; Fig. 3(a) (lower row).

Moreover, we want to ensure that the user is able to isolate a single event by systematically shrinking the time window. For example, in Fig. 3(b) (upper row) the labels A, C, and E disappear repeatedly although they are still contained in the time window. The user may think that A, C, and E are not contained in the time window after shrinking. Fig. 3(b) (lower row) shows a solution that allows the isolation of events.

To summarize, we require the following properties.

- REPRODUCIBILITY. The labeling of a time window Q is always the same. That means it is independent of the basic interactions that happened before.
- MAXINFO. Integrated over all possible time-window queries the sum of the weights of the displayed labels is maximized.
- STABILITY. Changing the time window by one basic interaction, a label appears and disappears at most once.
- CONTAINMENT. If a label of an event is displayed for a time window Q then it is also displayed for all the time windows that are contained in Q and contain the timestamp of the event.

We ensure REPRODUCIBILITY by pre-computing a data structure that encodes for every time window the set of displayed labels. With the properties STABILITY and CONTAINMENT, we enforce the consistency of time-window labelings: STABILITY avoids flickering effects, while CONTAINMENT allows the user to isolate events. As we show later, CONTAINMENT subsumes STABILITY in our formal model. Without the properties STABILITY and CONTAINMENT, the property MAXINFO can be implemented by optimizing each query independently. Hence, requiring STABILITY and CONTAINMENT substantially changes the problem. Our Contribution consists of three parts:

1) A *new model* for consistent map labeling during time-slider interactions that ensures the properties REPRODUCIBILITY, STABILITY, CONTAINMENT, and MAXINFO.
2) *Algorithms* for pre-computing the labeling for every query complying with our model. The labeling can be stored in a standard data structure that enables efficient retrieval of maps during interaction. We show that queries for maps essentially correspond to rectangle-stabbing queries and, hence, we can propose to use STR-packed R-trees [8].
3) An *experimental evaluation* based on a comparison of our methods with a baseline method that does not incorporate any consistency criteria.

We invite the reader to try out our prototypical implementation at https://www.geoinfo.uni-bonn.de/twl. We want to emphasize that although this implementation of the visualization exists, our contribution is not a visualization system but the model and algorithms for consistent map labeling.

II. RELATED WORK

The interactive visualization of spatio-temporal data is an important branch in the research areas visual analytics [9], [10], [11], geovisualization [12], and interactive cartography [13]. The research field *visual analytics* was introduced by Thomas and Cook [14] focusing on how to support data processing by users with digital visualizations. One of its main areas is interactive visual interfaces for supporting users in the analytical process [10]. The subfield of visual analytics that focuses on spatial data is called geovisual analytics [15]. The research area *geovisualization* comprises work on algorithms and data structures for creating visualizations for spatial data that offer a high level of interaction and aim at data exploration [13]. For a detailed survey on the exploration of spatio-temporal data we refer to Andrienko et al. [16]. The research field *interactive cartography* focuses on the design and model of user interactions that change the visualization of spatial data. For details, we refer to the surveys by Roth [13] and Harrower and Fabrikant [17]. In information visualization, a commonly used technique for interactions is *dynamic query interfaces*, which were introduced by Williamson and Shneiderman [18]. These are used for spatio-temporal data and for various other kinds of data. A dynamic query interface enables the user to graphically define the query and receive a real-time response and continuous animations. User studies showed good performance in comparison to paper printout, text search, and form-fill-in interfaces [18], [19]. Especially the implementation of a dynamic query interface with a time slider is a standard tool for information visualization systems [5], [18], [20], [21], [22], [23]. Due to the required real-time response, the problem of efficiency arises [24]. In computational geometry, *time-windowed data structures* aim at efficiently answering time-window queries as they are emitted by time sliders. The idea is to pre-process the possibly large data set into a data structure that can be queried for time windows in real time. Current research on time-windowed data structures focuses on relational event graphs [25], [26], [27], basic problems from computational geometry [28], [29], [30], [31], [32], and also on event visualization based on α-shapes [33] and density maps [34]. Our approach is a time-windowed data structure that uses labeling as the underlying visualization technique. *Map Labeling* is a widely investigated field where plenty of algorithms with respect to (i) label placement, (ii) label geometry, (iii) animation, and (iv) user interaction have been considered. For the static case, i.e., the non-animated and non-interactive case, a common goal is to maximize the number of the displayed labels (or their total weight) while avoiding overlapping labels [6], [35], [36]. For the non-interactive animated case, additional stability constraints are added [37], [38], [39]. An example of non-interactive animations can be fly-bys or fly-throughs with continuously changing viewpoints. For the interactive animated case, Been et al. [7] introduced the concept of active ranges for labels, considering zooming, panning, and rotations of the map. This concept was investigated intensively from an algorithmic point of view. Been et al. [40] proved that active range maximization is NP-hard for zooming and give constant-factor approximations. Active range maximization for zooming has been similarly considered for further variants [1], [41], [42]. For active range maximization in rotating maps, Gemsa et al. [43] presented an NP-hardness proof and proposed approximation algorithms. For the same scenario, Gemsa et al. [44] experimentally evaluated approximation algorithms and greedy heuristics utilizing ILP formulations. Similarly to active range maximization, Funke et al. [45] and Bahrdt et al. [2] considered circular and prioritized labels whose radius grows with the scale of the map. They present algorithms for efficiently computing an elimination order of the labels that avoid overlaps. With a similar goal, algorithms for map generalization have been developed that ensure consistency during interactions. This includes methods for the consistent selection of roads from a detailed road data set during continuous zooming [46] or continuous movement of a focus area [47] as well as methods for the

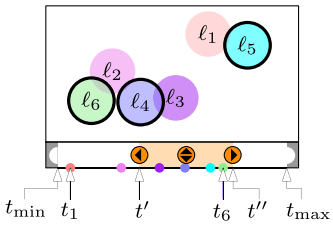


Fig. 4. Map and time slider for events $e_1, \ldots, e_6$, time-window query $Q = [t', t'']$ and labeling for $[t', t'']$ consisting of ℓ_4, ℓ_5, ℓ_6 (illustrated with black stroke).

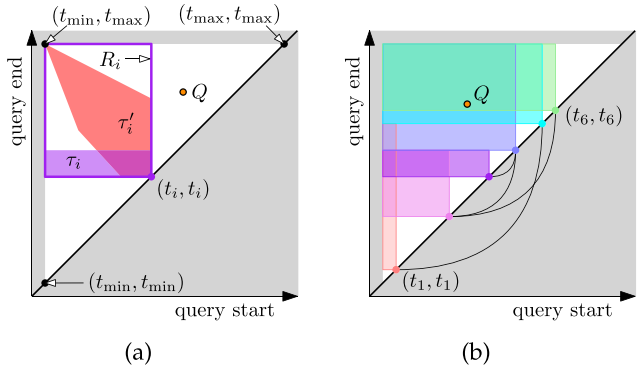


Fig. 5. Time-window query Q and query region (white triangle). (a) activity range R_i, activity regions τ_i and τ_i', τ_i' is never part of an optimal activity diagram. (b) valid activity diagram of events presented in Fig. 5; pairs of conflicting events are connected with a curve.

consistent aggregation of areas during continuous zooming [3]. A common approach is to pre-compute a data structure from which a map corresponding to a scale or preferences specified by the user can be rapidly retrieved [4].

From a more technical point of view, our work is also related to data structures that aim at improving query times for the interactive visualization of spatio-temporal data, e.g., the Data Cubes [48], or more recent variants such as NanoCubes [49] or TimeLattice [50]. These data structures do not take consistency criteria for the visualization into account. The focus is solely on the improvement of query time with less additional memory consumption.

III. Formal Model

In the following, at first, we introduce our model in a formal way. Second, we discuss the structural results that come with our model.

A. Problem Definition

We assume that we are given n spatio-temporal *events* $e_1, \ldots, e_n$. Each event e_i is represented by a point p_i in the plane, a timestamp t_i specifying when the event occurred, and a weight $w_i \in \mathbb{R}^+$ reflecting the event's importance. We assume that the events are ordered such that $t_1 \leq \ldots \leq t_n$. Let $E = \{e_1, \ldots, e_n\}$ be the set of all events. For each event we are given a *label* ℓ_i which is a geometric object in the plane, e.g., an axis-aligned square or a disk centered at p_i as shown in Figs. 1 and 2. We say that two labels (and correspondingly their events) are in *conflict* if the labels overlap in the plane.

Due to the application scenario, where the user changes the time-window query by moving sliders, we are given a minimal slider position $t_{\min}$ and a maximal slider position $t_{\max}$ such that $t_i \in [t_{\min}, t_{\max}]$ for each $1 \leq \check{} i \check{} \leq n$. We call a time interval $Q = [t', t'']\check{} \subseteq \check{}[t_{\min}, t_{\max}]$ a *time-window query*.

For a time-window query Q, let E_Q^* be the subset of E that contains each event $e_i \in E$ for which t_i lies in Q, i.e., $t_i \in Q$. Let L_Q^* denote the set of labels of events in E_Q^*. Note that displaying all labels in L_Q^* might lead to label overlaps. We call $L_Q \subseteq L_Q^*$ where no two labels of L_Q overlap a *time-window labeling* or more shortly a *labeling*. Fig. 4 illustrates a time-window query and a labeling. Further, we call a label $\ell \in L_Q$ an *active label* of Q, i.e., it is displayed for Q. We denote the set of events that correspond to labels in L_Q by E_Q.

Assuming we are considering only a single time-window query, we get the standard static labeling problem. To implement MaxInfo in this static case, one would look for a labeling L_Q that maximizes either the number of active labels or the sum of their weights $\sum_{e_i \in E_Q} w_i$. However, our use-case is not limited to one time-window query but the user may interact with the time slider. We want to emphasize that optimizing the labeling for each time-window query independently might lead to a violation with Reproducibility, Stability, and Containment. In the following, we introduce our data structure and show how to optimize MaxInfo while requiring Reproducibility, Stability, and Containment for time-slider interactions.

To ensure Reproducibility, we pre-compute the labelings of all time-window queries in advance and store them in a specially defined data structure. As shown in Fig. 5, each time-window query $Q = [t', t'']$ corresponds to a point (t', t'') in the plane. For brevity depending on the context we interpret Q either as interval $[t, t'']$ or point (t', t'') in the plane. Each point representing a time-window query lies in the triangle spanned by $(t_{\min}, t_{\min})$, $(t_{\min}, t_{\max})$, and $(t_{\max}, t_{\max})$, which we call the *query region*; see Fig. 5(a). We call the line through $(t_{\min}, t_{\min})$ and $(t_{\max}, t_{\max})$ the *main diagonal*. An event e_i corresponds to a point (t_i, t_i) on the main diagonal. We observe that the label ℓ_i can only be active for queries that lie in the rectangle R_i that is spanned by (t_i, t_i) and $(t_{\min}, t_{\max})$; we call R_i the *range* of e_i. For the proposed data structure, we pre-compute for each event e_i a region τ_i in R_i; see the colored regions in Fig. 5(b). We call τ_i the *activity region* of e_i. The activity region τ_i exactly contains the queries for which the label ℓ_i is active.

To enforce the properties Stability and Containment, we can only allow activity regions of certain shapes. At first, we formalize the property Containment. An activity region τ_i is *monotone* if, for two time-window queries Q and Q' in the range of e_i with $Q' \subseteq Q$, it holds that Q' lies in τ_i if Q lies in τ_i. Fig. 5(a) illustrates two monotone activity regions. Clearly, an event with a monotone activity region fulfills the property Containment and we show later that it also satisfies property Stability. Roughly speaking such an activity region of an event e_i is the union of a set of axis-aligned rectangles whose bottom-right corners are (t_i, t_i), where t_i is the timestamp of e_i; for proof see Lemma 1. Later we argue that we can reduce activity regions to be rectangles; as illustrated in Fig. 5(b).

Let $T = \{\tau_1, \ldots, \tau_n\}$ be a set of monotone activity regions of the events $E = \{e_1, \ldots, e_n\}$. A query Q on T yields the set $L_Q^T = \{\ell_i \mid Q \in \tau_i \text{ with } 1 \leq i \leq n\}$. We call T an *activity diagram* of the events E. Further, it is *valid* if each query Q on T yields a time-window labeling L_Q^T. This is equivalent to requiring that no two activity regions τ_i and τ_j intersect when ℓ_i and ℓ_j are in conflict. In the following, we formalize the property MAXINFO. Let $E = \{e_1, \ldots, e_n\}$ be a set of spatio-temporal events. We call $v(\tau_i) = w_i \cdot \tilde{}\text{area}(\tau_i)$ the *volume* of τ_i where $\text{area}(\tau_i)$ is the area of τ_i and we call $v(T) = \sum_{i=1}^{n} v(\tau_i)$ the *total volume* of T. The total volume $v(T)$ corresponds to the sum of weights of displayed labels integrated over all possible time-window queries, and hence, maximizing $v(T)$ leads to property MAXINFO.

TIMEWINDOWLABELING summarizes our problem setting to optimize MAXINFO while guaranteeing REPRODUCIBILITY and CONTAINMENT. Later we show that TIMEWINDOWLABELING also guarantees STABILITY.

TIMEWINDOWLABELING

Given:: *A set $E = \{e_1, \ldots, e_n\}$ of spatio-temporal events with labels; the bounds t_{min} and t_{max} of the activity diagram.*

Find: *A valid activity diagram $T = \{\tau_1, \ldots, \tau_n\}$ of monotone activity regions for E maximizing $\sum_{i=1}^{n} w_i \cdot \tilde{}\,area(\tau_i)$, where $area(\tau_i)$ is the area of τ_i in the activity diagram.*

We say that an optimal solution for TIMEWINDOWLABELING is an *optimal* activity diagram.

B. Structural Results

We show that for solving TIMEWINDOWLABELING it suffices to consider rectangular activity regions.

Lemma 1. The activity regions of an optimal activity diagram T for TIMEWINDOWLABELING are axis-aligned rectangles whose bottom right corners lie on the main diagonal.

Proof. We first prove that the property CONTAINMENT implies that the activity region τ_i of any event $e_i \in E$ is the union of a set of axis-aligned rectangles whose bottom-right corners are (t_i, t_i), where t_i is the timestamp of e_i. Afterward, we show that optimizing the property MAXINFO implies that τ_i is a single axis-aligned rectangle whose bottom-right corner is (t_i, t_i). Assume that $\tau_i \in T$ of e_i is a monotone activity region and, hence, satisfies the property CONTAINMENT. Consider an arbitrary time-window query $Q = [t', t'']$ that lies in τ_i. For any query $Q' \subseteq Q$ it holds that it lies in the axis-aligned rectangle R_Q spanned by (t', t'') and (t_i, t_i). By the property CONTAINMENT it follows that R_Q is part of the activity region τ_i. Hence, we obtain that τ_i is the union of a (possibly infinitely large) set A_i of axis-aligned rectangles whose bottom-right corners are (t_i, t_i). Now assume that T is optimal with respect to the property MAXINFO, and hence it is maximal with respect to the total area of the activity regions. Among all rectangles in A_i let R_{top} be a rectangle whose top side has a maximal y-coordinate and let R_{left} be a rectangle whose left side has minimal x-coordinate; see Fig. 6(a). Let y_{top} and x_{left} be the corresponding y-coordinate and x-coordinate, respectively. We observe that any activity region that intersects the rectangle H_i spanned by $(x_{\text{left}}, y_{\text{top}})$ and (t_i, t_i) intersects either R_{top} or R_{left}. Hence, for any event $e_j \in E$ that is in conflict with e_i its activity region τ_j cannot intersect H_i. Thus, as T maximizes the total volume, which increases with the area of the activity regions, the rectangle H_i is part of τ_i. By the extremal choice of R_{top} and R_{left}, we further obtain that τ_i is exactly H_i. Thus, we obtain the statement of the lemma. □

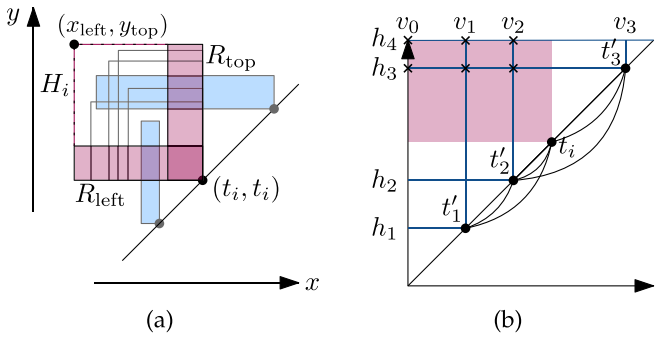


Fig. 6. (a) Proof of Lemma 1. Illustrated is event e_i with timestamp t_i; range R_{left} with minimal x-coordinate x_{left} and R_{top} with maximal y-coordinate y_{top} of e_i; and rectangle H_i spanned by the lower-right corner (t_i, i_i) and the upper-left corner $(x_{\text{left}}, x_{\text{top}})$. Any activity regions (blue) of other events that intersect H_i also intersect either R_{left} or R_{top}. (b) Discretization of solution space. Shooting vertical and horizontal rays from the timestamps induces a grid that is the basis for the candidate activity regions.

Due to Lemma 1 we can discretize the solution space such that for each event we can choose its activity region from $O(n^2)$ rectangles. We define for each event $e_i \in E$ a *candidate set* C_i as follows. Let $e'_1, \ldots, e'_k$ be the events that are in conflict with e_i and let $t'_1, \ldots, t'_k$ be the timestamps of these events, respectively. Further, for each event e'_j with $1 \leq \tilde{}j\tilde{} \leq k$, let v_j be the vertical segment that connects (t'_j, t'_j) with (t'_j, t_{max}). Similarly, let h_j be the horizontal segment that connects (t'_j, t'_j) with $(0, t'_j)$; see Fig. 6(b). Further, let v_0 be the vertical line through $(t_{\text{min}}, 0)$ and let h_{k+1} be the horizontal line through $(0, t_{\text{max}})$. Let S be the set of pairwise intersection points between $h_1, \ldots, h_k, h_{k+1}$ and $v_0, v_1, \ldots, v_k$. For e_i the candidate set C_i contains the axis-aligned rectangles that are spanned by (t_i, t_i) and the intersection points of S that lie in the range R_i of e_i.

Lemma 2. Let T be an optimal activity diagram for E. For each event $e_i \in E$ its activity region $\tau_i \in T$ is a rectangle in the candidate set C_i.

Proof. We show that, for each optimal activity diagram of E, the activity region τ_i of an event $e_i \in E$ is contained in C_i. By Lemma 1, τ_i is an axis-aligned rectangle whose bottom right corner is (t_i, t_i). A similar statement holds for any event e_j that is in conflict with e_i. In particular, the horizontal line supporting the lower boundary and the vertical line supporting the right boundary of an activity region in the optimal solution are fixed a priori. Assume for the sake of contradiction that the upper boundary of τ_i lies on a horizontal line that is not defined by any of the timestamps $t_1, \ldots, t_n$. Then, either we can extend τ_i upwards, or there exists an activity region τ_j that blocks τ_i from above. However, since the bottom boundary of this other activity region is fixed at t_j, the latter cannot be the case. Therefore, we could extend τ_i implying that the solution is not optimal. A symmetric argument can be made for the left boundary of τ_i. Therefore, it must be that $\tau_i \in C_i$. □

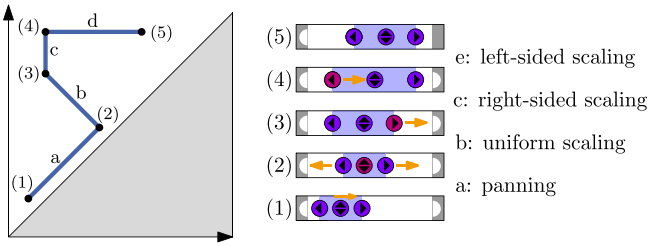


Fig. 7. Illustration of a query path. (1)–(5): the time windows at the vertices of the query path.

In the following, we argue that the property CONTAINMENT subsumes STABILITY, which implies that a solution of TIMEWINDOWLABELING also satisfies the property STABILITY. As described in Section I, we assume that the time-window query is chosen interactively using sliders. We allow the user to choose the time window $Q = [t', t'']$ by four basic interactions: *panning*, *left-sided scaling*, *right-sided scaling*, and *uniform scaling*. In more detail, when panning the time window by Δ the resulting time-window query is $[t' + \Delta, t'' + \Delta]$. Further, the left-sided scaling changes the left boundary t' of the time window to $t' + \Delta$, and the right-sided scaling changes the right boundary t'' of the time window to $t'' + \Delta$ by an amount Δ. Finally, the uniform scaling changes the time window by an amount Δ in both directions to $[t' - \Delta, t'' + \Delta]$. We note that Δ can also be negative. Consider the interaction of a user with the time sliders in an activity diagram. The sequence of time-window queries issued by the user forms a trajectory in the activity diagram; we call it a *query path*. We observe that a query path consists of a sequence of vertical, horizontal, and diagonal segments such that each segment corresponds to a basic interaction. Fig. 7 gives an exemplary query path and the associated basic interactions. For optimal activity diagrams, it follows from Lemma 1 that the property STABILITY is satisfied: for a basic interaction a label appears and disappears only once, as the intersection of a segment with a rectangle is at most a single segment.

IV. COMPLEXITY AND EXACT SOLUTION

In this section, we prove that constructing an optimal activity diagram is NP-hard and present an ILP formulation for TIMEWINDOWLABELING. Once we have constructed the activity diagram, we can efficiently answer time-window queries utilizing rectangle-stabbing queries, i.e., for a time query Q we return all labels whose activity regions contain Q.

A. Computational Complexity

First of all, we show that TIMEWINDOWLABELING is NP-hard by providing a reduction from a closely related static labeling problem.

Theorem 1. Let E be a set of events with either unit-disk or axis-aligned unit-square labels. It is NP-hard to find an optimal activity diagram of E, even if each event $e \in E$ has weight 1.

Proof. Given a set S of unit disks or axis-aligned unit squares, it is NP-hard to find a set $S' \subseteq S$ of maximum cardinality such that no two elements in S' intersect [51]. We call the version of the problem with squares MISS (as a shorthand for Maximum Independent Set of Squares). By showing that TIMEWINDOWLABELING contains MISS as a special case, we prove that TIMEWINDOWLABELING is NP-hard as well. More precisely, every instance of MISS (i.e., every input set of squares), can be solved by constructing and solving a corresponding instance of TIMEWINDOWLABELING, i.e., a set of events, a set of labels, $t_{\min}$, and $t_{\max}$. We set $t_{\min} = 0$ and $t_{\max} = 2$ and annotate every square in S with the same timestamp $t_i = 1$ for our construction procedure (or *reduction*). This means that an intersection between any two activity regions is allowed if and only if the two corresponding labels intersect. Therefore, if we were given an optimal solution to the instance of TIMEWINDOWLABELING, we could simply return the set of all labels with non-empty activity regions as an optimal solution to the MISS instance. The proof for unit disks works analogously. □

B. ILP Formulation

Due to Theorem 1, we pursue solutions based on ILP formulations. The general idea of an ILP formulation is to formalize the given optimization problem as a linear objective function subject to linear inequality constraints. As the variables are integers in an ILP formulation, solving it is NP-hard in general [52]. However, there are powerful solvers that often can be used to solve such formulations in practice. For the proposed ILP formulation, we interpret TIMEWINDOWLABELING as a problem of finding a maximum-weight selection of rectangles that represent possible activity regions. Lemma 2 yields that it is sufficient to consider the rectangles contained in the set C_i of candidates for each event e_i. For each event $e_i \in E$ and each rectangle $r \in C_i$ we introduce a binary variable $x_{i,r}$, which we interpret such that $x_{i,r} = 1$ if and only if the rectangle r is selected as activity region τ_i for e_i. For each event e_i we enforce that at most one activity region can be selected by the constraint

$$\sum_{r \in C_i} x_{i,r} \leq 1. \tag{1}$$

For every pair of distinct events $e_i, \tilde{e}_j \in E$ that are in conflict, we need to ensure that their labels are not displayed at the same time. We formalize this by introducing for every pair $(r, r') \in \tilde{C}_i \times \tilde{C}_j$ where r and r' intersect a constraint

$$x_{i,r} + x_{j,r'\tilde{}} \leq 1. \tag{2}$$

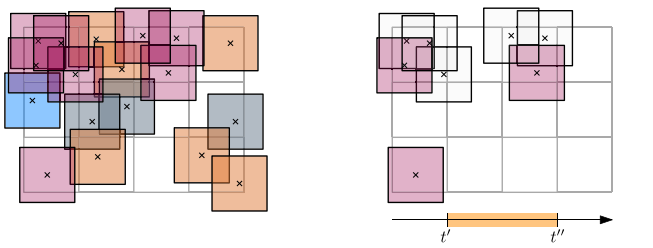
At last, our aim is to maximize the total volume of the activity diagram. This corresponds to maximizing the following objective

$$\sum_{e_i \in E} \sum_{r \in C_i} w_i \cdot \tilde{\text{area}}(r) \cdot x_{i,r}. \tag{3}$$

We obtain the optimal activity diagram T_{ILP} for E by setting for each event $e_i \in \tilde{E}$ its activity region τ_i as the rectangle $r \in \tilde{C}_i$ with $x_{i,r} = 1$.

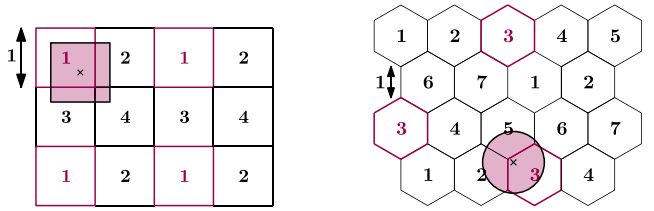
Theorem 2. The set T_{ILP} is an optimal activity diagram for E.

With our ILP formulation, we can replace the volume of the activity diagram with other measures. For example, the square



(a) partitioning of labels (b) labeling for time-window query

Fig. 8. Approximation algorithm for unit-square labels (see Fig. 9(a)). (a) Labels of one color correspond to one subset. (b) Assume the solution for subset 1 is best. Then, the labeling of a time-window query contains only labels from subset 1 (pink labels).



(a) square grid (b) hexagonal grid

Fig. 9. Grids used to partition a set of (a) unit-square labels and (b) unit-disk labels into four and seven subsets, respectively.

root of the area of the activity regions could lead to very small activity regions being avoided in the optimal solution.

V. Non-Exact Solutions

As TimeWindowLabeling is NP-hard, we tackle the problem with faster algorithms that guarantee Reproducibility, Stability, and Containment while they might not perform best for MaxInfo. We present approximation algorithms that guarantee a certain approximation factor α, i.e., the ratio between the total volume of the obtained activity diagram and the total volume of an optimal activity diagram is at most the approximation factor. Further, we present a greedy heuristic and a combination of the approximation and greedy approaches.

A. Approximation Based on a Partitioning Scheme

In this section, we discuss approximation algorithms for specific types of labels: (i) the labels are axis-aligned rectangles of equal width and equal height, or (ii) the labels are disks of equal size. For rectangular labels we prove an approximation factor of 4 and for disk-shaped labels an approximation factor of 7. Note that for any problem instance, i.e., for any input to TimeWindowLabeling, the map can be scaled even with different scale factors in the x- and y-dimension without changing the structural properties such as the intersection relationships between labels. Therefore, we assume, without loss of generality, that our labels are either unit squares or unit disks.

Both approximation algorithms are based on the idea of (1) partitioning the given set of events into subsets that can each be solved efficiently, (2) computing an optimal solution for each subset, and (3) returning the solution of the subset with the highest objective value. Fig. 8 gives an exemplary instance, its partition, and labeling resulting from the approximation algorithm. To make it more clear, in the solution of the approximation algorithms, only events from the subset with the highest objective value have a non-empty activity region. The activity regions of labels of all other subsets are empty. For static map labeling, *line stabbing* is a common technique for partitioning a given set of labels into subsets whose label-label conflict graph is an interval graph [35]. In a similar fashion, we apply a partitioning scheme yielding subsets whose label-label conflict graph is a set of mutually disjoint cliques. This means that two labels of the same subset are in conflict with each other if and only if they are in the same clique. The following lemma implies that we can solve such instances efficiently.

Lemma 3. TimeWindowLabeling can be solved in $O(n)$ time if the label-label conflict graph is a clique and the events are given in the order of their timestamps.

We prove Lemma 3 by providing a linear-time algorithm for problem instances that satisfy the conditions of Lemma 3 in Appendix A, available online.

To use Lemma 3 for an approximation algorithm, we partition the events and their labels into cliques. For this we use a square grid in the case of unit squares and a hexagonal grid in the case of unit disks, where every edge has length one. We illustrate both grids in Fig. 9. Every label ℓ is assigned to the grid cell containing the center point of ℓ. The lexicographical order ($<$) of the cell's center points is used to break ties, meaning that a label whose center lies on the boundary between two cells c_1 and c_2 with $c_1 < c_2$ is assigned to c_1. With this, the grid with the assigned labels has the following properties:

i) For every cell, the conflict graph of the assigned labels is a clique, i.e., all assigned labels are pairwisely in conflict.
ii) For the square grid, there exists an assignment that maps each grid cell to one of the numbers 1,2,3,4, such that no two labels assigned to different cells with the same number intersect (see Fig. 9(a)).
iii) The latter holds for the hexagonal grid and numbers $1, \ldots, 7$, as shown by Chan et al. [53] in another context (see Fig. 9(b)).

Because of (i), the events that are assigned to the same grid cell can be solved optimally with the algorithm described in the proof of Lemma 3 (see Appendix A, available online). Because of (i) and (ii) or (iii), respectively, we can combine the resulting labelings for grid cells with the same number into a single labeling (in which no two labels overlap). Let $T_1, \ldots, T_k$ be the k solutions that we obtain, where $k = 4$ for square labels and $k = 7$ for disk labels. Among these solutions, we return a solution of maximum total volume, leading to the following result.

Theorem 3. The algorithm based on a partitioning scheme approximates TimeWindowLabeling with factor 4 for unit squares and with factor 7 for unit disks. If the events are given in the order of their timestamps, the algorithm can be implemented to run in $O(n + d)$ time, where d is the number of grid cells.

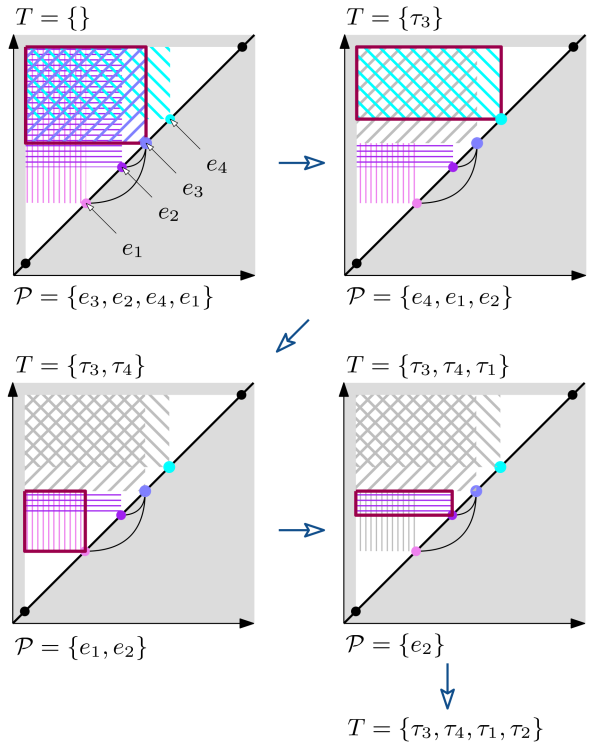


Fig. 10. Greedy heuristic for four events e_1, e_2, e_3, e_4 and equal weights. There is a label conflict for the pairs (e_1, e_3) and (e_2, e_3).

Proof. Let $T^\star$ be an optimal solution. By applying the partitioning scheme of our algorithm to the labels, we partition $T^\star$ into k solutions, which we denote with $T_1^\star, \ldots, T_k^\star$. For $i = 1, \ldots, k$, it holds that $v(T_i)\tilde{} \geq \tilde{}v(T_i^\star)$ since (i) both $T_i^\star$ and T_i are solutions for the same set of labels and (ii) T_i is optimal. Therefore, $\sum_{i=1}^{k} v(T_i)\tilde{} \geq \tilde{}v(T^\star)$ and, thus, at least one of the solutions $T_i, \ldots, T_k$ has total volume greater or equal $v(T^\star)/k$. Constructing the grid and assigning every label to its cell requires $O(n + d)$ time. For each cell, the corresponding instance of n' labels can be solved in $O(n')$ time with the algorithm for cliques. Since the instances are disjoint, solving all instances amounts to $O(n)$ time. □

B. Greedy Heuristic

In this section, we present a greedy heuristic for computing a valid activity diagram which is not based on a partitioning scheme. For illustration see Fig. 10. The greedy heuristic successively selects activity regions that yield the largest gain. While doing so, it maintains for each event that has not yet been placed in the activity diagram its maximal potential activity region. Each time a new event is selected and placed in the diagram, all remaining activity regions that are in conflict with this event are trimmed and their potential contribution is updated accordingly.

More in detail, we initialize for each event $e_i \in E$ its largest possible activity region τ_i, i.e., the region that is spanned by $(t_{\min}, t_{\max})$ and (t_i, t_i) and further, its volume as $v(\tau_i) = w_i \cdot \tilde{}\text{area}(\tau_i)$. We initialize a priority queue $\mathcal{P}$ of events increasingly ordered by their volumes and the empty solution set T; see step 1 of Fig. 10. Then, we remove the first event e_i from $\mathcal{P}$ (with largest volume) and add τ_i to the solution set T. For each event e_j that is in $\mathcal{P}$ and that is in conflict with e_i we trim τ_j to the largest possible activity region $\tau_j' \subseteq \tau_j$ that does not intersect τ_i. Finally, we update the volume of e_j in $\mathcal{P}$ to $w_j \cdot \tilde{}\text{area}(\tau_j')$, possibly changing the position of e_j in the order of $\mathcal{P}$. We repeat this process of removing the first event in $\mathcal{P}$ until $\mathcal{P}$ is empty. Then we return the valid activity diagram T.

As for the running time, note that each time an event e_i is removed from $\mathcal{P}$ we trim the activity regions and update the volumes of $O(n)$ events that are in conflict with e_j. Trimming takes $O(1)$ time and updating $\mathcal{P}$ takes $O(\log n)$ time per conflicting event if we implement $\mathcal{P}$ as a binary heap. Hence, we need $O(n^2 \log n)$ time in total.

The greedy heuristic works for arbitrarily-shaped labels, as long as it can check the existence of conflicts between labels efficiently. However, it cannot guarantee an approximation factor as good as that of the partitioning-based algorithm from Section V-A. Consider the following input with 15 events, each with weight 1 and a square-shaped label of size 6×6. The center points of the labels are: (0,0), (6,0), (0,6), (6,6), (4,4), (3,3), (9,3), (3,9), (9,9), (7,7), (6,6), (12,6), (6,12), (12,12), and (10,10). The timestamps are: 8, 8, 8, 8, 8.002, 16, 16, 16, 16, 16.001, 21, 21, 21, 21, and 20.999, respectively, and $[t_{\min}, t_{\max}] = [0, 24]$. For this input, the greedy heuristic achieves a total activity region size of less than 207.107, whereas the optimal solution has a total activity region size of at least $900.025 > 4.34 \cdot 207.107$. Thus, the approximation factor of the greedy heuristic is at least 4.34, whereas the partitioning scheme guarantees approximation factor 4.

With events of different weights we can even construct inputs that cause the greedy heuristic's performance to become arbitrarily bad; see Appendix B, available online.

C. Combining Partitioning Scheme and Greedy Heuristic

The approximation algorithms based on the partitioning scheme yield labelings in which the labels cover the underlying grid in an undesirable systematic pattern. We therefore enhance the activity diagram T_0 from these algorithms using the greedy heuristic. To that end, we copy the initial solution of the approximation T_0 to T. For every event e that has no activity region in T_0, we add the largest possible activity region τ to the priority queue $\mathcal{P}$ as before and use its volume as the sorting key for $\mathcal{P}$. More precisely, before we add τ to $\mathcal{P}$ we trim τ such that it does not intersect any activity region τ' in T_0 that corresponds to a label ℓ' that overlaps the label of e. The remaining part of the greedy heuristic remains unchanged. Hence, it successively adds activity regions in $\mathcal{P}$ to T until the solution is maximal. In particular, by trimming activity regions it ensures that the result is a valid activity diagram. We note that filling up T_0 does not negatively affect the approximation quality stated in Theorem 3. It is an open question whether there exists a better approximation algorithm.

VI. Experiments

In this section, we evaluate the presented model and algorithms on real-world data. We first consider the construction phase of our approach, in which the activity diagram is created. Afterwards, we consider the query phase. To use real-world query paths for the evaluation, we conducted a study with users.

A. Experimental Setup

We considered two different data sets. The first data set $\mathbf{E}_{\text{tornado}}$ contains 5 900 spatio-temporal events of tornadoes in the years 2015–2019. It is obtained from the National Oceanic and Atmospheric Administration of the United States[1]. The second data set $\mathbf{E}_{\text{bird}}$ contains 10 000 observations of gulls of two different species in Western Europe and at the west coast of Africa in the year 2015 [54]. For the data set of tornadoes, we rated each event e by its storm strength $z \in \{0, 1, 2, 3, 4\}$. More precisely, we set $w(e) = 2^z$, thus favoring strong over weak tornadoes. For the occurrences of gulls, each event was rated with the same weight. We note that the data sets have different spatial patterns. While the observations of gulls form large clusters, the occurrences of tornadoes appear more evenly distributed. We evaluate the algorithms discussed in Sections V-A–V-C by comparing their solutions to an exact solution of TimeWindowLabeling obtained by the ILP formulation presented in Section IV. We consider both unit disks (D) and unit squares (S) as labels, obtaining eight variants for the computation of the activity diagram: the exact solutions ILP-D and ILP-S, the approximations Part-D and Part-S by the partitioning scheme, the solutions Greedy-D, and Greedy-S of the greedy heuristic, and the combined approaches Combi-D and Combi-S. For the evaluation of the construction phase (Section VI-B) we sampled subsets of different sizes where a subset of size k contains the first k events. For the evaluation of the query phase (Sections VI-C and VI-D) we have reduced both data sets to 2000 events, each by randomly sampling them using a uniform distribution. We refer to the reduced sets as E_{tornado} and E_{bird}. The random sampling maintains the overall structure of the data but enables us to evaluate our non-exact algorithms using a slow ILP-based approach. In Section VI we show that our non-exact algorithms actually run on much larger sets. The implementations were done in Java, and the ILP formulations were solved by Gurobi 9.1. We ran the experiments on an Intel(R) Xeon(R) W-2125 CPU clocked at 4.00GHz with 128 GiB RAM.

B. Construction Algorithm Evaluation

In this section we compare the quality and the running time of our algorithms for the previously presented subsets of the data sets.

Quality of the Activity Diagrams: For a non-exact algorithm, we assess the constructed activity diagram T by comparing it to an optimal solution, i.e., we consider its *relative quality*, which is defined as

$$\frac{\text{total volume of activity diagram T}}{\text{total volume of optimal activity diagram}} \cdot 100\% \qquad (4)$$

[1]Available under public domain at https://www.spc.noaa.gov/wcm/.

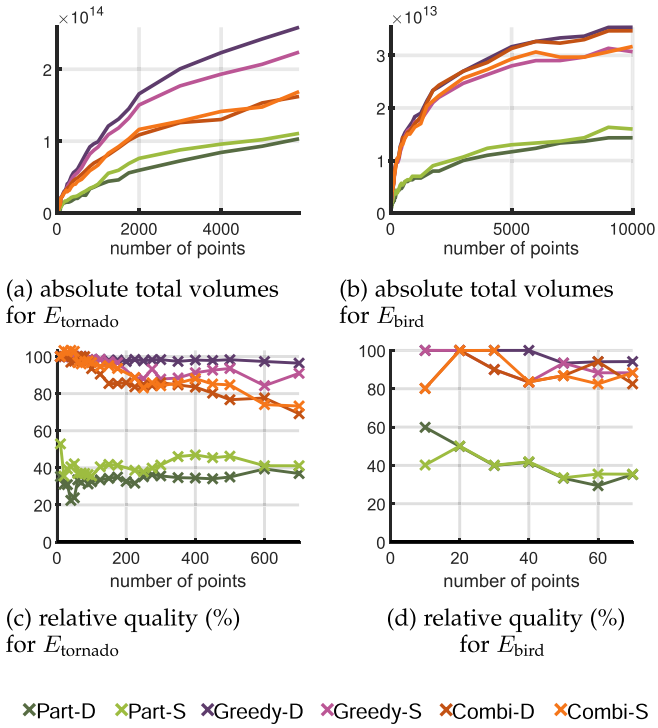


Fig. 11. Results for the quality of the data structures obtained by the approximation algorithm (Part-D, Part-S), by the greedy heuristic (Greedy-D, Greedy-S), and by the combination of the approximation and greedy (Combi-D, Combi-S). (a),(b) Total volumes for E_{tornado} and E_{bird}, respectively. (c),(d) The relative quality for E_{tornado} and E_{bird}, respectively; see (4).

For E_{tornado} the experiments illustrated in Fig. 11(a) show that the relative qualities of Greedy-D and Greedy-S are at least 84.27%, Part-D and Part-S are at least 22.45%, and Combi-D and Combi-S are at least 69.16%. We have obtained similar results for E_{bird} illustrated in Fig. 11(b). As the relative quality could be computed only for small data sets, we also consider absolute total volumes; see Fig. 11(c) and (d).

Hence, although the greedy heuristic does not give any guarantees, it gives solutions of high quality in practice. Both Combi-D and Combi-S have a head-to-head race with Greedy-D and Greedy-S, respectively, without a clear winner. Although the partitioning scheme has a theoretical approximation factor, the greedy heuristic provides substantially better results for both real-world data sets and clearly prevails over the partitioning scheme. Altogether, we suggest running both the greedy heuristic as well as the combined approach and to take the better solution of both.

Running Time: From the sampled data we consider the largest data set for which the ILP formulation could be solved and the largest data set that has been sampled; see Table I. The running times of the non-exact algorithms lie in the range of milliseconds and seconds. In particular, the approximation algorithms are clearly faster than the greedy heuristics. We deem the running times to be sufficiently small, as the activity diagrams are only created once in advance. In contrast, as solving the ILP formulations may take hours, even for small data sets, they are less applicable in practice.

TABLE I
CONSTRUCTION TIMES FOR THE OCCURRENCES OF TORNADOES (700 AND 5900 EVENTS) AND THE OBSERVATION OF GULLS (70 AND 10000 EVENTS)

algorithm	tornadoes		gulls	
	700	5900	70	10000
ILP-D	58.43 min	–	6.68 h	–
Part-D	14.98 ms	24.34 ms	19.41 ms	153.02 ms
Greedy-D	23.62 ms	1.67 sec	31.82 ms	15.02 sec
Combi-D	27.32 ms	1.69 sec	4.99 ms	14.71 sec
ILP-S	23.14 h	–	6.31 h	–
Part-S	1.01 ms	4.68 ms	1.15 ms	9.05 ms
Greedy-S	23.76 ms	1.68 sec	2.61 ms	14.95 sec
Combi-S	23.85 ms	1.66 sec	2.62 ms	13.25 sec

C. User-Generated Query Paths

To acquire query paths for the evaluation of our algorithms and model under realistic conditions, we conducted an online study; see Section VI-D. It gives us the possibility of analyzing the usage of the basic interactions and assessing the consistency of our model. Note that we do not aim for a user evaluation of our model with this study.

Study Setup The main visualization component of our interface is an interactive map that supports time-window filtering. We illustrate it for the map for the tornado events in Fig. 1. For the bird observation events, the labels' color (orange and blue) encode the two different species of gulls. The study is split into two phases. In the first phase, the participants were asked to do a tutorial. During this tutorial, the participants solved exercises on a data set of earthquakes to get familiar with the interaction of the map (see Fig. 2(a)). In the second phase, the actual experiments were performed. The participants were asked to solve eight tasks: four tasks on each data set.

1) Find a time window that roughly spans one year, starting at some day in December 2019 and ending at some day in December 2020.
2) Find a time window that spans between 30 and 40 days and shows at least 10 tornadoes in 2016.
3) Find a time window such that the majority of tornadoes (at least 20) occurred to the right of the blue line, while only a few tornadoes occurred to the left of the blue line.
4) In which time period did the highlighted tornado (blue circle) occur? The event occurred in the [first half/ second half] of [January/ February/ . . . /December] in [2015/ 2016/ . . ./ 2019].
5) Find a time window that spans between 20 and 25 days and shows at least 20 occurrences of gulls.
6) Checkmark the correct answers.
   - ☐ Some blue gulls stay in Western Europe for the whole year.
   - ☐ Some blue gulls migrate to Africa in winter.
   - ☐ Some orange gulls stay in Western Europe for the whole year.
   - ☐ Some orange gulls migrate to Africa in winter.
7) Find a time window for which all occurrences of gulls (at least 6) lie above the blue line.
8) Find a time window for which most occurrences of orange gulls lie below the blue line.

TABLE II
STATISTIC FOR THE 41 PARTICIPANTS OF THE SECOND RUN OF THE STUDY

Age	16-25:	22	26-35:	11
	36-45:	4	46-55:	2
	56-65:	1	66-99:	1
Gender	female:	13	male:	26
	other:	0	no answer:	2
Personal use of maps	daily:	5	weekly:	12
	monthly:	9	less:	15
Professional use of maps	yes:	10	no:	31

TABLE III
NUMBER OF PARTICIPANTS ANSWERING THE QUESTIONS ON PERSONAL OPINION (1 = WORST SCORE, 5 = BEST SCORE)

Question	Scoring 1	2	3	4	5	Avg.
Is the use of the time window intuitive?	1	2	3	14	21	4.3
Is the use of the timeline intuitive?	0	2	4	16	19	4.3
Is the visual connection between timeline and geographic map intuitive?	1	1	2	10	27	4.5
How useful are such interactive maps to explore data?	1	0	2	11	27	4.5
Would you like to use such interactive map in practice?	1	0	3	10	27	4.5

As the study was conducted online, we paid special attention to the correct execution by the participants.

- The screen resolution was automatically checked.
- Full-screen mode was enforced.
- The participants were asked to compare the interactive map with a screenshot confirming that it is displayed correctly.
- In the tutorial we ensured that the participants used each control at least once, thereby checking its functionality automatically.

User Feedback: Although the only purpose of the study was to get real-world query paths, we asked the users to fill in two questionnaires: one on personal preferences concerning the interactive map and one on their background using maps. We conducted the study twice. The first time we did a pilot study with nine participants which led us to change the appearance of the time sliders. Up to that point, the controls for the time window were placed on a separate slider bar. Multiple participants noted that this is not intuitive. Based on that feedback, we switched to the current design, which is inspired by the work of Haslett et al. [55] and Hochheiser and Shneiderman [21]. In the second run, we conducted the actual study for eleven days with 41 participants; see Table II. We consider the number of 41 participants who have not taken part in the first phase to be sufficient. The participants were acquired from our research network as well as from non-research-related networks. We did not specify whether the participants should complete the study in one sitting. We also asked the participants in the second run for personal feedback; see Table III. They mostly find the use of the time sliders intuitive and would like to use such maps in practice. Hence, the research on such visualizations is of high relevance.

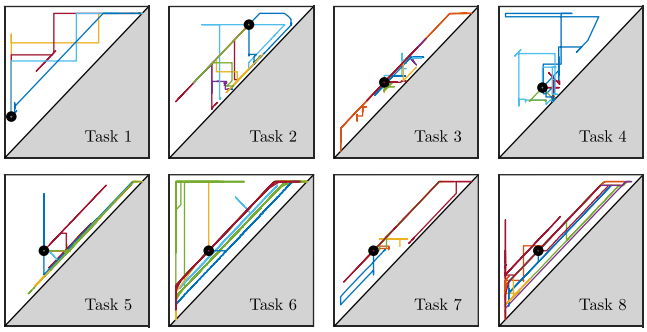


Fig. 12. Examples of query paths recorded during the study. The query path of each participant is illustrated in a different color. The starting point of the query path is symbolized with a black disc. See Fig. 7 for the interpretation of the paths' segments. .

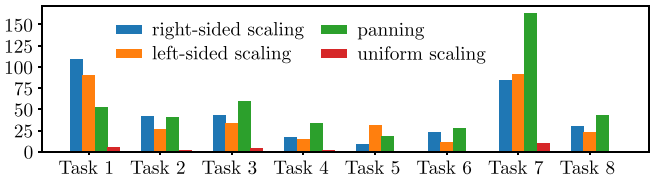


Fig. 13. Number of basic interactions per participant.

In the free-text answers the main criticism is that the number of events shown on the map should better reflect the distribution shown in the histogram. We see this as an interesting alternative objective for the scenario that the user wants to explore the spatio-temporal patterns. In Section VII, we will discuss how such an objective can be incorporated into our model. However, for the use-case of finding one event, e.g., hotel-use-case, we deem that the objective MAXINFO supports the user's needs better. One can also support the visualization of spatio-temporal patterns by displaying all events in the queried time-window as points on the map.

User Strategies for Tasks: Fig. 12 shows examples of recorded query paths. For each task, different strategies can be recognized. For example, in Task 1 some participants preferred left-sided and right-sided scaling (horizontal and vertical segments), while others preferred to first pan the window (diagonal segment). Fig. 13 shows that mostly left-sided and right-sided scaling as well as panning are used as basic interactions. Uniform scaling was rarely used, which reflects some of the comments that the interaction was less intuitive. One could therefore also omit uniform scaling from the set of basic interactions.[2] The average processing time and the error rate indicate that the tasks were solvable but were still demanding; see Table IV. We had a detailed look at Task 6, as several participants did not solve it correctly. It turned out that the possible choices of the answer were misleading, which often resulted in answers that were almost correct. On account of these observations, we conclude that with this study we have obtained a set of target-oriented query paths that are suitable for evaluating our algorithms.

[2]We note that this would not change our analysis, in particular, optimal activity regions will still be rectangles.

TABLE IV
THE PERFORMANCE OF THE 41 PARTICIPANTS, I.E., THE NUMBER OF CORRECT ANSWERS AND THE AVERAGE RESPONSE TIMES (IN SECONDS)

Task	1	2	3	4	5	6	7	8
Correct	37	32	38	36	28	24	36	41
Avg. time	47	48	33	84	73	97	20	25

Acquired Query Paths: Overall the query set comprises 328 query paths that consist of 136462 time-window queries in total. More specifically we recorded each time-window query sent by the discrete sampling of the web interface during the study.[3] From these time-window queries, we computed the basic interactions. Over all query paths, we obtained 10997 basic interactions.

D. Query Phase Evaluation

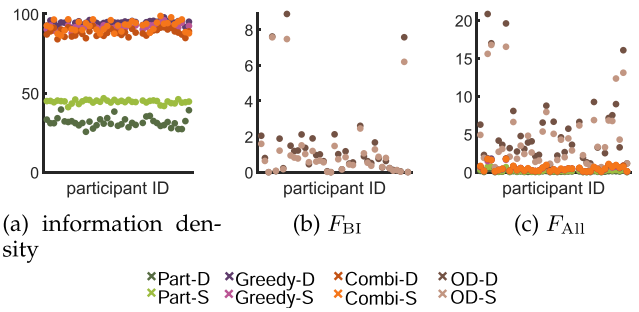
We analyze the information density and consistency with respect to the query paths generated by the study. We compare our presented algorithms to an *on-demand* solution, in which we compute each labeling for each time-window query without considering REPRODUCIBILITY, STABILITY, or CONTAINMENT. This is a well-established approach and we use it as our baseline for comparison. In more detail, given a time-window query we solve a maximum-weight independent set problem for the set of labels for all events contained in the time window. We obtained these labelings by an ILP formulation and denote them by OD-D and OD-S for unit disks and unit squares as labels, respectively. We build the evaluation on the data described in the study; see Section VI-C.

For Task 7 the ILP formulations for OD-S could not be solved in a reasonable time. When analyzing the query paths of Task 7, large time-window queries occurred, which led to large instances of the ILP formulation and high running times. We therefore have excluded this task for square labels.

Information Density: For each query Q issued in the study, we compare the weight of the labeling received from our precomputed data structure with the weight of an on-demand labeling that we optimized for Q without consideration of consistency (OD-D, OD-S); see Fig. 14(a) for Task 4. We refer to the ratio of these weights as *information density*. For both data sets the solutions of Greedy-D reach at least an information density of 81.92% and the ones of Greedy-S at least 78.86%. The algorithms Combi-D and Combi-S reach 62.33% and 69.54%, respectively. Part-D and Part-S drop behind with at least 16.97% and 20.44%. Hence, concerning information density the greedy heuristic is the best choice.

Consistency: For evaluating the consistency of the time-window labelings, we look at the change of displayed labels between labelings of two consecutive time-window queries. More formally, let $E = \tilde{}\{e_1, \ldots, e_n\}$ be a set of events such that t_i is the timestamp, and ℓ_i the label of event e_i. The label ℓ_i *flickers* between two queries Q and Q' if the timestamp t_i is contained in both Q and Q', and ℓ_i is either contained in L_Q or $L_{Q'}$

[3]We used JavaScript for the implementation of the web interface and sent a time-window query for every triggered "mouse move" event.



(a) information density (b) F_{BI} (c) F_{All}

Fig. 14. Results for analysis of the query phase for Task 4. We compare the data structure obtained by the approximation algorithm (Part-D, Part-S), by the greedy heuristic (Greedy-D, Greedy-S), by the combination of the approximation and greedy (Combi-D, Combi-S) with on-demand implementations (OD-D, OD-S) that do not take any consistency criteria into account. (a) The ratio between the weight of the labeling with respect to the optimal weight obtained by OD-D and OD-S, respectively. (b) Average degree of flickering. (c) Average degree of flickering over all queries within a query path P.

but not in both sets. We define the *degree* $f_{Q,Q'}$ *of flickering* between two consecutive queries Q and Q' as the number of labels that flicker between Q and Q', i.e., $f_{Q,Q'} = |\{e_i \tilde{} \in \tilde{} E \tilde{} \mid \tilde{} \ell_i$ flickers between Q and $Q'\}|$. Hence, the smaller $f_{Q,Q'}$ is, the higher is the consistency of the visual transition from the time-window labeling L_Q to the time-window labeling $L_{Q'}$ preserved. We assess the effect of the properties STABILITY and CONTAINMENT by the average number of flickering effects per query within the basic interactions.

More specifically, let P be a query path that is partitioned into its basic interactions $I_1, \ldots, I_m$ and let $Q_1^j, \ldots, Q_{r_j}^j$ be the queries of a basic interaction I_j. The *average degree of flickering over all basic interactions for path* P is defined as

$$F_{\mathrm{BI}} = \frac{1}{m} \sum_{j=1}^{m} \sum_{i=2}^{r_j} f_{Q_{i-1}^j, Q_i^j}. \tag{5}$$

We note that since ILP-D, ILP-S, Part-D, Part-S, Greedy-D, and Greedy-S are based on rectangular activity ranges, we have $F_{\mathrm{BI}} = 0$ for each of these variants. Fig. 14(b) shows F_{BI} for OD-D and OD-S. Over all query paths of all participants and all tasks the maximal value of F_{BI} is 18.13 flickering effects per query for OD-D and 12.08 flickering effects per query for OD-S. Hence, within a basic interaction, which consists of 12.41 queries on average, the user encounters 1334.5 and 648.17 for OD-D and OD-S at maximum, respectively. This shows that STABILITY and CONTAINMENT eliminate unnecessary flickering effects. We further evaluate the design of our model by counting the flickering effects between two consecutive queries over the entire query path P consisting of the queries $Q_1, \ldots, Q_m$. Hence, we particularly consider the flickering effects that occur during the transition from one basic interaction to the next. The *average degree of flickering over all queries within a query path* P is defined as

$$F_{\mathrm{All}} = \frac{1}{m} \sum_{i=2}^{m} f_{Q_{i-1}, Q_i}. \tag{6}$$

Fig. 14(c) shows F_{All} for each query path generated by one user for Task 4. Over all query paths of all participants, the maximal value of F_{All} is 0.6 for Part-D, 0.61 for Part-S, 1.32 for Greedy-D, 1.24 for Greedy-S, 1.75 for Combi-D, 1.4 for Combi-S, 37.56 for OD-D, and 26.24 for OD-S. Hence, while in the on-demand solutions, many flickering effects can be observed, at least eighteen times fewer flickering effects can be observed for the constructed activity diagrams. Thus, for the target-oriented usage of our interactive map, we observe a high consistency.

Query Time: In our implementation, we have used STR-packed R-trees for the query phase to process the rectangle-stabbing queries. A query took 3 μs on average and 86 μs at maximum over all considered activity diagrams and query paths. In contrast, computing the on-demand solutions OD-D and OD-S took substantially longer, e.g., for Task 6 we obtained 0.62 sec on average and 1.73 sec at maximum. We emphasize that the obtained query times of the data structure allow real-time applications as we have shown in our web application.

VII. CONCLUSION & OUTLOOK

We have presented a data structure for map labeling that ensures consistency during basic interactions with a time slider. As the underlying optimization problem is NP-hard, we focused on efficient non-exact algorithms. On the one hand, we developed approximation algorithms with a theoretical quality guarantee: the weight of the returned solution is at least a certain constant fraction ($1/4$ in the case of unit-square labels and $1/7$ in the case of unit disks) of the weight of an optimal solution. On the other hand, we developed an efficient greedy heuristic. Although the greedy heuristic does not have a constant approximation factor, it performs astonishingly well on real-world instances. In our experiments, the greedy heuristic achieved at least 83.41% of the quality of an optimal solution. We showed that our approach is efficient enough to compute the data structure and support queries for real-world applications. The evaluation of the generated activity diagrams for sequences of queries stemming from our study shows that our model preserves consistency between two consecutive time-window labelings while maintaining a high information density. We see a large potential for future research on map labeling in the context of dynamic query interfaces for spatio-temporal data.

- Motivated by the user feedback, we consider it promising to explore alternative models where the set of displayed events reflects the distribution over space and time. This is important for scenarios where the user wants to explore spatio-temporal patterns. We deem that we can formalize this property by adjusting the weights. Events that have many temporally and spatially close events get a large weight and events that are temporally and spatially isolated get a small weight.
- We presented a study for the generation of query paths. We deem it to be important to conduct an extensive user study that evaluates whether our model properties REPRODUCIBILITY, STABILITY, CONTAINMENT, and MAXINFO improve the user performance for solving user tasks.

- From the perspective of human-computer interaction an important question is how the interface is used for different tasks and what additional information could enhance the interaction. This would possibly have implications on the model and algorithm design.
- We consider it promising to extend our model to express that the priorities of the events depend on the query, e.g., because an event may be particularly characteristic for certain time spans.
- It would be interesting to revise the objective function or to introduce constraints ensuring that the shares of different event categories are preserved when selecting the events for a labeling.
- An important step is the integration of other types of map interactions such as zooming and panning.
- From the perspective of theoretical computer science it would be interesting to find algorithms with better approximation guarantees than those of our partitioning-based algorithms.

REFERENCES

[1] X. Zhang, S. Poon, S. Liu, M. Li, and V. C. S. Lee, "Consistent dynamic map labeling with fairness and importance," *Comput. Aided Geometric Des.*, vol. 81, 2020, Art. no. 101892.

[2] D. Bahrdt et al., "Growing balls in $\mathbb{R}^d$," in *Proc. 19th Workshop Algorithm Eng. Experiments*, SIAM, 2017, pp. 247–258.

[3] D. Peng, A. Wollf, and J. Haunert, "Finding optimal sequences for area aggregation: A* vs. integer linear programming," *ACM Trans. Spatial Algorithms Syst.*, vol. 7, no. 1, pp. 4:1–4:40, 2020.

[4] M. Meijers, P. van Oosterom, M. Driel, and R. Šuba, "Web-based dissemination of continuously generalized space-scale cube data for smooth user interaction," *Int. J. Cartogr.*, vol. 6, no. 1, pp. 152–176, 2020.

[5] G. L. Andrienko and N. V. Andrienko, "Interactive maps for visual data exploration," *Int. J. Geographical Inf. Sci.*, vol. 13, no. 4, pp. 355–374, 1999.

[6] P. Yoeli, "The logic of automated map lettering," *Cartographic J.*, vol. 9, no. 2, pp. 99–108, 1972.

[7] K. Been, E. Daiches, and C. Yap, "Dynamic map labeling," *IEEE Trans. Vis. Comput. Graph.*, vol. 12, no. 5, pp. 773–780, Sep./Oct. 2006.

[8] S. T. Leutenegger, M. A. Lopez, and J. Edgington, "STR: A simple and efficient algorithm for R-tree packing," in *Proc. 13th Int. Conf. Data Eng.*, 1997, pp. 497–506.

[9] D. A. Keim, G. L. Andrienko, J. Fekete, C. Görg, J. Kohlhammer, and G. Melançon, "Visual analytics: Definition, process, and challenge," in *Information Visualization - Human-Centered Issues and Perspectives*. Berlin, Germany: Springer, 2008, pp. 154–175.

[10] G. L. Andrienko et al., "Space, time and visual analytics," *Int. J. Geographical Inf. Sci.*, vol. 24, no. 10, pp. 1577–1600, 2010.

[11] G.-D. Sun, Y.-C. Wu, R.-H. Liang, and S.-X. Liu, "A survey of visual analytics techniques and applications: State-of-the-art research and future challenges," *J. Comput. Sci. Technol.*, vol. 28, pp. 852–867, 2013.

[12] M.-J. Kraak, "Geovisualization illustrated," *ISPRS J. Photogrammetry Remote Sens.*, vol. 57, no. 5/6, pp. 390–399, 2003.

[13] M. Williams, W. Kuhn, and M. Painho, "Interactive maps: What we know and what we need to know," *J. Spatial Inf. Sci.*, vol. 6, no. 1, pp. 59–115, 2013.

[14] K. A. Cook and J. J. Thomas, *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. Piscataway, NJ, USA: IEEE Computer Society, 2005. [Online]. Available: https://www.osti.gov/biblio/912515

[15] G. L. Andrienko et al., "Geovisual analytics for spatial decision support: Setting the research agenda," *Int. J. Geographical Inf. Sci.*, vol. 21, no. 8, pp. 839–857, 2007.

[16] N. V. Andrienko, G. L. Andrienko, and P. Gatalsky, "Exploratory spatio-temporal visualization: An analytical review," *J. Vis. Lang. Comput.*, vol. 14, no. 6, pp. 503–541, 2003.

[17] J. W. Crampton, "Interactivity types in geographic visualization," *Cartogr. Geographic Inf. Sci.*, vol. 29, no. 2, pp. 85–98, 2002.

[18] T. Kapler and W. Wright, "Geotime information visualization," *Inf. Visual.*, vol. 4, no. 2, pp. 136–146, 2005.

[19] C. Ahlberg, C. Williamson, and B. Shneiderman, "Dynamic queries for information exploration: An implementation and evaluation," in *Proc. Conf. Hum. Factors Comput. Syst.*, 1992, pp. 619–626.

[20] C. Ahlberg and B. Shneiderman, "Visual information seeking: Tight coupling of dynamic query filters with starfield displays," in *The Craft of Information Visualization*. San Mateo, CA, USA: Morgan Kaufmann, 2003, pp. 7–13.

[21] H. Hochheiser and B. Shneiderman, "Dynamic query tools for time series data sets: Timebox widgets for interactive exploration," *Inf. Sci.*, vol. 3, no. 1, pp. 1–18, 2004.

[22] A. C. Robinson, D. J. Peuquet, S. Pezanowski, F. A. Hardisty, and B. Swedberg, "Design and evaluation of a geovisual analytics system for uncovering patterns in spatio-temporal event data," *Cartogr. Geographic Inf. Sci.*, vol. 44, no. 3, pp. 216–228, 2017.

[23] S. Burigat and L. Chittaro, "Visualizing the results of interactive queries for geographic data on mobile devices," in *Proc. 13th ACM Int. Workshop Geographic Inf. Syst.*, 2005, pp. 277–284.

[24] E. Tanin, R. Beigel, and B. Shneiderman, "Incremental data structures and algorithms for dynamic query interfaces," *SIGMOD Rec.*, vol. 25, no. 4, pp. 21–24, 1996.

[25] M. J. Bannister, C. DuBois, D. Eppstein, and P. Smyth, "Windows into relational events: Data structures for contiguous subsequences of edges," in *Proc. 24th Annu. ACM-SIAM Symp. Discrete Algorithms*, SIAM, 2013, pp. 856–864.

[26] F. Chanchary and A. Maheshwari, "Time windowed data structures for graphs," *J. Graph Algorithms Appl.*, vol. 23, no. 2, pp. 191–226, 2019.

[27] F. Chanchary, A. Maheshwari, and M. Smid, "Querying relational event graphs using colored range searching data structures," *Discrete Appl. Math.*, vol. 286, pp. 51–61, 2019.

[28] M. J. Bannister, W. E. Devanny, M. T. Goodrich, J. A. Simons, and L. Trott, "Windows into geometric events: Data structures for time-windowed querying of temporal point sets," in *Proc. 26th Can. Conf. Comput. Geometry*, 2014.

[29] D. Bokal, S. Cabello, and D. Eppstein, "Finding all maximal subsequences with hereditary properties," in *Proc. Symp. Comput. Geometry*, 2015, pp. 240–254.

[30] T. Chan and S. Pratt, "Time-windowed closest pair," in *Proc. Can. Conf. Comput. Geometry*, 2015.

[31] T. Chan and S. Pratt, "Two approaches to building time-windowed geometric data structures," in *Proc. Symp. Comput. Geometry*, 2016, pp. 28:1–28:15.

[32] F. Chanchary, A. Maheshwari, and M. Smid, "Window queries for problems on intersecting objects and maximal points*," in *Proc. 4th Int. Conf. Algorithms Discrete Appl. Math.*, Springer, 2018, pp. 199–213.

[33] A. Bonerath, B. Niedermann, and J. Haunert, "Retrieving alpha-shapes and schematic polygonal approximations for sets of points within queried temporal ranges," in *Proc. 27th Int. Conf. Adv. Geographic Inf. Syst.*, 2019, pp. 249–258.

[34] A. Bonerath, B. Niedermann, J. Diederich, Y. Orgeig, J. Oehrlein, and J. Haunert, "A time-windowed data structure for spatial density maps," in *Proc. 28th Int. Conf. Adv. Geographic Inf. Syst.*, 2020, pp. 15–24.

[35] P. K. Agarwal, M. J. van Kreveld, and S. Suri, "Label placement by maximum independent set in rectangles," *Comput. Geometry*, vol. 11, no. 3/4, pp. 209–218, 1998.

[36] J. Haunert and A. Wolff, "Beyond maximum independent set: An extended integer programming formulation for point labeling," *ISPRS Int. J. Geo-Inf.*, vol. 6, no. 11, 2017, Art. no. 342.

[37] L. Barth, B. Niedermann, M. Nöllenburg, and D. Strash, "Temporal map labeling: A. new unified framework with experiments," in *Proc. 24th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2016, pp. 23:1–23:10.

[38] A. Gemsa, B. Niedermann, and M. Nöllenburg, "A unified model and algorithms for temporal map labeling," *Algorithmica*, vol. 82, no. 10, pp. 2709–2736, 2020.

[39] P. Bobák, L. Cmolík, and M. Cadík, "Temporally stable boundary labeling for interactive and non-interactive dynamic scenes," *Comput. Graph.*, vol. 91, pp. 265–278, 2020.

[40] K. Been, M. Nöllenburg, S.-H. Poon, and A. Wolff, "Optimizing active ranges for consistent dynamic map labeling," *Comput. Geometry*, vol. 43, no. 3, pp. 312–328, 2010.

[41] C. Liao, C. Liang, and S. Poon, "Approximation algorithms on consistent dynamic map labeling," *Theor. Comput. Sci.*, vol. 640, pp. 84–93, 2016.

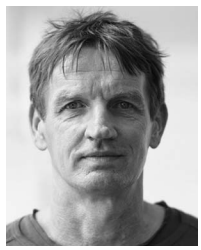
[42] N. Schwartges, D. Allerkamp, J. Haunert, and A. Wolff, "Optimizing active ranges for point selection in dynamic maps," in *Proc. 16th ICA Generalisation Workshop*, 2013, pp. 84–93.
[43] A. Gemsa, M. Nöllenburg, and I. Rutter, "Consistent labeling of rotating maps," *J. Comput. Geometry*, vol. 7, no. 1, pp. 308–331, 2016.
[44] A. Gemsa, M. Nöllenburg, and I. Rutter, "Evaluation of labeling strategies for rotating maps," *ACM J. Exp. Algorithmics*, vol. 21, no. 1, pp. 1.4:1–1.4:21, 2016.
[45] S. Funke, F. Krumpe, and S. Storandt, "Crushing disks efficiently," in *Proc. 27th Int. Workshop - Combinatorial Algorithms*, Springer, 2016, pp. 43–54.
[46] M. Chimani, T. C. van Dijk, and J. Haunert, "How to eat a graph: Computing selection sequences for the continuous generalization of road networks," in *Proc. 22nd ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2014, pp. 243–252.
[47] T. C. van Dijk, K. Fleszar, J. Haunert, and J. Spoerhase, "Road segment selection with strokes and stability," in *Proc. 1st ACM SIGSPATIAL Int. Workshop MapInteraction*, 2013, pp. 72–77.
[48] J. Gray et al., "Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub totals," *Data Min. Knowl. Discov.*, vol. 1, no. 1, pp. 29–53, 1997, doi: 10.1023/A:1009726021843.
[49] L. D. Lins, J. T. Klosowski, and C. E. Scheidegger, "Nanocubes for real-time exploration of spatiotemporal datasets," *IEEE Trans. Vis. Comput. Graph.*, vol. 19, no. 12, pp. 2456–2465, Dec. 2013.
[50] F. Miranda et al., "Time lattice: A data structure for the interactive visual analysis of large time series," *Comput. Graph. Forum*, vol. 37, no. 3, pp. 23–35, 2018, doi: 10.1111/cgf.13398.
[51] R. J. Fowler, M. Paterson, and S. L. Tanimoto, "Optimal packing and covering in the plane are np-complete," *Inf. Process. Lett.*, vol. 12, no. 3, pp. 133–137, 1981.
[52] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA, USA: Freeman., 1979.
[53] J. W.-T. Chan, F. Y. L. Chin, X. Hong, and H.-F. Ting, "Dynamic offline conflict-free coloring for unit disks," in *Proc. 6th Int. Workshop Approximation Online Algorithms*, Springer, 2008, pp. 241–252.
[54] E. W. Stienen et al., "Bird tracking–GPS tracking of lesser black-backed gulls and herring gulls breeding at the southern north sea coast. version 5.6," Research Institute for Nature and Forest (INBO), 2017. [Online]. Available: GBIF.org
[55] J. Haslett, R. Bradley, P. Craig, A. Unwin, and G. Wills, "Dynamic graphics for exploring spatial data with application to locating global and local anomalies," *Amer. Statistician*, vol. 45, no. 3, pp. 234–242, 1991.

Annika Bonerath received the BSc and MSc degrees in geodesy and geoinformation from the University of Bonn. She has been a researcher with the University of Bonn, Germany since 2018 and defended her PhD in 2024. Her research interests include computational geometry, geovisualization, and problems from automated cartography.

Anne Driemel received the PhD degree from Utrecht University, in 2013. After stops with the TU Dortmund and the TU Eindhoven she joined the University of Bonn, in 2018 as junior fellow with the Hausdorff Center for Mathematics and professor of computer science. She publishes in the research area of computational geometry where she also serves as editor and program committee member of internationally leading journals and conferences. Since 2022 She is co-chair of the computational geometry steering committee.

Jan-Henrik Haunert received the diploma and doctoral degree in geodesy and geoinformatics from the University of Hannover. He was a postdoctoral researcher with the Institute of computer science, University of Würzburg, and a professor for geoinformatics with the University of Osnabrück. In 2016, he took up a full professorship for geoinformation, University of Bonn. His research is concerned with the development of efficient algorithms for geovisualization and spatial analysis. In particular, he applies methods from combinatorial optimization and computational geometry to tasks of automated cartography, such as map generalization and cartographic label placement.

Herman Haverkort received the PhD degree in computer science from the University of Utrecht. After that he worked as a post-doctoral researcher with the Karlsruhe Institute of Technology and with Aarhus University, Denmark. From 2005 until 2018 he worked as a lecturer and researcher with the Eindhoven University of Technology in the Netherlands. Since 2018 he teaches and researches with the University of Bonn. His research interests include computational geometry, theory and applications of space-filling curves, and algorithms for geographic data analysis and cartography.



Elmar Langetepe received the MSc degree in mathematics from the University of Osnabrück, Germany, the PhD from degree from the University of Hagen, Germany and the Habilitation for computer science from the University of Bonn, Germany. He is a senior lecturer and researcher with the University of Bonn, Germany. His research interests are algorithms, computational geometry, algorithmic motion planning, and online algorithms.

Benjamin Niedermann received the PhD degree in computer science from the Karlsruhe Institute of Technology (KIT), Germany, in 2017. From 2017 to 2021 he was a member of the research group Geoinformation, University of Bonn. His research interests comprise the development of efficient algorithms in computational geometry, computational cartography, and geoprocessing. One main focus of his research is label placement in figures, maps, and dynamic scenes. It includes mathematical models, the design of algorithms with provable guarantees as well as the empirical evaluation of the algorithms in real-world scenarios. Currently, he is an engineer with yWorks working on algorithms for automatic graph drawing.