Algorithmica
https://doi.org/10.1007/s00453-020-00694-7

A Unified Model and Algorithms for Temporal Map Labeling

Andreas Gemsa[1] · **Benjamin Niedermann**[2] · **Martin Nöllenburg**[3]

Abstract
We consider map labeling for the case that a map undergoes a sequence of operations such as rotation, zoom and translation over a specified time span. We unify and generalize several previous models for dynamic map labeling into one versatile and flexible model. In contrast to previous research, we completely abstract from the particular operations and express the labeling problem as a set of time intervals representing the labels' presences, activities and conflicts. One of the model's strength is manifested in its simplicity and broad range of applications. In particular, it supports label selection both for map features with fixed position as well as for moving entities (e.g., for tracking vehicles in logistics or air traffic control). We study the active range maximization problem in this model. We prove that the problem is NP-complete and W[1]-hard, and present constant-factor approximation algorithms. In the restricted, yet practically relevant case that no more than k labels can be active at any time, we give polynomial-time algorithms as well as constant-factor approximation algorithms.

Keywords Dynamic map labeling · Unified map labeling model · Approximation algorithms · Complexity results · Geometric packing algorithms · Dynamic programming · Polynomial time algorithms

Preliminary versions of this paper have appeared as follows. *Trajectory-Based Dynamic Map Labeling In: Proc. 24th Int. Symp. on Algorithms and Computation (ISAAC'13), volume 8283 of Lect. Notes in Comput. Sci., pages 413–423. Springer, 2013* and *Temporal Map Labeling: A New Unified Framework with Experiments In: Proc. 24th ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems (GIS'16), pages 23:1–23:10, ACM, 2016.*

✉ Benjamin Niedermann
niedermann@uni-bonn.de

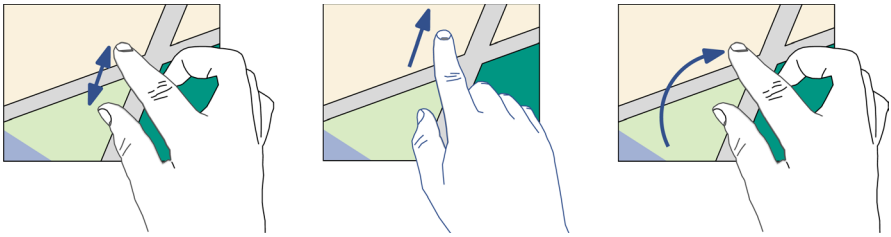
Martin Nöllenburg
noellenburg@ac.tuwien.ac.at

[1] Karlsruhe Institute of Technology, Karlsruhe, Germany

[2] University of Bonn, Bonn, Germany

[3] TU Wien, Vienna, Austria

Published online: 07 April 2020



(a) Zooming. **(b)** Panning. **(c)** Rotation.

Fig. 1 Illustration of interactive maps providing *zooming*, *panning*, and *rotation* as user interaction

1 Introduction

Dynamic digital maps are becoming more and more ubiquitous, especially with the rising numbers of location-based services and smartphone users worldwide. Consumer applications that include personalized and interactive map views range from classic navigation systems to map-based search engines and social networking services. Likewise, interactive digital maps are a core component of professional geographic information systems. All these map services have in common that the content of the map view is changing over time based on interaction with the system (i.e., *zooming*, *panning*, *rotating*, content filtering, etc.; see Fig. 1 for a schematic illustration of interactive maps) or based on the physical movement of the user or a set of tracked entities. Creating smooth visualizations under such map dynamics induces challenging geometric problems, e.g., continuous generalization [22] or *dynamic map labeling*.

In the past decade, dynamic map labeling has therefore captured the interest of researchers. In 2003, Petzold et al. [21] presented a framework for automatically placing labels on dynamic maps. They split the label placement procedure into two phases, namely a (possibly time-consuming) pre-processing phase and a query phase which computes the labeling of custom-scale maps. However, this approach does not guarantee that labels do not *jump* or *flicker* while transforming the map.

In 2006, Been et al. [1] introduced the first formal model for dynamic maps and dynamic labels, formulating a general optimization problem. They describe the change of a map by the operations *zooming*, *panning*, and *rotation*. They observe that the selection and placement of labels must be temporally coherent (or *consistent*) during all map animations resulting from interactions, rather than being optimized individually for each map view as in static map labeling. In order to avoid *flickering* and *jumping* labels while transforming the map with zooming and panning, they require four desiderata for consistent dynamic map labeling. These comprise *monotonicity*, i.e., labels should not vanish when zooming in or appear when zooming out (or any of the two when panning), *invariant point placement*, i.e., label positions and size remain invariant during movement, and *history independence*, i.e., placement and selection of labels should be a function of the current map state only. Monotonicity is modeled as selecting for each label at most one scale interval, the so-called *active range*, during which the label is

displayed. They introduce the *active range optimization problem* (ARO) maximizing the sum of active ranges over all labels such that no two labels overlap and all desiderata are fulfilled. They prove that ARO is NP-hard for star-shaped labels and present a greedy algorithm that computes an optimal solution for a simplified variant in polynomial time.

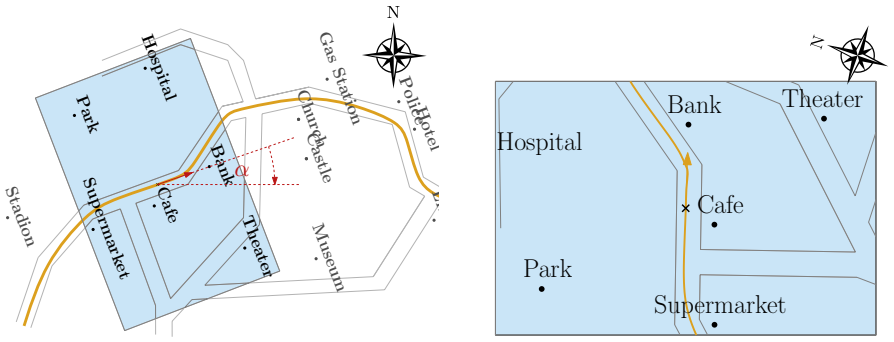
That model is the starting point for several subsequent papers considering the operations *zooming*, *panning* and *rotation*, mostly independently. Been et al. [4] take a closer look at different variants of ARO for zooming. They show NP-hardness and give approximation algorithms. In the same manner further variants are investigated by Liao et al. [14]. Gemsa et al. [9] present a fully polynomial-time approximation scheme (FPTAS) for a special case of ARO, where the given map is one-dimensional and only zooming is allowed. However, they combine the selection problem with a placement problem in a slider model. Zhang et al. [27] also consider the model of Been et al. [1] for zooming, however, instead of maximizing the total sum of active ranges, they maximize the minimum active range among all labels. They discuss similar variants as Liao et al. [14] and Been et al. [4], and also prove NP-hardness and give approximation algorithms.

Gemsa et al. [10, 11] extend the ARO model to *rotation* operations. They first show that the ARO problem is NP-hard in the considered setting and introduce an efficient polynomial-time-approximation scheme (FPTAS) for unit-height rectangles [10]. In subsequent work, they experimentally evaluate heuristics, algorithms with approximation guarantees and optimal approaches based on integer linear programming [11]. A similar setting for rotating maps is considered by Yokosuka and Imai [26]. However, instead of ARO, they aim at finding the maximum font size for which all labels can always be displayed without overlapping.

Apart from the results based on the consistency model of Been et al. [1], other approaches and models are considered, too. Maass et al. [17] describe a view management system for interactive three-dimensional maps of cities also considering label placement. Mote [20] presents a fast label placement strategy without a preprocessing phase. Luboschik et al. [15] describes a fast particle-based strategy that locally optimizes the label placement. None of these approaches takes consistency criteria for dynamic map labeling into account.

A different generalization of static point labeling is dynamic point labeling. Instead of transforming the map, the point set changes over time by adding or removing points as well as by moving points continuously. Inspired by air-traffic control, De Berg and Gerrits [6] consider moving points on a static map that must be labeled. They present a sophisticated heuristic for finding a reasonable trade-off between label speed and label overlap. Finally, Buchin and Gerrits [2] show that dynamic point labeling is strongly PSPACE-complete.

Embedded labels for road maps are also considered for interactive and dynamic maps. Maass and Döllner [18] provide a heuristic for labeling interactive 3D road maps taking obstacles into account. Vaaraniemi et al. [25] present a study on a force-based labeling algorithm for dynamic maps considering both point and line features. Schwartges et al. [24] investigate embedded labels in interactive maps allowing panning, zooming and rotation of the map. They evaluate a simple heuristic for maximizing the number of placed labels. In [23] Schwartges et al. take another approach



(a) Overall view. **(b)** Viewport.

Fig. 2 Trajectory-based labeling. **a** Viewport moves and aligns along a given trajectory (fat orange line). Labels align to the viewport. **b** The user's view on the scene

using *billboards* (labels with short leaders) for naming roads in interactive 3D maps to avoid label distortion.

Contribution and Outline Most existing algorithmic results in dynamic map labeling take a *global* view on the map, which optimizes over the *whole* interaction space, regardless of which portion of that space is actually explored by the user. In this paper we take a more *local* view on dynamic map labeling. Our aim is to develop algorithms that optimize the labeling for a specific map animation given offline as input. Any feature or label that is not relevant for that particular animation—for example, because it never enters the map view—can be ignored by our algorithms. This approach not only allows us to compute better labelings by removing unnecessary dependencies and non-local effects, but it also reduces the problem size, since fewer features and labels must be taken into account.

In Sect. 2, we first formulate an abstract, generic model for offline, temporal labeling problems, in which labels and potential conflicts between labels are represented as intervals over time. To represent a label's presence, we use a *presence* interval, which corresponds to the time that a label is present (but not necessarily displayed) in the map view. That is, whenever a label enters the map view, a corresponding presence interval starts, and whenever a label leaves the view, its current presence interval ends. Next, a *conflict interval* (or simply *conflict*) between two present labels starts and ends at the points in time at which the two labels start and stop intersecting. A temporal labeling is then simply represented as a set of subintervals—the labels' *activity intervals*, during which the labels are displayed such that no two conflicting labels are displayed simultaneously. Depending on the objective and additional consistency constraints of the labeling model, different sets of subintervals may be chosen by the algorithm.

This is a very versatile model, which includes, for instance, map labeling for car navigation systems, in which the map view changes position, angle, and scale according to the car's position, heading, and speed following a particular route; see Fig. 2. To give another, seemingly different example, it also includes the problem of labeling a set of moving entities in a map view (e.g., for tracking vehicles in logistics

applications or planes in air traffic control). Also non-map related applications such as labeling 3D scenes as they occur in medical information systems is covered by our model. Put differently, the model comprises any application in which start and end times of label presences and conflicts can be determined in advance. Further, the conflicts are not restricted to label–label conflicts but may also include label–object conflicts. This flexibility of the abstract model comes at the cost of concealing the geometric information behind the particular problem settings. Hence, one might find faster algorithms for the concrete geometric problem setting. On the other hand, our approach is easily implementable for rather different problem settings. In particular, it splits the problem into two independent sub-problems. The first is about transforming a particular geometric instance into intervals and the second is about finding good activity ranges based on these intervals. We deem both problems to be two relevant research challenges.

Based on this model we introduce the two optimization problems GeneralMaxTotal and k-RestrictedMaxTotal. While in the former problem we simply maximize the number of visible labels integrated over time, in the latter problem we further require that at most k labels are active at any time for some constant k. We note that limiting the number of simultaneously active labels is of practical interest as to avoid overly dense labelings, in particular for dynamic maps on small-screen devices such as in car navigation systems. We further refine our model by the three activity models AM1, AM2 and AM3, which introduce special requirements to enforce temporal consistency avoiding flickering when switching labels on and off.

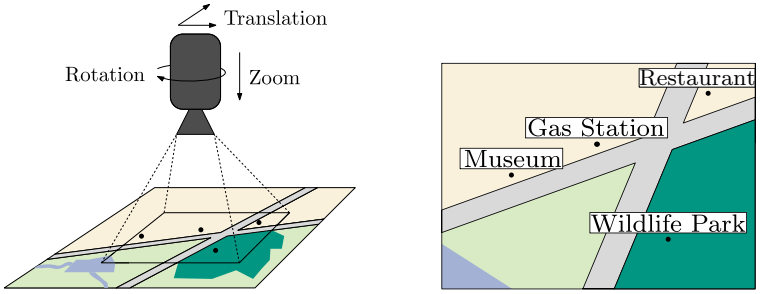
In Sect. 3, we investigate the problem GeneralMaxTotal. We first prove that GeneralMaxTotal is NP-complete; in fact it is even W[1]-hard and thus it is unlikely that a fixed-parameter tractable algorithm exists. For the special case of unit-square labels, we give an efficient approximation algorithm with different approximation ratios depending on the actual label activity model. In Sect. 4 we present polynomial-time algorithms for k-RestrictedMaxTotal in AM1 and AM2. For AM3 we show that the problem is NP-hard for $k \geq 2$. Further, for k-RestrictedMaxTotal we present efficient constant-factor approximation algorithms for all three activity models assuming that all labels are unit squares.

2 Model

In this section we formally describe our temporal labeling model. In particular, it unifies and generalizes the models presented by Been et al. [1] and by de Berg and Gerrits [6].

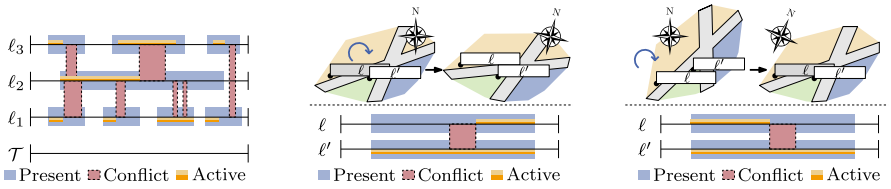
2.1 Basic Model

We are given a set $O = \{o_1, \ldots, o_n\}$ of objects in a scene over a given time span $\mathcal{T} = [0, 1]$. Further, for each object o we are given a label ℓ, e.g., text describing o. We denote the set of labels by $L = \{\ell_1, \ldots, \ell_n\}$, where ℓ_i is the label of o_i.



(a) Overall view. **(b)** Viewport.

Fig. 3 Model. **a** Overall view of the scene. Depending on the rotation, translation, and zoom, the camera shoots a restricted part of the scene. The objects to be labeled are represented by black dots. **b** The corresponding viewport of the camera. The labels are placed near their objects



(a) Valid activity. **(b)** End of conflict. **(c)** Start of conflict.

Fig. 4 Label activity. The intervals illustrate presence, conflict and activity intervals. **a** A set of valid activity intervals. **b**, **c** The maps rotate clockwise, while the labels keep aligned horizontally. White labels are active, while gray labels are inactive. The witness label ℓ' justifies **b** the start **c** the end of ℓ's activity interval

To quantify the importance of a label, we define for each label $\ell \in L$ a positive weight $w_\ell \in \mathbb{R}^+$.

We have a restricted view on the scene through a camera, i.e., the objects are projected onto an infinite plane P such that we can only see a restricted section V of P, where V models the *viewport* of the camera; see Fig. 3 for an example. During the time interval $\mathcal{T}$, the objects are moving and the camera changes its perspective by changing its position, direction and zoom. We denote the plane P and the viewport V at time t by $P(t)$ and $V(t)$, respectively. Depending on the position of the object $o_i \in O$, each label ℓ_i has a certain shape and position on $P(t)$ at time t; we denote the geometric shape of ℓ at time t by $\ell(t)$. Following typical map labeling models we may assume that $\ell(t)$ is a (closed) rectangle enclosing the text; one may also consider other shapes. In the following, we introduce some further notations to describe the setting precisely.

According to the perspective and position of the camera, not every label $\ell(t)$ is contained in the viewport at time t. We say that a label ℓ is *present* at time t if $\ell(t)$ is (partly) contained in $V(t)$; that is, $\ell(t) \cap V(t) \neq \emptyset$. We assume that the time intervals, during which a label ℓ is present, are given by a set Ψ_ℓ of disjoint, closed sub-intervals of $\mathcal{T}$; see Fig. 4. For such an interval $[a, b] \in \Psi_\ell$ we also

write $[a,b]_\ell$ indicating that it belongs to ℓ. We denote the union of all those sets Ψ_ℓ by Ψ and assume that Ψ is a multi-set, as it may contain the same interval $[a, b]$ multiple times, where each occurrence of $[a, b]$ belongs to a different label.

Two labels ℓ and ℓ' are in *conflict* at time $t \in \mathcal{T}$, if the geometric shapes of both labels intersect, i.e., $\ell(t) \cap \ell'(t) \neq \emptyset$. We describe the occurrences of conflicts between two labels $\ell, \ell' \in L$ by a set of closed intervals, $C_{\ell,\ell'} = \{[a,b] \subseteq \mathcal{T} |$ ℓ and ℓ' are in conflict for all $t \in [a,b]$ and[a,b]is maximal$\}$. For such an interval $[a,b] \in C_{\ell,\ell'}$ we also write $[a,b]_{\ell,\ell'}$ indicating that it is a *conflict interval* between ℓ and ℓ'. We denote the set of all conflict intervals over all pairs of labels by the multi-set C.

To avoid overlaps between labels, we display a label ℓ only at certain times when no other displayed label overlaps ℓ; the label ℓ is said to be *active* at those times. We describe the activity of ℓ, by a set Φ_ℓ of disjoint intervals.[1] For such an interval $[a,b] \in \Phi_\ell$ we also write $[a,b]_\ell$ to indicate that the *activity interval* belongs to ℓ. The union of all activity intervals over all labels is denoted by the multi-set Φ.

We say that two activity (presence) intervals $[a,b]_\ell$ and $[c,d]_{\ell'}$ of two labels ℓ and ℓ' are in *conflict* if there is a time t in the intersection of the open intervals $(a,b) \cap (c,d)$ such that the labels ℓ and ℓ' are in conflict at t.

An instance of temporal labeling is then defined by the set L of labels, the set Ψ of presence intervals and the set C of conflict intervals. We thus completely abstract away the geometry of the problem, while all essential information of the temporal labeling instance is captured combinatorially in Ψ and C. In this paper, we primarily focus on conflict-free label selection, and therefore assume that Ψ and C are given as input. In [3] we describe how to construct Ψ and C for the specific application of navigation systems.

Similarly to Been et al. [1] we require the following temporal consistency criteria:

(C1) A label should not be set active and inactive repeatedly to avoid *flickering*.
(C2) The position and size of a label should be changed continuously, it should not *jump*.
(C3) Labels should not overlap.

We formalize these consistency criteria and say the activity set Φ is *valid* (see Fig. 4a) if

(R1) for each activity interval $I_\ell \in \Phi$ there is a presence interval $I'_\ell \in \Psi$ with $I_\ell \subseteq I'_\ell$,
(R2) for each presence interval $I'_\ell \in \Psi$ there is at most one activity interval $I_\ell \in \Phi$ with $I_\ell \subseteq I'_\ell$, and
(R3) no two activity intervals of Φ are in conflict.

[1] Technically, one needs to distinguish between open and closed intervals, i.e., for closed rectangular labels, the presence and conflict intervals are closed but the activity intervals are open. However, including or excluding the interval boundaries makes no difference in our algorithms and hence we decided to simply use the notation $[a, b]$ for all respective intervals unless stated otherwise.

(a) GeneralMaxTotal **(b)** 10-RestrictedMaxTotal

Fig. 5 Frame of a dynamic map labeling. While in **a** 54 labels are displayed at the same time, in **b** 10 labels are displayed in order to limit the informational content

Requirement (R1) enforces that a label is only displayed if it is present in the viewport. Requirement (R2) prevents a label from *flickering* during a presence interval (C1), while (R3) enforces that no two displayed labels overlap (C3). In fact, (R2) is only a minimum requirement for avoiding flickering labels, which we later extend to stronger variants. By assuming that labels' positions are fixed relative to their anchors, labels may not *jump* (C2) as long as labeled objects are either fixed or move continuously. From now on we assume that an activity set is valid, unless stated otherwise.

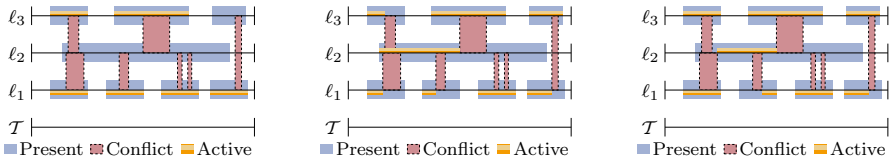
2.2 Optimization Problems

Based on the introduced model we investigate two optimization problems for temporal labeling that aim to maximize the overall active time of labels. The first problem allows for any number of labels to be active at the same time, and the second allows at most k labels to be active at the same time, which reduces the amount of presented information. We define the weight of an activity interval $[a, b]_\ell \in \Phi$ to be $w([a, b]_\ell) = (b - a) \cdot w_\ell$.

Problem 1 (GeneralMaxTotal)

Given: *Instance* (L, Ψ, C).
Find: *Activity set* Φ *maximizing* $\sum_{[a,b]_\ell \in \Phi} w([a, b]_\ell)$.

Figure 5a shows an example of a single frame of a temporal labeling that is optimal with respect to GeneralMaxTotal. While such a labeling is acceptable for general applications such as spatial data exploration, for small-screen devices, such as car navigation systems, the same labeling may overwhelm or distract the user with too much additional information. In fact, psychological studies have shown that untrained users are strongly limited in receiving, processing, and remembering information (e.g., see [19]). For applications that do not receive a user's full



(a) AM1 **(b)** AM2 **(c)** AM3

Fig. 6 The activity models AM1, AM2 and AM3

attention it is therefore desirable to restrict the number of simultaneously displayed labels (see Fig. 5b), which we formalize as an alternative optimization problem as follows.

Problem 2 (k-RestrictedMaxTotal)

Given: *Instance* (L, Ψ, C), $k \in \mathbb{N}$.
Find: *Activity set* Φ *maximizing* $\sum_{[a,b]_\ell \in \Phi} w([a,b]_\ell)$, *such that at any time* t *at most* k *labels are active.*

2.3 Activity Models

So far labels may become active or inactive within the viewport without any external influence, see, e.g., the second activity interval of ℓ_3 in Fig. 4a. Hence, the activity behavior of labels, even in an optimal solution Φ, is not necessarily explainable to a user by simple and direct observations such as "the label becomes inactive at time t, because at that moment an overlap starts with another active label". The absence of those simple logical explanations may lead to unnecessary irritations of the user. To account for that we introduce the concept of justified activity intervals.

Consider a label ℓ with activity interval $[a,b]_\ell \in \Phi$. We say that the start of $[a,b]_\ell$ is *justified* if ℓ enters the viewport at time a or if there is a *witness* label ℓ' such that a conflict of ℓ and ℓ' ends at a and ℓ' is active at a; see Fig. 4b.

Analogously, we say that the end of $[a,b]_\ell$ is *justified* if ℓ leaves the viewport at time b or if there is a witness label ℓ' such that a conflict of ℓ and ℓ' begins at b and ℓ' is active at b; see Fig. 4c. If both the start and end of $[a,b]_\ell$ are justified, then $[a,b]_\ell$ is justified.

We distinguish the three activity models AM1, AM2, and AM3 that consider justified activity intervals; see Fig. 6. While for AM1, a label may only become active and inactive when it enters and leaves the viewport, for AM2 it may also become inactive before leaving the viewport if a witness label justifies this event. AM3 further allows a label to become active after entering the viewport if a witness label justifies that event.

AM1 An activity set Φ satisfies AM1 if any activity interval $[a,b]_\ell \in \Phi$ is justified and there is a presence interval $[c,d]_\ell \in \Psi$ of the same label ℓ with $[a,b]_\ell = [c,d]_\ell$.

AM2 An activity set Φ satisfies AM2 if any activity interval $[a,b]_\ell \in \Phi$ is justified and there is a presence interval $[c,d]_\ell \in \Psi$ of the same label ℓ with $a = c$.

AM3 An activity set Φ satisfies AM3 if any activity interval $[a,b]_\ell \in \Phi$ is justified. For AM2 one might also consider the symmetric variant in which the activity interval starts in the middle of the presence interval and ends with the presence interval; due to symmetry we do not consider the variant explicitly. We have described only the core of the model. Depending on the application it can be easily extended to more complex variants, e.g., requiring minimum durations of activity intervals.

3 Solving GeneralMaxTotal

In this section we discuss GeneralMaxTotal. First, we present results on its computational complexity and then we present a simple approximation algorithm for the case that the labels are unit squares.

3.1 Computational Complexity

We first prove that the corresponding decision problem of GeneralMaxTotal is NP-complete with respect to the three activity models. The membership in NP follows from the fact that the start and the end of an active interval may be assumed to coincide with the start or end of a presence interval or a conflict interval. Thus, there is a finite number of candidates for the endpoints of the active intervals so that a solution $\mathcal{L}$ can be guessed. Verifying that $\mathcal{L}$ is valid in one of the three models and that its value exceeds a given threshold can obviously be checked in polynomial time.

For the NP-hardness we apply a straight-forward reduction from the NP-complete maximum independent set of rectangles problem [7]. We simply interpret the set of rectangles as a set of labels with unit weight, choose a short vertical trajectory T and a viewport R that contains all labels at any point of T. Since the conflicts do no change over time, the reduction can be used for all three activity models. By means of the same reduction and Marx' result [16] that finding an independent set for a given set of axis-parallel unit squares is W[1]-hard we derive the next theorem.

Theorem 1 *GeneralMaxTotal is* NP*-hard and* W[1]*-hard for all activity models AM1, AM2, AM3. In particular, the respective decision problem is* NP*-complete for all activity models AM1, AM2, AM3.*

As a consequence, GeneralMaxTotal is not fixed-parameter tractable unless W[1]=FPT. Note that this also means that for k-RestrictedMaxTotal we cannot expect to find an algorithm that runs in $O(f(k) \cdot n^c)$ time, where c is a constant, n is the number of presence and conflict intervals, and $f(k)$ is a computable function that

depends only on the parameter k. Assume for the sake of contradiction that there is an algorithm $\mathcal{A}_k$ for k-RestrictedMaxTotal with $O(f(k) \cdot n^c)$ running time. We then could use $\mathcal{A}_k$ to check whether the input set of unit squares contains an independent set of size κ as follows. We use the same reduction as above and determine the largest $k \in \{1, \ldots, \kappa\}$ for which the solution of $\mathcal{A}_k$ contains exactly k labels. This particularly implies that for every k' with $k' > k$ the algorithm $\mathcal{A}_{k'}$ also yields a solution with k labels. If $k = \kappa$, this implies that the input instance contains an independent set of κ unit squares, which contradicts that selecting κ intersection-free unit squares from a set of intersecting unit squares is $\mathtt{W}[1]$-hard.

3.2 Approximation of GeneralMaxTotal

We now describe a simple greedy algorithm for GeneralMaxTotal in all three activity models assuming that all labels are unit squares anchored at their lower-left corner. We assume that the map changes via the standard operations panning, rotation and zooming. Further, the weight of each presence interval $[a, b]_\ell$ is its length $w([a, b]_\ell) = b - a$.

Starting with an empty solution Φ, our algorithm GreedyMaxTotal removes the maximal interval I from Ψ and adds it to Φ, i.e., the label of I is set active during I. Then, depending on the activity model, it updates all presence intervals that have a conflict with I in Ψ and continues until the set Ψ is empty.

For AM1 the update process simply removes all presence intervals from Ψ that are in conflict with the newly selected interval I. For AM2 and AM3 let $I_j \in \Psi$ and let $I_j^1, \ldots, I_j^k$ be the longest disjoint sub-intervals of I_j for some $k \geq 0$ that are not in conflict with the selected interval I. We assume that $I_j^1, \ldots, I_j^k$ are sorted by their left endpoint. The update operation for AM2 replaces every interval $I_j \in \Psi$ that is in conflict with I with I_j^1. As I_j^1 is a prefix of I_j, selecting I_j in a later step satisfies the condition that the label becomes active with the beginning of its presence interval. Further, the conflict with I justifies the end of I_j^1. In AM3 we replace I_j by I_j^1 if I_j^1 is not fully contained in I. Otherwise, I_j is replaced by I_j^k. Note that this discards some candidate intervals, but the chosen replacement of I_j is enough to prove the approximation factor. We observe that due to the conflicts with I (and possibly other intervals chosen in a previous step) the new interval is justified. Note that after each update all intervals in Ψ are valid choices according to the specific model. Hence, we can conclude that the result Φ of GreedyMaxTotal is also valid in that model.

In the following, we analyze the approximation quality of GreedyMaxTotal. To that end we first introduce a purely geometric packing lemma. Similar packing lemmas have been introduced before, but to the best of our knowledge for none of them it is sufficient that only one prescribed corner of the packed objects lies within the container.

Lemma 1 *Let C be a circle of radius $\sqrt{2}$ in the plane and let Q be a set of non-intersecting closed and axis-parallel unit squares with their bottom-left corner in C. Then Q cannot contain more than eight squares.*

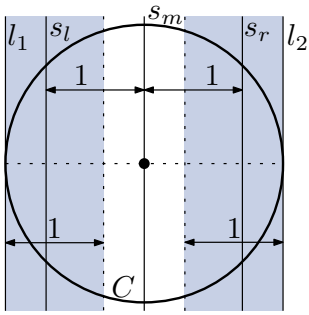


Fig. 7 Illustration for the proof of Lemma 1

Proof First, we show that Q cannot contain more than nine squares and extend the result to the claim of the lemma. We begin by proving the following claim.

(S) At most three squares of Q can be stabbed by a vertical line. In order to prove (S) let $Q' \subseteq Q$ be a set of squares that is stabbed by an arbitrary vertical line l and let q_t be the topmost square stabbed by l and let q_b be the bottommost square stabbed by l. Since both the bottom-left corner of q_t and q_b are in C, their vertical distance is at most $2\sqrt{2}$. Consequently, there can be at most one other square in Q' that lies in between q_t and q_b, which shows the claim (S).

Now let l_1 be the left vertical tangent of C and let l_2 be its right vertical tangent; see Fig. 7. We define $Q_l \subseteq Q$ to be the set of squares whose bottom-left corner has distance of at most 1 to l_1. Hence, there is a vertical line that stabs all squares in Q_l. By (S) it follows that $|Q_l| \leq 3$. We can analogously define the set $Q_r \subseteq Q$ whose bottom-left corner has distance of at most one to the vertical line l_2. By the same argument it follows that $|Q_r| \leq 3$. Further, the bottom-left corners of the squares $Q_m = Q \setminus \{Q_l, Q_r\}$ are contained in a vertical strip of width $2\sqrt{2} - 2 < 1$. Hence, there is a vertical line that stabs all squares of Q_m and $|Q_m| \leq 3$ follows. We conclude that the set Q contains at most nine squares; in fact, $|Q| \leq 8$ as we show next.

For the sake of contradiction we assume that $|Q| = 9$, i.e., $|Q_l| = |Q_m| = |Q_r| = 3$. We denote the topmost square in Q_l by t_l and the bottommost square by b_l, and define t_r and b_r for Q_r analogously.

Further, let s_m be the vertical line through the center of C, let s_l be the vertical line that lies one unit to the left of s_m and let s_r be the vertical line that lies one unit to the right of s_m. Note that the length of the segment of s_l and s_r that is contained in C has length 2. Since the bottom-left corners of t_l and b_l have vertical distance strictly greater than 2, both squares lie to the right of s_l. Hence, t_l and b_l intersect s_m. Analogously, the bottom-left corners of t_r and b_r lie to the left of s_r, and, hence intersect s_r. The line s_m is intersected by two squares of Q_l. By (S) there can be at most one additional square of Q_m that intersects s_m. Thus, there are two squares in Q_m whose anchors lie to the right of s_m. But then they both intersect s_r which itself is already intersected by at least the squares t_r and b_r. This is a contradiction to (S), and concludes the proof. □

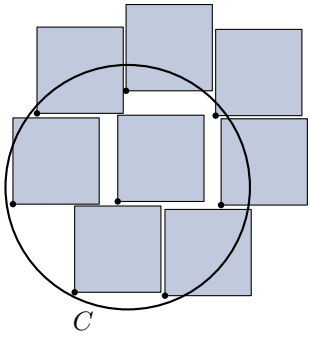


Fig. 8 Example configuration of eight axis-aligned, non-intersecting, unit-squares whose bottom-left corners lie inside a circle C of radius $\sqrt{2}$

Figure 8 shows that the bound is tight. Based on Lemma 1 we now show that for any label with anchor p there is no point of time $t \in [0, 1]$ for which there can be more than eight active labels whose anchors are within distance $\sqrt{2}$ of p. We call a set $X \subseteq \Psi$ *conflict-free* if it contains no pair of presence intervals that are in conflict. Further, we say that X is in conflict with $I \in \Psi$ if every element of X is in conflict with I, and we say that X contains $t \in [0, 1]$ if every element of X contains t.

Lemma 2 *For every $t \in [0, 1]$ and every $I \in \Psi$ any maximum cardinality conflict-free set $X_I(t) \subseteq \Psi$ that is in conflict with I and contains t satisfies $|X_I(t)| \leq 8$.*

Proof Assume that there is a time t and an interval I such that there is a set $X_I(t)$ that contains more than eight intervals. Let ℓ be the label that corresponds to I. For an interval $I' \in X_I(t)$ to be in conflict with I the anchors of the two corresponding labels must have a distance of at most $\sqrt{2}$. Hence, there are $|X_I(t)|$ labels corresponding to the intervals in $X_I(t)$ with anchors of distance at most $\sqrt{2}$ to the anchor of ℓ. By Lemma 1 we know that two of these labels must overlap. This implies that there is a conflict between the corresponding intervals contained in $X_I(t)$, which is a contradiction. □

With this lemma we can finally obtain the approximation guarantees for GREEDY-MAXTOTAL for all activity models.

Theorem 2 *Assuming that all labels are unit squares and $w([a, b]) = b - a$, GREEDY-MAXTOTAL is a 1/24-, 1/16-, 1/8-approximation for AM1–AM3, respectively, and needs $O(n \log n)$ time for AM1 and $O(n^2)$ time for AM2 and AM3.*

Proof To show the approximation ratios, we consider an arbitrary step of GREEDY-MAXTOTAL in which the presence interval $I = [a, b]_\ell$ is selected from Ψ. Let C^I_ℓ be the set of presence intervals in Ψ that are in conflict with I.

Consider the model AM1. Since I is the longest interval in Ψ when it is chosen, the intervals in C^I_ℓ are completely contained in $J = [a - w(I), b + w(I)]$. As C^I_ℓ contains all presence intervals that are in conflict with I, it is sufficient to consider J to bound the effect of selecting I. Obviously, the interval J is three times as long

as I. By Lemma 2 we know that for any $t \in J$ it holds that $|X_I(t)| \leq 8$. Hence, in an optimal solution there can be at most eight active labels at each point $t \in J$ that are discarded when $[a, b]_\ell$ is selected. Thus, the cost of selecting $[a, b]_\ell$ is at most $3 \cdot 8 \cdot w(I)$.

For AM2 we apply the same arguments, but restrict the interval J to $J = [a, b + w(I)]$, which is only twice as long as I. To see that consider for an interval $[c, d]_{\ell'} \in C^I_\ell$ the prefix $[c, a]$ if it exists. If $[c, a]$ does not exist (because $a < c$), removing $[c, d]_{\ell'}$ from Ψ changes Ψ only in the range of J. If $[c, a]$ exists, then again Ψ is only changed in the range of I, because by definition $[c, d]_{\ell'}$ is shortened to an interval that at least contains $[c, a]$ and is still contained in Ψ. Thus, the cost of selecting I is at most $2 \cdot 8w(I)$.

Analogously, for AM3 we can argue that it is sufficient to consider the interval $J = [a, b]$. By definition of the update operation of GreedyMaxTotal at least the prefix or suffix subinterval of each $[c, d]_{\ell'} \in C^I_\ell$ remains in Ψ that extends beyond I (if such an interval exists). Thus, selecting I influences only the interval J and its cost is at most $8w(I)$. The approximation bounds of 1/24, 1/16, and 1/8 follow immediately.

We use a binary search tree to achieve the time complexity $O(n \log n)$ of GreedyMaxTotal for AM1. We first add all intervals sorted by their weights to the binary tree. We then successively remove the intervals with maximal weight and its conflicting neighbors. As each interval is inserted and removed exactly once, we obtain $O(n \log n)$ running time. For AM2 and AM3 we use a linear sweep to identify the longest interval contained in Ψ. In each step we need $O(n)$ time to update all intervals in Ψ, and we need a total of $O(n)$ steps. Thus, GreedyMaxTotal needs $O(n^2)$ time in total for AM2 and AM3. □

In case that we exclude rotation as an operation and only consider panning and zooming, we can improve the approximation by a factor of two, as we can use a more restrictive packing argument instead of Lemma 1.

Observation 1 *Let S be a square with side length 2 in the plane and let Q be a set of non-intersecting closed and axis-parallel unit squares with their bottom-left corner in S. Then Q cannot contain more than four squares.*

With this observation we can strengthen Lemma 2 and thus Theorem 2 by a factor of two and hence obtain the following corollary.

Corollary 1 *Assuming that all labels are unit squares, the map operations are panning and zooming, and $w([a, b]) = b - a$, GreedyMaxTotal is a 1/12-, 1/8-, 1/4-approximation for AM1–AM3, respectively, and needs $O(n \log n)$ time for AM1 and $O(n^2)$ time for AM2 and AM3.*

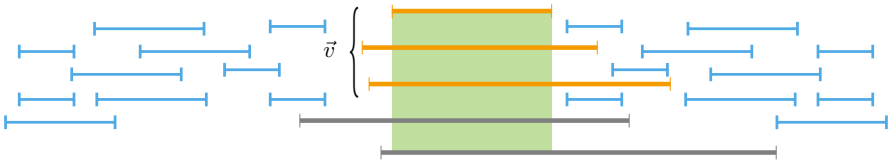


Fig. 9 The separating 3-tuple (orange) splits the set of presence intervals into a left and right subinstance (blue). The presence intervals (gray) intersecting $K_{\vec{v}}$ (green box) cannot be selected since at most three labels can be active at the same time (Color figure online)

4 Solving k-RestrictedMaxTotal

In this section we prove that unlike GeneralMaxTotal the problem k-RestrictedMaxTotal can actually be solved in polynomial time for AM1 and AM2. We give a detailed description of our algorithm for AM1, and then show how it can be extended to AM2. Note that solving k-RestrictedMaxTotal is related to finding a maximum cardinality k-colorable subset of n intervals in interval graphs. This can be done in polynomial time in both n and k [5]. However, we have to consider additional constraints due to conflicts between labels, which makes our problem more difficult. First, we discuss how to solve the case for $k = 1$ and then give an algorithm that solves k-RestrictedMaxTotal for $k > 2$. Altogether, we obtain an algorithm that runs in $O(n^{f(k)})$ time, where f is a polynomial function. Hence, k-RestrictedMaxTotal lies in the complexity class XP for AM1 and AM2. Due to the W[1]-hardness of GeneralMaxTotal, we cannot expect running times in $O(f(k) \cdot n^c)$, where f is a computable function and c is a constant. In contrast, for AM3 we prove that k-RestrictedMaxTotal is NP-hard for $k \geq 2$. Since the running times of the presented algorithms for AM1 and AM2 are, even for small k, prohibitively expensive in practice, we finally propose an approximation algorithm for k-RestrictedMaxTotal in all three activity models.

4.1 An Algorithm for k-RestrictedMaxTotal in AM1

In this section we present a dynamic programming approach for computing an optimal solution for an instance (L, Ψ, C) of k-RestrictedMaxTotal in AM1. The approach is based on the idea to systematically identify and solve smaller subinstances that are independent from each other. As separators of these sub-instances we systematically explore all choices of k intervals that have a point in time in common. More precisely, a *separating k-tuple* $\vec{v}$ is a set of k presence intervals that are not in conflict with each other and that have a non-empty intersection $K_{\vec{v}} = \bigcap_{I \in \vec{v}} I$. The weight of a separating tuple $\vec{v}$ is defined as $w(\vec{v}) = \sum_{I \in \vec{v}} w(I)$. We observe that a separating k-tuple $\vec{v}$ contained in a solution of k-RestrictedMaxTotal splits the set of presence intervals into two independent subsets. Specifically, a left (right) subset $L_k[\vec{v}] \subseteq \Psi$ ($R_k[\vec{v}] \subseteq \Psi$) that contains only intervals which lie completely to the left (right) of $K_{\vec{v}}$ and are not in conflict with any interval of $\vec{v}$; see Fig. 9.

Given two separating k-tuples $\vec{u}$ and $\vec{v}$ (where $K_{\vec{u}}$ lies to the left of $K_{\vec{v}}$) we present an algorithm $\mathcal{A}_k(\vec{u}, \vec{v})$ that computes an activity set $\Phi \subseteq L_k[\vec{v}] \cap R_k[\vec{u}]$ such that

(C1) no two intervals of $\Phi \cup \vec{u} \cup \vec{v}$ are in conflict,
(C2) at any time t at most k intervals of $\Phi \cup \vec{u} \cup \vec{v}$ contain t, and
(C3) the total activity $\sum_{I_\ell \in \Phi} w(I_\ell)$ is maximized.

We observe that with respect to these conditions the activity set Φ respects the separating tuples $\vec{u}$ and $\vec{v}$, but Φ does not contain any intervals of $\vec{u}$ and $\vec{v}$. We call $\vec{u}$ and $\vec{v}$ the *boundary separating k-tuples* of Φ.

We define the algorithm $\mathcal{A}_k$ recursively such that it uses $\mathcal{A}_{k-1}$ as subroutine. Initially, for the input instance we call $\mathcal{A}_k(d_l, d_r)$ where d_l and d_r are two dummy separating k-tuples such that the presence intervals of d_l are strictly to the left of 0 and the presence intervals of d_r are strictly to the right of 1. Since all original presence intervals are completely contained in [0, 1], every optimal solution contains both dummy separating k-tuples.

So assume in the following that the algorithm $\mathcal{A}_k$ is called with the two separating k-tuples $\vec{u}$ and $\vec{v}$ and let $X = L_k[\vec{v}] \cap R_k[\vec{u}]$. We first explain how to solve the base case $k = 1$ and then present the algorithm for $k > 1$.

Case $k = 1$. First note that for the case that at most one label can be active at any given point in time ($k = 1$), conflicts between labels do not matter. Thus, for solving 1-RestrictedMaxTotal it is sufficient to find an independent subset of Ψ of maximum weight. This is equivalent to finding a maximum weight independent set on interval graphs, which can be done in $O(n^2)$ time using dynamic programming given n sorted intervals [13]. The algorithm basically computes a table $\mathcal{T}_1$ indexed by the intervals in X, where an entry $\mathcal{T}_1[I_j]$ stores the value of a maximum weight independent set Q of $L_1[I_j]$ and a pointer to the rightmost interval in Q. Hence, using this approach we obtain an activity set Φ. As no interval of X intersects any interval of the singletons $\vec{u}$ and $\vec{v}$, the set Φ also satisfies the conditions (C1)–(C3) from above.

Case $k > 1$. We say that a separating k-tuple $\vec{v}$ is *smaller* than a separating k-tuple $\vec{w}$ if $K_{\vec{v}}$ begins to the left of $K_{\vec{w}}$. Ties are broken arbitrarily. This lets us define the ordered set $S_k = \{\vec{v}_1, \ldots, \vec{v}_z\}$ of all separating k-tuples of X. In order to ease notation we denote the given boundary separating k-tuples $\vec{u}$ by $\vec{v}_0$ and $\vec{v}$ by $\vec{v}_{z+1}$. Further, we say a set C of presence intervals is *k-compatible* if no more than k intervals in C intersect at any point and there are no conflicts in C. Two separating k-tuples $\vec{x}$ and $\vec{y}$ are *k-compatible* if they are disjoint, i.e., they have no interval in common, and $\vec{x} \cup \vec{y}$ is k-compatible. Further, for a separating k-tuple $\vec{x}$ let $l_{k-1}(\vec{x})$ be the set of the $k-1$ intervals of $\vec{x}$ whose right end points are rightmost; in case that this is not uniquely defined, we choose any of those subsets. Analogously, let $r_{k-1}(\vec{x})$ be the set of the $k-1$ intervals of $\vec{x}$ whose left end points are leftmost. We observe that $l_{k-1}(\vec{x})$ and $r_{k-1}(\vec{x})$ are separating $k-1$-tuples; in the recursion step we use $l_{k-1}(\vec{x})$ and $r_{k-1}(\vec{x})$ as left and right boundary separating $k-1$-tuples, respectively.

The algorithm $\mathcal{A}_k$ fills a one-dimensional table $\mathcal{T}_k$. Similarly to the case $k = 1$, each entry $\mathcal{T}_k[\vec{v}]$ stores the value of the optimal solution for $L_k[\vec{v}]$. Then the solution of the given subinstance bounded by $\vec{v}_0$ and $\vec{v}_{z+1}$ can be obtained from $\mathcal{T}_k[\vec{v}_{z+1}]$. We

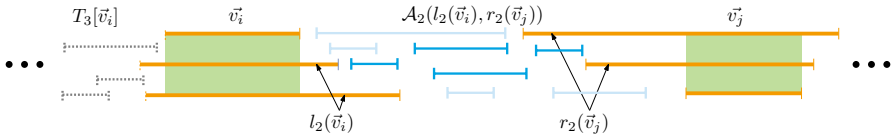


Fig. 10 Illustration for computing the table entry $\mathcal{T}_k[\vec{v}_j]$ for $k = 3$. The intervals $K_{\vec{v}_i}$ and $K_{\vec{v}_j}$ are marked in green. The dotted intervals illustrate the optimal solution represented by $\mathcal{T}_k[\vec{v}_i]$. The separating k-tuples (orange) enclose an instance for 2-RestrictedMaxTotal (blue). The algorithm $\mathcal{A}_2(l_2(\vec{v}_i), r_2(\vec{v}_j))$ computes an optimal solution (blue intervals) for that instance respecting $\vec{v}_i$ and $\vec{v}_j$ (Color figure online)

initialize $\mathcal{T}_k[\vec{v}_0] = 0$. Then, the remaining entries of $\mathcal{T}_k$ can be obtained by computing (see also Fig. 10)

$$\mathcal{T}_k[\vec{v}_j] = \max_{i<j}\{\mathcal{T}_k[\vec{v}_i] + w(\vec{v}_i) + w(\mathcal{A}_{k-1}\left(l_{k-1}(\vec{v}_i), r_{k-1}(\vec{v}_j)\right)) \mid \vec{v}_i \in S_k,\ \vec{v}_i \subseteq L_k[\vec{v}_j] \cup \vec{v}_0,$$
$$\vec{v}_j \subseteq R_k[\vec{v}_i] \cup \vec{v}_{z+1}, \vec{v}_0 \cup \vec{v}_{z+1} \cup \vec{v}_i \cup \vec{v}_j \text{ is } k\text{-compatible}\}.$$

Apart from $\mathcal{T}_k[\vec{v}_j]$ we also store for each separating k-tuple v_j the separating k-tuple v_i for which the expression above is maximized; we define $\text{pred}(\vec{v}_j) = \vec{v}_i$. Starting a standard backtracking procedure at $\vec{v}_{z+1}$ we obtain a activity set $\Phi(\vec{v}_{z+1})$ recursively defined by

$$\Phi(\vec{x}) = \begin{cases} \emptyset & \text{if } \vec{x} = \vec{v}_0, \\ \vec{x} \cup \Phi(\text{pred}(\vec{x})) & \text{otherwise.} \end{cases}$$

As the result of the algorithm we return the set $\Phi = \Phi(v_{z+1}) \setminus (\vec{v}_0 \cup \vec{v}_{z+1})$. Hence, Φ is an activity set for the instance X.

Theorem 3 *Algorithm $\mathcal{A}_k$ solves k-RestrictedMaxTotal in AM1 in $O(n^{k^2+k-1})$ time and $O(n^k)$ space.*

Proof We first show the correctness of $\mathcal{A}_k$ by induction on k. Given two separating k-tuples $\vec{u}$ and $\vec{v}$ (where $\vec{u}$ is smaller than $\vec{v}$) we prove that $\mathcal{A}_k(\vec{u}, \vec{v})$ computes an activity set $\Phi \subseteq L_k[\vec{v}] \cap R_k[\vec{u}]$ such that conditions (C1)–(C3) are satisfied.

For the case $k = 1$ this is clearly the case. So assume $k > 1$. Since $\mathcal{A}_k$ considers only solutions where adjacent separating k-tuples are k-compatible with each other and both boundary k-tuples, we cannot produce an invalid solution, i.e., a solution with conflicts or more than k active intervals at any point. Hence, both condition (C1) and (C2) are satisfied. We prove condition (C3) by contradiction. So assume that there is an instance X for which $\mathcal{A}_k$ does not compute an optimal solution and let OPT be an optimal solution. There must be a smallest separating k-tuple $\vec{v}_j$, $j > 0$, for which $\mathcal{T}_k[\vec{v}_j]$ is less than the value of OPT for $L_k[\vec{v}_j]$. Let $\vec{v}_i$, $i < j$ be the rightmost disjoint separating k-tuple in OPT that precedes $\vec{v}_j$ such that the set $\vec{v}_0 \cup \vec{v}_i \cup \vec{v}_j \cup \vec{v}_{z+1}$ is k-compatible. By our assumption $\mathcal{T}_k[\vec{v}_i]$ has the same value as OPT on $L_k[\vec{v}_i]$. For the set of intervals $L_k[\vec{v}_j] \cap R_k[\vec{v}_i]$ there are at most $k-1$ active intervals at any point (otherwise $\vec{v}_i$ is not rightmost). This means that when we run algorithm $\mathcal{A}_{k-1}$ on that instance with the boundary tuples $l_{k-1}(\vec{v}_i)$ and $r_{k-1}(\vec{v}_j)$, we obtain by the

induction hypothesis a solution that is at least as good as the restriction of OPT to that instance. Since $\vec{v}_i$ is a valid predecessor k-tuple for $\vec{v}_j$ the algorithm $\mathcal{A}_k$ considers it. So $\mathcal{T}_k[\vec{v}_j] \geq \mathcal{T}_k[\vec{v}_i] + w(\vec{v}_i) + \mathcal{A}_{k-1}(l_{k-1}(\vec{v}_i), r_{k-1}(\vec{v}_j))$, which is at least as good as OPT restricted to $L_k[\vec{v}_j]$. This is a contradiction and proves condition (C3).

For proving the time and space complexity let z_i be the number of separating i-tuples in an instance for $1 < i \leq k$. Each z_i is in $O(n^i)$. We again use induction on k. Algorithm $\mathcal{A}_1$ needs $O(n^2)$ time and $O(n)$ space, which match the bounds to be shown. So let $k > 1$. The table $\mathcal{T}_k$ has $O(z_k) \subseteq O(n^k)$ entries and each of the recursive computations of $\mathcal{A}_{k-1}$ need $O(n^{k-1})$ space by the induction hypothesis. Thus the overall space is dominated by $\mathcal{T}_k$ and the bound follows. Checking whether a separating k-tuple $\vec{v}_i \in S_k$ is a feasible predecessor for a particular $\vec{v}_j$ can easily be done in $O(k^2)$ time, which is dominated by the time to compute $\mathcal{A}_{k-1}(l_{k-1}(\vec{v}_i), r_{k-1}(\vec{v}_j))$. So for the running time we observe that each entry in $\mathcal{T}_k$ makes $O(z_k)$ calls to $\mathcal{A}_{k-1}$ and hence the overall running time is indeed $O(n^{2k} \cdot n^{(k-1)^2+(k-1)-1}) = O(n^{k^2+k-1})$. □

4.2 Extending the Algorithm for k-RestrictedMaxTotal to AM2

With some modifications and at the expense of another polynomial factor in the running time we can extend algorithm $\mathcal{A}_k$ of the previous section to the activity model AM2, which shows that k-RestrictedMaxTotal in AM2 can still be solved in polynomial time. In the following, we give a sketch of the modifications. The important difference between AM1 and AM2 is that presence intervals can be truncated at their right side if there is an active conflicting witness label causing the truncation. We need two modifications to model this behavior. First, we create for each *original* presence interval $I_i = [a_i, b_i]$ in Ψ at most n prefix intervals $I_i^j = [a_i, c_{ij}]$, where c_{ij} is the start of the first conflict between I_i and $I_j \in \Psi$. Each interval I_i^j inherits the conflicts of I_i that intersect I_i^j. We obtain a modified set of presence intervals $\Psi' = \Psi \cup \{I_i^j \mid I_i, I_j \in \Psi \text{ and } I_i, I_j \text{ in conflict}\}$ of size $O(n^2)$. We create mutual conflicts among all intervals that are prefixes of the same original interval. This will enforce that at most one of them is active. We still have to take care that a truncated interval I_i^j can only be active if I_j (or a prefix of I_j) is active at c_{ij} as a witness.

In order to achieve this we instantiate the algorithm $\mathcal{A}_{k'}$ for every $k' \leq k$ not only with its two boundary k-tuples $\tilde{\vec{v}}_0$ and $\tilde{\vec{v}}_{z+1}$ but also with a set W of at most k witness intervals that are k-compatible and must be made active at some stage of the algorithm. In a valid solution we have $W \subseteq L_{k'}[\vec{v}] \cup \vec{v}$ for the leftmost separating k'-tuple $\vec{v}$, since otherwise more than k' intervals are active in $K_{\vec{v}}$. However, the truncated intervals in $\vec{v}$ themselves define a family of $O(n^{k'})$ possible witness sets $W(\vec{v})$ to be respected to the right of $\vec{v}$. So when we compute the table entry for a separating k'-tuple $\vec{v}_j$ and consider a particular predecessor k'-tuple $\vec{v}_i$ we must in fact iterate over all possible witness sets $W(\vec{v}_i)$ as well. We need to make sure that $\vec{v}_j$ is $W(\vec{v}_i)$-*compatible*, i.e., $\vec{v}_j \cup W(\vec{v}_i)$ is k-compatible and $W(\vec{v}_i) \subseteq L_{k'}[\vec{v}_j] \cup \vec{v}_j$. For the recursive call to

$\mathcal{A}_{k'-1}(\tilde{\vec{v}}_i, \tilde{\vec{v}}_j)$ the initial witness set W' consists of $W(\vec{v}_i) \setminus \vec{v}_j$, i.e., those witness intervals of $W(\vec{v}_i)$ that are not part of $\vec{v}_j$.

The increase in running time is caused by dealing with $O(n^2)$ intervals in Ψ' and by the fact that instead of one call to $\mathcal{A}_{k-1}(\tilde{\vec{v}}_i, \tilde{\vec{v}}_j)$ in the computation of table $\mathcal{T}_k$ we make $O(n^k)$ calls, one for each possible witness set of $\tilde{\vec{v}}_i$. By an inductive argument one can show that the running time is in $O(n^{3k^2+2k})$.

Theorem 4 *k-RestrictedMaxTotal in AM2 can be solved in $O(n^{3k^2+2k})$ time.*

4.3 Complexity of k-RestrictedMaxTotal in AM3

In this section we discuss the complexity of k-RestrictedMaxTotal with respect to AM3. For $k = 1$ the problem is equivalent to finding a maximum weighted independent set on interval graphs, which can be done in linear time [13].

We show that the corresponding decision problem, which is defined as follows, is NP-complete for $k \geq 2$.

Problem 3 (k-RestrictedMaxTotalDecision)

Given: *Instance $\mathcal{I} = (L, \Psi, C)$, $k \in \mathbb{N}$ and $W \in \mathbb{R}^+$.*
Question: *Is there an activity set Φ for $\mathcal{I}$ such that $\sum_{[a,b]_\ell \in \Phi} w([a,b]_\ell) \geq W$ and at most k labels are active at the same time?*

We first show that the decision problem is NP-hard for $k \geq 2$, which implies that the corresponding optimization problem is NP-hard as well. We reduce from the NP-complete problem 3-Partition. For the convenience of the reader, we repeat the definition given by Garey and Johnson [8].

Problem 4 (3-Partition)

Given: *Set X of 3m elements, a* bound $B \in \mathbb{Z}^+$, *and a* size $s(x) \in \mathbb{Z}^+$ *for each element $x \in X$ such that $\frac{B}{4} < s(x) < \frac{B}{2}$ and $\sum_{x \in X} s(x) = mB$.*
Question: *Can X be partitioned into m disjoint sets $X_1, X_2, \ldots, X_m$ such that $\sum_{x \in X_i} s(x) = B$ for all i with $1 \leq i \leq m$?*

Note that each X_i must contain exactly three elements. We first present a construction that transforms a given instance $\mathcal{I}$ of 3-Partition into an instance $\mathcal{I}'$ of k-RestrictedMaxTotal. As the construction is rather technical, we give a short sketch beforehand discussing the main ideas. Afterwards, we present the formal details. Finally, we show that $\mathcal{I}$ is a *yes*-instance of 3-Partition if and only if $\mathcal{I}'$ is a *yes*-instance of k-RestrictedMaxTotal with respect to AM3.

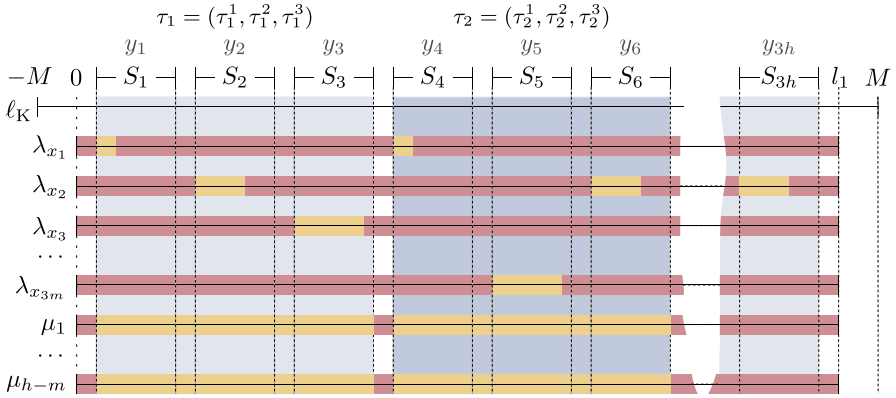


Fig. 11 Illustration of the construction used for the reduction from 3-Partition to k-RestrictedMaxTotal. Times for which a label $\ell \in L_\lambda \cup L_\mu$ is in conflict with ℓ_K are marked red. For each label $\lambda_x \in L_\lambda$ the times when it can possibly be active are marked orange (Color figure online)

Sketch. The construction is schematically illustrated in Fig. 11 and an example for a concrete instance of 3-Partition is given in Fig. 12. We observe that 3-Partition can be slightly reformulated as follows. Let $\tau \subseteq X$ be a triplet such that the three elements in τ are mutually distinct and $\sum_{x \in \tau} s(x) = B$ and let $T = \{\tau_1, \ldots, \tau_h\}$ denote the set of all these triplets. The problem 3-Partition is then equivalent to selecting a subset $T' \subseteq T$ with size m such that each element $x \in X$ is contained in exactly one triplet $\tau \in T'$. We model this selection problem using three groups of labels. The first group consists of a single label ℓ_K. We choose its presence interval appropriately long in order to enforce that it is active for its complete presence interval. Further, for all other labels we will choose their presence intervals such that they are contained in ℓ_K's presence interval. By introducing conflicts between ℓ_K and these labels, this provides us with the possibility of defining exact ranges in which a label can be active during its presence interval. The second group L_λ consists of $3m$ labels each representing one element of X. In our construction we will enforce that the label $\lambda_x \in L_\lambda$ of each element $x \in X$ is active for exactly $s(x)$ time. Further, we ensure that its activity interval can be uniquely assigned to one of the triplets in T that contains x. The third group L_μ consists of $h - m$ many labels representing the triplets in T that are not selected in a solution. More precisely, we ensure for each label in L_μ that its activity interval is uniquely assigned to one of the triplets in T such that we interpret this label as not selected for the solution.

Further, we ensure that if a label $\lambda_x \in L_\lambda$ is active for triplet τ, then no label μ of L_μ is active for τ. Choosing W appropriately, we enforce that all defined labels must have an activity interval. Depending on the chosen activity intervals we then derive a solution for 3-Partition.

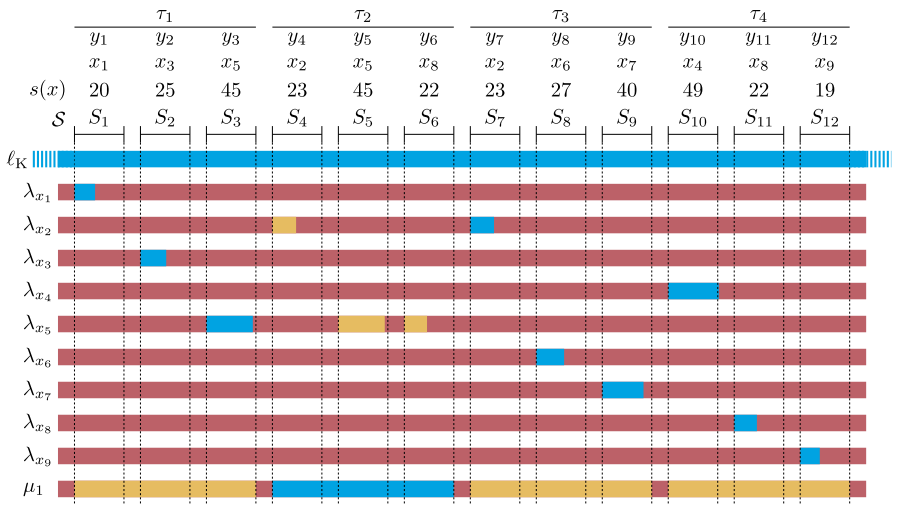


Fig. 12 Example for the reduction from 3-Partition to k-RestrictedMaxTotal. The input are the nine elements $x_1, \ldots, x_9$ with $s(x_1) = 20$, $s(x_2) = 23$, $s(x_3) = 25$, $s(x_4) = 49$, $s(x_5) = 45$, $s(x_6) = 27$, $s(x_7) = 40$, $s(x_8) = 22$ and $s(x_9) = 19$. Times for which a label $\ell \in L_\lambda \cup L_\mu$ is in conflict with ℓ_K are marked red. For each label its activity interval is blue. Further, for each label $\lambda_x \in L_\lambda$ the other times when it can possibly be active are marked orange. The solution is $\tau_1 = \{x_1, x_3, x_5\}$, $\tau_3 = \{x_2, x_6, x_7\}$ and $\tau_4 = \{x_4, x_8, x_9\}$ (Color figure online)

Details. Let $s_{\max} = \max_{x \in X} s(x)$. We first construct every triplet $\tau \subseteq X$ such that the three elements in τ are mutually distinct and $\sum_{x \in \tau} s(x) = B$. Let $T = \{\tau_1, \ldots, \tau_h\}$ denote the set of those triplets. For each triplet $\tau_i \in T$ we fix an arbitrary order of its elements, and denote its j-th element by τ_i^j (with $j \in \{1, 2, 3\}$). We define

$$y_1 = \tau_1^1, \quad y_2 = \tau_1^2, \quad y_3 = \tau_1^3, \quad y_4 = \tau_2^1, \quad y_5 = \tau_2^2, \ldots, y_{3h} = \tau_h^3.$$

In case that $h < m$, the given instance is a *no*-instance. Hence, in the following we assume that $m \geq h$. For each element $x \in X$ we create one label λ_x with presence interval

$$I = [0, 3h \cdot (s_{\max} + 1) + 1].$$

Let L_λ denote the set of those labels, and let l_1 denote the common length of those intervals. The interval I can be partitioned into a set $\mathcal{S}$ of $3h$ intervals of length $s_{\max}$ and a set $\mathcal{U}$ of $3h + 1$ intervals of unit length such that the intervals in $\mathcal{S}$ and $\mathcal{U}$ alternate, starting with an interval of $\mathcal{U}$; see Fig. 11. We call an interval in $\mathcal{S}$ a *slot*. We denote the slots by $S_1 = [a_1, b_1], \ldots, S_{3h} = [a_{3h}, b_{3h}]$ in increasing order, i.e., $a_i < a_j$ for $i < j$. We say that the slots S_{3i-2}, S_{3i-1} and S_{3i} *belong to* the triplet τ_i ($1 \leq i \leq h$).

Thus, for each x_i we have a slot S_i with $1 \leq i \leq 3h$. In our construction we will enforce that a label $\lambda_x \in L_\lambda$ can only be active during a slot S_i ($1 \leq i \leq 3h$) if $y_i = x$. More precisely, we define

$$S_{x,i} = \begin{cases} [a_i, a_i + s(x)], & \text{if } y_i = x \\ \emptyset, & \text{otherwise,} \end{cases}$$

Thus, $S_{x,i} \subseteq S_i$. We will enforce that λ_x can only be active during $S_{x,i}$. To that end we introduce a single label ℓ_{K} with presence interval

$$K = [-M, l_1 + M],$$

where we choose $M > 3l_1$. We denote the length of K by l_2 and note that $l_2 \geq 2M$.

For each $x \in X$ we introduce conflicts between λ_x and ℓ_{K} such that λ_x and ℓ_{K} are *not* in conflict at time t if and only if there is an i with $1 \leq i \leq 3h$ and $t \in S_{x,i}$. Put differently, λ_x and ℓ_{K} are in conflict during the time expressed by

$$I \setminus \bigcup_{i=1}^{3h} S_{x,i}.$$

Assume that ℓ_{K} is active during the complete interval $[0, l_1]$, then any label λ_x with $x \in X$ can only be active during $\bigcup_{i=1}^{3h} S_{x,i}$. Since the slots are disjoint, $S_{x,i} \subseteq S_i$, and each label can only be active once per presence interval, each label λ_x can only be active during at most one slot S_i, but not outside of a slot. Further, the element x must be contained in the corresponding triplet τ_j with $j = \lceil \frac{i}{3} \rceil$.

Moreover, we introduce $h - m$ labels with presence interval I. We denote the set of these labels by L_μ. We define that any label $\mu \in L_\mu$ and ℓ_{K} are in conflict during the time expressed by

$$I \setminus \bigcup_{j=1}^{h} [a_{3j-2}, b_{3j}].$$

Thus, μ is not in conflict with ℓ_{K} during any interval $T_i = [a_{3i-2}, b_{3i}]$ (with $1 \leq i \leq h$), which spans the slots S_{3i-2}, S_{3i-1}, S_{3i} of the triplet τ_i. We note that the length of T_i is $l_3 = 3s_{\max} + 2$. We further define that the labels in $L_\lambda \cup L_\mu$ are pairwise in conflict during the complete interval I, which implies that at most two labels can be active at any time, namely ℓ_{K} and at most one label of $L_\lambda \cup L_\mu$. Finally, for any constructed label ℓ we set $w_\ell = 1$ and define

$$W = l_2 + \sum_{x \in X} s(x) + (h - m) \cdot l_3$$

Hence, we have $w([a, b]_\ell) = (b - a)$ for any presence/activity interval of ℓ.

Altogether, we obtain the instance $\mathcal{I}' = (L = L_\lambda \cup \{\ell_{\mathrm{K}}\} \cup L_\mu, \Psi, C, W)$, which can obviously be constructed in polynomial time with respect to the given 3-Partition instance $\mathcal{I}$.

Lemma 3 *Let $\mathcal{I}$ be an instance of* 3-Partition *and let $\mathcal{I}'$ be an instance of k-RestrictedMaxTotalDecision ($k \geq 2$) with respect to AM3 constructed from $\mathcal{I}$ as described. The instance $\mathcal{I}$ is a yes-instance if and only if $\mathcal{I}'$ is a yes-instance.*

Proof First assume that $\mathcal{I}$ is a *yes*-instance of 3-Partition. Let $T' \subseteq T$ be the set of the m triplets such that each element $x \in X$ is contained in exactly one triplet of T' and the sizes of the elements in each triplet of T' sum up to B. We now construct an activity set Φ for $\mathcal{I}'$ with weight $W_\Phi \geq W$. To that end we first set ℓ_K active for the interval K. Thus, $W_\Phi \geq l_2$.

Further, for each triplet $\tau \in T'$ and each element $x \in \tau$ we add the slot $S_{x,i}$ to Φ, where $1 \leq i \leq 3h$, $x = x_i$ and $S_{x,i}$ belongs to τ. By construction λ_x is not in conflict with ℓ_K during $S_{x,i}$, but directly before and after that interval. Since K is completely contained in Φ, the label ℓ_K is also active directly before and after that interval, which altogether satisfies AM3. Further, only two labels are active at the same time. This follows directly from the fact that the slots are disjoint and we set for each slot at most two labels active, namely λ_x and ℓ_K. Since each added interval $S_{x,i}$ has length $s(x)$ and each x belongs to exactly one triplet $\tau \in T'$, the activity set Φ now has weight $W_\Phi = l_2 + \sum_{x \in X} s(x)$.

There are $h - m$ triplets $\tau_{i_1}, \ldots, \tau_{i_{h-m}}$ in $T \setminus T'$. For each such triplet τ_{i_j} we know by construction that no label $\lambda_x \in L_\lambda$ is active during T_{i_j}. Hence, we set the label μ_j active during the interval T_{i_j}. Since μ_j is in conflict with ℓ_K directly before and after T_{i_j}, the model AM3 is satisfied. Further, since $T_{i_1}, \ldots, T_{i_{h-m}}$ are pairwise disjoint, at most two labels are active at the same time. In total we obtain a valid activity set Φ with weight

$$W_\Phi = l_2 + \sum_{x \in X} s(x) + (h - m) \cdot l_3 = W.$$

Hence, Φ is a solution for k-RestrictedMaxTotalDecision for any $k \geq 2$.

Now assume that $\mathcal{I}'$ is a *yes*-instance of k-RestrictedMaxTotal ($k \geq 2$) with respect to AM3, i.e., we are given an activity set Φ for $\mathcal{I}'$ with $W_\Phi \geq W$. We show how to construct a valid solution for $\mathcal{I}$.

Recall that since the labels in $L_\lambda \cup L_\mu$ are pairwise in conflict for the complete interval I, for any $k \geq 2$ at most two labels can be active at the same time, namely ℓ_K and one label of $L_\lambda \cup L_\mu$.

We now argue that ℓ_K is active for the complete interval $[0, l_1]$. If this was not the case, then the prefix $[-M, 0]$ or the suffix $[l_1, l_1 + M]$ of K cannot belong to the activity of ℓ_K. Further, for the interval $[0, l_1]$ we obtain less than $\frac{2}{3}M$ weight in total, because at most two labels can be active at the same time and $l_1 < \frac{M}{3}$. Thus, we obtain $W_\Phi < M + \frac{2}{3}M < 2M$. On the other hand, because of $l_2 \geq 2M$ and $W_\Phi \geq W$, we have $W_\Phi \geq 2M$, which is a contradiction. On that account, the label ℓ_K is active during the complete interval $[0, l_1]$. It is even active for the complete interval K to sustain the requirements of AM3. As reasoned previously, the activity interval of any label $\ell \in L_\lambda$ must be thus contained in one of the slots $S_1, \ldots, S_{3h}$; otherwise there would be an unresolved conflict between ℓ and ℓ_K. Similarly, the activity interval of any label $\ell \in L_\mu$ must be contained in one of the intervals $T_1, \ldots, T_h$.

By the construction of the conflicts, a label $\lambda_x \in L_\lambda$ can be active for at most $s(x)$ time and a label $\mu_i \in L_\mu$ can be active for at most l_3 time. Due to $W_\Phi \geq W$, each label $\lambda_x \in L_x$ is therefore active for exactly $s(x)$ time and, consequently, there is a slot $S_{x,i}$ with $1 \leq i \leq 3h$ that is the activity interval of λ_x. Similarly each label $\mu \in L_\mu$

is active for exactly l_3 time and, consequently, there is an interval T_i with $1 \leq i \leq h$ that is the activity interval of μ.

More precisely, there are m triplets $\tau_{i_1}, \ldots, \tau_{i_m}$ such that the activity intervals of the labels in L_λ are contained in $\bigcup_{j=1}^{m} T_{i_j}$. To see that, consider any label $\mu \in L_\mu$. By the previous reasoning there is a triplet τ_i with $1 \leq i \leq h$ such that the activity interval T_i of μ contains $S_{3i-2}, S_{3i-1}, S_{3i}$. This implies that no label $\lambda_x \in L_\lambda$ with $x \in \tau_i$ can be active during T_i. Since no two labels $\mu, \mu' \in L_\mu$ can be active during the same interval T_i and $|L_\mu| = h - m$, the m remaining intervals $T_{i_1}, \ldots, T_{i_m}$ contain the activity intervals of the labels in L_λ.

Those triplets $\tau_{i_1}, \ldots, \tau_{i_m}$ form the desired solution for $\mathcal{I}$: Since each label $\lambda_x \in L_\lambda$ is active for exactly one interval, each element $x \in X$ is contained in exactly one triplet τ_{i_j} with $1 \leq j \leq m$. Further, we have $\sum_{x \in \tau_{i_j}} s(x) = B$, which concludes the proof. □

The lemma directly implies that the problem k-RestrictedMaxTotal is NP-hard for $k \geq 2$ and AM3. Moreover, it is easy to see that the decision problem is in NP. For each label $\ell \in L$ we guess intervals that are contained in the presence intervals of ℓ and whose beginnings and ends coincide with the beginnings and ends of presence and conflict intervals in $\Psi \cup C$. We further check whether the intervals are valid activity intervals with respect to AM3, whether their weights sum up at least to W, and whether at most k labels are active at the same time. Obviously this takes time polynomial in the size of the input. We summarize the results of this section in the following theorem.

Theorem 5 *For $k = 1$ the problem k-RestrictedMaxTotal can be solved in linear time for AM3, while k-RestrictedMaxTotalDecision is NP-complete for $k \geq 2$.*

We observe that from a computational point of view this hardness result justifies the use of AM2. While AM2 is more flexible then AM1, it is less difficult to solve than AM3. It is an open question, whether k-RestrictedMaxTotal in AM3 remains NP-hard for $k \geq 2$ if the concrete geometric setting is also taken into account. For example, in case that the viewport follows a predefined trajectory, does this help to solve the problem in polynomial time?

4.4 Approximation of k-RestrictedMaxTotal

Since the running times of our algorithms for k-RestrictedMaxTotal are, even for small k, prohibitively expensive in practice, we propose an approximation algorithm for k-RestrictedMaxTotal based on GreedyMaxTotal for all activity models.

Our algorithm GreedyRestrictedMaxTotal is a simple extension of GreedyMaxTotal. Recall that GreedyMaxTotal greedily removes the longest interval I from Ψ and adds it to the set Φ that contains the active intervals of the solution. Then, it updates all intervals contained in Ψ that are in conflict with I. This process is repeated until Ψ is empty. For approximating k-RestrictedMaxTotal we need to

ensure that there is no point in time t that is contained in more than k intervals in Φ. We call intervals which we cannot add to Φ without violating this property *invalid*.

Our modification of GreedyMaxTotal is as follows. After adding an interval I to Φ and handling conflicts as before, we remove intervals from Ψ that became invalid. We say that we *ensure that I is valid*. Note that we cannot shorten those intervals because then we could not ensure that adding an interval from Ψ to Φ is valid according to our model.

In order to prove approximation ratios we first introduce the following lemma that describes the structure of a solution of k-RestrictedMaxTotal.

Lemma 4 ([12]) *Let S be a set of intervals such that there is no number that is contained in more than k intervals from S. Then, there is a partition of S into k sets $M_1, \ldots, M_k$, such that no two intersecting intervals are in the same set M_i.*

For the proof we refer to [12]. With this lemma we now can prove the following theorem that makes a statement about the approximation ratio of GreedyRestrictedMaxTotal

Theorem 6 *Assuming that all labels are unit squares and $w([a,b]) = b - a$, GreedyRestrictedMaxTotal is a $1/\min\{3+3k, 27\}$, $1/\min\{3+2k, 19\}$, $1/\min\{3+k, 11\}$-approximation for AM1–AM3, respectively, and runs in $O(n^2)$ time.*

Proof We begin by proving its correctness and then we show its time complexity.

Consider the step in which we add an interval $I = [a,b]$ to Φ, and let $J = [a - w(I), b + w(I)]$. Let $\mathcal{L}$ be a fixed, but arbitrary optimal solution. If $I \in \mathcal{L}$, there is no lost weight compared to the optimal solution when choosing I.

Thus, assume that $I \notin \mathcal{L}$. Let $C(I) \subseteq \mathcal{L}$ be the set of intervals that are in conflict with I. Identically to the proof of Theorem 2 we can argue that $w(C(I)) = \sum_{I \in C} w(I) \leq (4 - X) \cdot 8 \cdot w(I)$ considering activity model AMX with $X \in \{1, 2, 3\}$.

We now show that at most $3w(I)$ weight of the optimal solution is lost when ensuring that I is valid.

By Lemma 4 we can partition $\mathcal{L}$ into k sets $M_1, \ldots, M_k$ such that no two intersecting intervals are in the same set M_i. If I is in $\mathcal{L}$, then we do not lose any weight compared to the optimal solution. Hence, assume that I is not in $\mathcal{L}$. Take any $M_i, 1 \leq i \leq k$, remove all intervals of $\mathcal{L} \setminus \Phi$ that intersect I. We denote the set of removed intervals by R. In the following, we bound the cost of removing the intervals in R. If there are intervals in R that are longer than I, then we have already accounted for them in previous steps. This relies on the fact that we consider the intervals sorted by their length in non-ascending order, and, hence if those longer intervals are not in Φ, we must have removed them in an earlier step. Thus, we only need to bound the length of intervals in R which have length at most $w(I)$. Those intervals must lie in J, and due to the definition of M_i they must be disjoint. Hence, the cost for ensuring that I is valid is bounded by $3w(I)$.

Altogether choosing I causes that at most $3w(I) + (4 - X) \cdot 8 \cdot w(I)$ weight is lost compared to the optimal solution considering activity model AMX with $X \in \{1, 2, 3\}$. Finally, this yields an approximation factor of 1/27, 1/19, 1/11 for AM1, AM2, and AM3, respectively.

For $k < 8$ we can improve $w(C(I)) \leq (4 - X) \cdot 8 \cdot w(I)$ to $w(C(I)) \leq (4 - X) \cdot k \cdot w(I)$ considering activity model AMX with $X \in \{1, 2, 3\}$ because we know that I cannot be in conflict with more than k intervals of the optimal solution. Thus, we can bound the loss of choosing I by $3w(I) + (4 - X) \cdot k \cdot w(I)$. In total this yields the claimed approximation ratios for the three activity models.

Finally, we argue the correctness of the claimed running time of $O(n^2)$. Since the worst-case running time of GreedyMaxTotal is $O(n^2)$ we only need to argue that we can delete those intervals from Φ, which are not valid anymore, in $O(n)$ time per step. To do this we simply sort the intervals in Ψ in non-decreasing order by their left-endpoint. We also maintain Φ in the same way. Then, we can check for non-valid intervals with a simple linear sweep over Ψ and Φ. Hence, each iteration of the algorithm requires $O(n)$ time, which yields a total running time of $O(n^2)$. □

5 Conclusions

In this paper, we introduced a temporal model for dynamic map labeling that satisfies the consistency criteria demanded by Been et al. [1], even in a stronger sense, where each activity change of a label must be explainable to the user by some witness label. Our model transforms the geometric information specified by the motion of the camera as well as the labels into the two combinatorial problems General-MaxTotal and k-RestrictedMaxTotal that are expressed in terms of presence and conflict intervals. Thus, our algorithms apply to any dynamic labeling problem that can be transformed into such an interval-based problem.

We showed that GeneralMaxTotal is NP-complete and W[1]-hard and presented constant-factor approximation algorithms for our three different activity models. The problem k-RestrictedMaxTotal, where at most k labels can be visible at any time, can be solved in polynomial time $O(n^{f(k)})$ in activity models AM1 and AM2 for any fixed k, where f is a polynomial function. Hence, k-RestrictedMaxTotal lies in the complexity class XP for AM1 and AM2. Due to the W[1]-hardness of General-MaxTotal we cannot expect to find better results for the running times, apart from improving upon the function f. Further, we proved NP-hardness for k-Restricted-MaxTotal in AM3 for $k \geq 2$. We also presented an $O(n^2)$-time approximation algorithm for k-RestrictedMaxTotal in all three activity models.

The analysis of the approximation algorithms for both k-RestrictedMaxTotal and GeneralMaxTotal relies on the assumption that labels are unit squares. We can soften this assumption by bounding the ratio between the largest and smallest label size. We observe, that the core of our packing arguments remains valid obtaining constant-factor approximations for this setting as well. However, the more the label size ratio increases, the more the approximation ratios decline.

In this paper we have restricted our attention to a combinatorial view on temporal map labeling, i.e., we have assumed that we are given all presence and conflict intervals in advance. This allowed us to completely abstract from the actual geometric setting obtaining generic algorithms for different problem variants such as map labeling for car navigation and labeling moving entities. It is future work to incorporate the specific geometric settings into our results. For example, does the geometric information based on viewport, its motion and the geometry of the labels imply a useful structure on the induced label conflict graph? Such insights promise further improvements on our results.

Acknowledgements Open Access funding provided by Projekt DEAL. We thank Lukas Barth and Darren Strash for many interesting discussions.

References

1. Been, K., Daiches, E., Yap, C.: Dynamic map labeling. IEEE Trans. Vis. Comput. Graph. **12**(5), 773–780 (2006)
2. Buchin, K., Gerrits, D.H.P.: Dynamic point labeling is strongly PSPACE-complete. Int. J. Comput. Geom. Appl. **24**(4), 373–395 (2014)
3. Barth, L., Gemsa, A., Niedermann, B., Nöllenburg, M.: Temporal map labeling: a new unified framework with experiments. In: Advances in Geographic Information Systems (ACM-GIS'16). ACM Press (2016)
4. Been, K., Nöllenburg, M., Poon, S.-H., Wolff, A.: Optimizing active ranges for consistent dynamic map labeling. Comput. Geom. Theory Appl. **43**(3), 312–328 (2010)
5. Carlisle, M.C., Lloyd, E.L.: On the k-coloring of intervals. Discrete Appl. Math. **59**(3), 225–235 (1995)
6. de Berg, M., Gerrits, D.H.P.: Labeling moving points with a trade-off between label speed and label overlap. In: Bodlaender, H.L., Italiano, G.F. (eds.) Algorithms (ESA'13). Lecture Notes in Computer Science, vol. 8125, pp. 373–384. Springer, Berlin (2013)
7. Fowler, R.J., Paterson, M.S., Tanimoto, S.L.: Optimal packing and covering in the plane are NP-complete. Inf. Process. Lett. **12**(3), 133–137 (1981)
8. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1979)
9. Gemsa, A., Nöllenburg, M., Rutter, I.: Sliding labels for dynamic point labeling. In: Canadian Conference on Computational Geometry (CCCG'11), pp. 205–210 (2011)
10. Gemsa, A., Nöllenburg, M., Rutter, I.: Consistent labeling of rotating maps. J. Comput. Geom. **7**(1), 308–331 (2016)
11. Gemsa, A., Nöllenburg, M., Rutter, I.: Evaluation of labeling strategies for rotating maps. J. Exp. Algorithmics **21**(1), 1–21 (2016)
12. Golumbic, M.C.: Interval graphs, chapter 8. In: Golumbic, M.C. (ed.) Algorithmic Graph Theory and Perfect Graphs, pp. 171–202. Academic Press, Cambridge (1980)
13. Hsiao, J.Y., Tang, C.Y., Chang, R.S.: An efficient algorithm for finding a maximum weight 2-independent set on interval graphs. Inf. Process. Lett. **43**(5), 229–235 (1992)

14. Liao, C.-S., Liang, C.-W., Poon, S.-H.: Approximation algorithms on consistent dynamic map labeling. In: Chen, J., Hopcroft, J.E., Wang, J. (eds.) Frontiers in Algorithmics (FAW'14). Lecture Notes in Computer Science, vol. 8497, pp. 170–181. Springer, Cham (2014)
15. Luboschik, M., Schumann, H., Cords, H.: Particle-based labeling: fast point-feature labeling without obscuring other visual features. IEEE Trans. Vis. Comput. Graph. **14**(6), 1237–1244 (2008)
16. Marx, D.: Efficient approximation schemes for geometric problems? In: Brodal, G.S., Leonardi, S. (eds.) Algorithms (ESA'05). Lecture Notes in Computer Science, vol. 3669, pp. 448–459. Springer, Berlin (2005)
17. Maass, S., Döllner, J.: Efficient view management for dynamic annotation placement in virtual landscapes. In: Butz, A., Fisher, B., Krüger, A., Olivier, P. (eds.) Smart Graphics (SG'06). Lecture Notes in Computer Science, vol. 4073, pp. 1–12. Springer, Berlin (2006)
18. Maass, S., Döllner, J.: Embedded labels for line features in interactive 3d virtual environments. In: Computer Graphics, Virtual Reality, Visualisation and Interaction (AFRIGRAPH'07), pp. 53–59. ACM Press (2007)
19. Miller, G.A.: The magical number seven, plus or minus two: some limits on our capacity for processing information. Psychol. Rev. **63**(2), 81–97 (1956)
20. Mote, K.: Fast point-feature label placement for dynamic visualizations. Inf. Vis. **6**(4), 249–260 (2007)
21. Petzold, I., Gröger, G., Plümer, L.: Fast screen map labeling—data structures and algorithms. In: International Cartographic Conference (ICC'03), pp. 288–298 (2003)
22. Sester, M., Brenner, C.: Continuous generalization for visualization on small mobile devices. In: Spatial Data Handling (SDH'04), pp. 355–368. Springer, Berlin (2004)
23. Schwartges, N., Morgan, B., Haunert, J.-H., Wolff, A.: Labeling streets along a route in interactive 3D maps using billboards. In: Bacao, F., Santos, M., Painho, M. (eds.) AGILE 2015. Lecture Notes in Geoinformation and Cartography, pp. 269–287. Springer, Cham (2015)
24. Schwartges, N., Wolff, A., Haunert, J.-H.: Labeling streets in interactive maps using embedded labels. In: Advances in Geographic Information Systems (ACM-GIS'14), pp. 517–520. ACM Press (2014)
25. Vaaraniemi, M., Treib, M., Westermann, R.: Temporally coherent real-time labeling of dynamic scenes. In: Computing Geospatial Research Applications (COM.Geo'12), pp. 17:1–17:10. ACM Press (2012)
26. Yokosuka, Y., Imai, K.: Polynomial time algorithms for label size maximization on rotating maps. In: Canadian Conference on Computational Geometry (CCCG'13), pp. 187–192 (2013)
27. Zhang, X., Poon, S.-H., Li, M., Lee, V.: On maxmin active range problem for weighted consistent dynamic map labeling. In: GEOProcessing 2015, pp. 32–37. IARIA (2015)