Sliding labels for dynamic point labeling

Andreas Gemsa* Martin Nöllenburg* Ignaz Rutter*

Abstract

We study a dynamic labeling problem for points on a line that is closely related to labeling of zoomable maps. Typically, labels have a constant size on screen, which means that, as the scale of the map decreases during zooming, the labels grow relatively to the set of points, and conflicts may occur due to overlapping labels. Our algorithmic problem is a combined dynamic selection and placement problem in a sliding-label model: (i) select for each label ℓ a contiguous *active range* of map scales at which ℓ is displayed, and (ii) place each label at an appropriate position relative to its anchor point by sliding it along the point. The *active range optimization* (ARO) problem is to select active ranges and slider positions so that no two labels intersect at any scale and the sum of the lengths of active ranges is maximized. We present a dynamic programming algorithm to solve the discrete k-position ARO problem optimally and an FPTAS for the continuous sliding ARO problem.

1 Introduction

With the increasing practical importance of dynamic maps that allow continuous operations like zooming, panning, or rotations, dynamic labeling of map features becomes a critical aspect of the visual quality of a map. Examples of dynamic maps range from maps on small-screen mobile devices to professional desktop GIS applications. The map dynamics add new dimensions to label placement, which result in challenging geometric optimization problems that are quite different from static labeling problems. Changes in dynamic maps due to continuous map movements need to be smoothly animated in order to preserve a coherent context and minimize the user's cognitive load for re-orientation [12]. This requirement is also known as "frame coherency" [2] or "temporal continuity" [5]. Hence we cannot simply solve the arising labeling problems independently for each intermediate map view during the animation; rather we need to solve the labeling problem globally such that the animations of all possible trajectories using the given set of navigation operations satisfy the quality constraints.

In order to avoid distraction and irritation of the user a dynamic map should—according to Been et al. [3]—adhere to the following quality constraints or *consistency desiderata* for dynamic map labeling: During monotone map movement labels should neither "jump" (non-continuously change position or size) nor "pop" (vanish when zooming in or appear when zooming out); moreover, the labeling should be a function of the selected map viewport and not depend on the navigation history. In this paper we are only interested in dynamic labelings that are consistent in that sense. Of course each static map view in a dynamic map also needs to satisfy the quality standards for static maps [8], i.e., all labels—usually modeled as rectangles—are pairwise disjoint, each label is close to its anchor point, and, globally over all possible map views, the number of visible labels is maximum.

Been et al. [4] presented a first extensive study of algorithms for dynamic map labeling in several different models for one- and two-dimensional input point sets. However, they focused on the *dynamic label selection* problem, i.e., which set of labels to select at which scale, and assumed that for each label a single, fixed position relative to the anchor point is given in the input. They left *dynamic label placement* as an open problem, i.e., the problem where to place each label relative to its anchor point. One model for label placement is the *k-position* or *fixed-position* model, where each label can be placed at a position from a set of k (usually 4 or 8) possible positions [1, 6, 17]. Another more general model is the *slider* model, where the finite-position assumption is dropped and each label can take any position such that the anchor point coincides with a point on the label boundary [15, 16]. In this paper we present labeling algorithms in a dynamic scenario that allows continuous zooming for the visualization of a one-dimensional input point set. To the best of our knowledge our algorithms are the first to combine the dynamic label selection problem with label placement in both the fixed-position and the slider model, thus answering (partially) an open question of Been et al. [4].

Related Work. Most previous algorithmic research on automated label placement deals with *static* fixed-position or slider models for point, line, or area features. The problem of maximizing the number of selected labels is NP-hard even for the simplest labeling models, whereas there are efficient algorithms for the decision problem that asks whether all points can be labeled in

*Department of Computer Science, Karlsruhe Institute of Technology (KIT), Germany, {gemsa, noellenburg, rutter}@kit.edu

some of the simpler models (see, e.g., the discussion by Klau and Mutzel [9] or the comprehensive map labeling bibliography [19]). Approximation results [1, 16], heuristics [18], and exact approaches [9] are known for many variants of the static label number maximization problem.

More recently, *dynamic* map labeling has emerged as a new research topic that gives rise to many unsolved algorithmic problems. Petzold et al. [13] used a preprocessing step to generate a reactive conflict graph that represents possible label overlaps for maps of all scales. For any fixed scale and map region, their method computes a conflict-free labeling in the slider model using heuristics. Poon and Shin [14] described algorithms for labeling one- and two-dimensional point sets that precompute a hierarchical data structure storing solutions for a number of different scales; this allows them to answer adaptive zooming queries efficiently. Mote [10] presented another fast heuristic method for dynamic conflict resolution in label placement that does not require preprocessing and assumes a 4-position model. The consistency desiderata of Been et al. [3] for dynamic labeling, however, are not satisfied by any of these three methods. Been et al. [4] showed NP-hardness of the label number maximization problem in the consistent labeling model and presented several approximation algorithms for labeling two-dimensional point sets and an exact algorithm for one-dimensional point sets. They focused on dynamic label selection, i.e., assumed a 1-position model for label placement. Nöllenburg et al. [11] recently studied a dynamic version of the alternative boundary labeling model allowing continuous zooming and panning, where labels are placed at the sides of the map and connected to their points by leaders. Algorithms and complexity results for dynamic label selection in fixed-scale rotating maps that satisfy similar consistency desiderata were presented by Gemsa et al. [7].

Contribution. In this paper we present algorithms for labeling a set of points on a line with labels of arbitrary, non-uniform length in a dynamic scenario that supports continuous zooming of the points' visualization. Unlike previous efforts [4] we consider label placement in a k-position and slider model: we must select both an interval of scales at which each label is selected (dynamic selection problem) and an admissible label position for each label relative to its anchor point (dynamic placement problem). We require that the label position remains the same for all scales. In Section 2 we introduce a model for dynamic point labeling with sliding labels in the framework of Been et al. [3]. Section 3 presents a dynamic programming algorithm for dynamically labeling points in the k-position model. Our main contribution is the fully polynomial-time approximation scheme (FPTAS) described in Section 4 for the more general sliding model. We conclude in Section 5 with several remaining open questions that arise from our results.

2 Preliminaries

In this section we describe our model for dynamic labeling in the general framework of Been et al. [3, 4].

Model. Let $P = \{p_1, \ldots, p_n\}$ be a set of points on the x-axis (also called the *base line*) together with a set $\mathcal{L} = \{\ell_1, \ldots, \ell_n\}$ of labels. The point p_i is called the *anchor point* of the label ℓ_i. Each label ℓ_i is a rectangle of (target) width w_i modeling the bounding box of the text describing the point p_i. Since we focus on labeling a one-dimensional point set, we can think of each label ℓ_i as actually being a line segment of width w_i. During zooming of the points' visualization we wish to keep the label size constant on screen, which means that if we scale the map by a factor of $1/s$ we need to increase the label size by a factor of s in order to maintain its width on screen constant. This is the *label size invariance property* of Been et al. [3]. So the width of ℓ_i on the base line required for a map of scale $1/s$ is given by the linear function $w_i(s) = w_i s$.

The label proximity constraint in map labeling says that each label must be close to its anchor point [8], i.e., we require for each label that the anchor point coincides with a point of the label. We consider sliding labels and define the shift position $t_i \in [0, 1]$ of a label ℓ_i as the fraction of ℓ_i that is to the right of p_i. For $t_i = 0$ the label is in its leftmost position, and for $t_i = 1$ it is in its rightmost position. In the fixed-position model only a finite subset of positions from $[0, 1]$ is allowed. In this paper we consider *invariant point placements* [3], i.e., once a shift position t is selected for a label ℓ, ℓ maintains that position relative to its anchor point. This immediately prevents the labels from jumping. Figure 1 shows a set of five points with labels zoomed to four different scales. Note that as the scale decreases, the points move closer together and some labels must be removed to avoid conflicting labels.

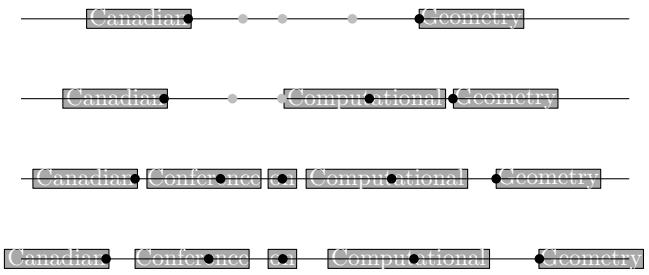

Figure 1: Five (partially) labeled points on a line zoomed from smaller (top) to larger scales (bottom).

Following Been et al. [3, 4] we define an extended two-dimensional coordinate system defined by the x-axis,

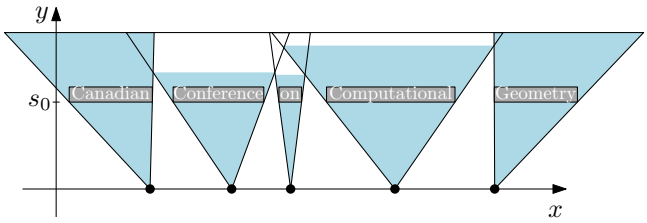

Figure 2: Triangular truncated extrusions (shaded blue) induced by the example of Figure 1.

which models the positions of the points P, and the y-axis, which models the inverse s of the scale $1/s$. We denote s as the *scale factor* that is used to enlarge the labels before the whole base line (including the labels) is scaled down by the target scale $1/s$ to produce the actual visualization. We say a label is *active* at scale factor s if it is selected as being visible at s; otherwise it is *inactive*. The (static) *placement* of an active label ℓ with target width w and anchor point p at scale factor s is determined by a shift position $t \in [0,1]$, i.e., the label is represented by the interval $[p-(1-t)ws, p+tws]$. A *dynamic placement* of ℓ is a placement of ℓ for each scale factor s at which ℓ is active. Since we consider invariant point placements, the shift position is the same for all scales. If we extrude the growing label segment with its constant shift position t along the y-axis from $y=0$ to some maximum scale factor $\mathrm{s}_{\max}$ we obtain a triangle whose apex is placed at the point p and whose top side is parallel to the x-axis, see Figure 2. We call this triangle the *extrusion* E of ℓ. The shift position t determines the slant of E, but for a label ℓ of width w the width of E at any fixed scale factor s is ws independent of t. Let the *trace* $\mathrm{tr}_s(E)$ of E at scale factor s be the intersection of E with the horizontal line $y=s$. By definition $\mathrm{tr}_s(E)$ corresponds to the placement of ℓ at s if ℓ is active at s.

If the extrusions E and E' of two labels intersect at some scale factor s this means that the two labels ℓ and ℓ' overlap at scale $1/s$. A standard requirement in point labeling, however, is that all labels must be pairwise disjoint [8]. Accordingly, at most one of ℓ or ℓ' can be active at scale factor s. Since one of the desiderata for consistent dynamic map labeling is that labels do not 'pop' during monotonous zooming in order to avoid flickering effects [3] we require that labels never vanish when zooming in and never appear when zooming out. This lets us define the *active range* of a label ℓ_i as an interval of scale factors $[0, a_i)$ for which ℓ_i is active. This active range implies that when zooming in the label ℓ_i appears exactly once at scale factor a_i and then remains active, or, conversely, when zooming out it disappears exactly once at scale factor a_i and remains inactive. The *truncated extrusion* T_i is the restriction of the extrusion E_i of ℓ_i to its active range $[0, a_i)$, see Figure 2 for an example. Now a consistent dynamic labeling for the points P corresponds to an assignment of a scale-independent shift position t_i and an active range $[0, a_i)$ for each label ℓ_i such that the truncated extrusions $\mathcal{T} = \{T_1, \ldots, T_n\}$ are pairwise disjoint. Hence we need to solve both a dynamic selection problem and a dynamic placement problem according to Been et al. [3]. Informally speaking, we can adjust the slant and the height of the truncated extrusions as long as they do not intersect each other.

Objective. A common objective in point labeling is to maximize the number of labeled points, and accordingly our goal is to maximize the *total active range length*, which is defined as the sum $H = \sum_{i=1}^{n} a_i$ of all active range lengths. Maximizing H corresponds to displaying a maximum number of labels integrated over all scale factors $s \in [0, \mathrm{s}_{\max}]$. This problem is known as the active range optimization problem (ARO) [4]. We consider two one-dimensional variants of ARO: In the discrete *k-position 1d ARO problem* the set of admissible shift positions is restricted to a subset $\mathcal{S}_i \subset [0,1]$ of cardinality $|\mathcal{S}_i| \leq k$. In the general *sliding 1d ARO problem* any shift position in $[0,1]$ is admissible.

3 A dynamic program for k-position 1d ARO

In this section we give a dynamic program for computing an optimal solution for the k-position version of the 1d ARO problem. For ease of notation we define two dummy points p_0 and p_{n+1}, where $p_0 = \min\{p_i - \mathrm{s}_{\max} w_i \mid 1 \leq i \leq n\}$ and $p_{n+1} = \max\{p_i + \mathrm{s}_{\max} w_i \mid 1 \leq i \leq n\}$. The only shift position of ℓ_0 is $\mathcal{S}_0 = \{0\}$ and the only shift position of ℓ_{n+1} is $\mathcal{S}_{n+1} = \{1\}$. Both labels have width 1. It is easy to see that in any optimal solution the height of T_0 and T_{n+1} must be $\mathrm{s}_{\max}$ since none of the extrusions T_i can intersect T_0 or T_{n+1}.

For a pair of points p_i and p_j with $i < j$ and shift positions $k_i \in \mathcal{S}_i$ and $k_j \in \mathcal{S}_j$ we define the free space $\Delta(i, j, k_i, k_j)$ as the polygon bounded by the line $s=0$, the supporting line of the right edge of T_i in shift position k_i, the supporting line of the left edge of T_j in shift position k_j, and, if the two supporting lines of T_i and T_j do not intersect below $\mathrm{s}_{\max}$, the line $s = \mathrm{s}_{\max}$. See Figure 3 for an example. Let $\mathcal{A}[i, j, k_i, k_j]$ be the maximum total active range height for the points $p_{i+1}, \ldots, p_{j-1}$, where all truncated extrusions $T_{i+1}, \ldots T_{j-1}$ are contained in $\Delta(i, j, k_i, k_j)$.

We observe that the tallest truncated extrusion T_l $(i < l < j)$ in any optimal solution of the subinstance I induced by $\Delta(i, j, k_i, k_j)$ must touch the left, right, or top boundary of $\Delta(i, j, k_i, k_j)$, otherwise we could improve the total active range height. We use T_l in order to split I into two smaller independent subinstances I' and I'' induced by $\Delta(i, l, k_i, k_l)$ and $\Delta(l, j, k_l, k_j)$, see Figure 3. For each $l = i+1, \ldots, j-1$ let h_{l,k_l}^{i,j,k_i,k_j} denote

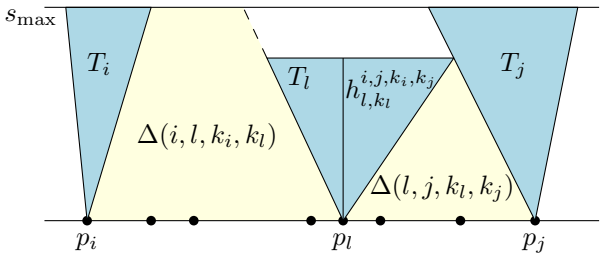

Figure 3: The subinstance induced by $\Delta(i,j,k_i,k_j)$ is split into two smaller independent subinstances by T_l.

the height at which T_l at shift position $k_l \in \mathcal{S}_l$ first hits a non-bottom edge of $\Delta(i,j,k_i,k_j)$. We initialize $\mathcal{A}[i,i+1,\cdot,\cdot] = 0$ for all $i = 0,\ldots,n$ and then recursively define $\mathcal{A}[i,j,k_i,k_j] = \max\{\mathcal{A}[i,l,k_i,k_l]+h_{l,k_l}^{i,j,k_i,k_j}+\mathcal{A}[l,j,k_l,k_j] \mid i<l<j \text{ and } k_l \in \mathcal{S}_l\}$. By definition of $\mathcal{A}$ the solution to the ARO problem is $\mathcal{A}[0,n+1,0,1]$. We can compute the value $\mathcal{A}[0,n+1,0,1]$ by dynamic programming in $O(n^3k^3)$ time: each of the $O(n^2k^2)$ values in $\mathcal{A}$ is defined as the maximum of a set of $O(nk)$ values, each of which can be computed by two table look-ups and two $O(1)$-time line intersection queries.

The correctness of the above dynamic program follows by induction on the number of points in a subinstance. Clearly for an empty subinstance $\Delta(i,i+1,k_i,k_{i+1})$ the maximum total active range height $\mathcal{A}[i,i+1,k_i,k_{i+1}]$ is 0. Let's consider a subinstance induced by $\Delta(i,j,k_i,k_j)$, where $j-i=r$ and assume by induction that the values in $\mathcal{A}$ are correct for all subinstances $\Delta(i',j',k_{i'},k_{j'})$, where $j'-i' < r$. Let $\mathcal{B}$ be an optimal active range assignment of the labels $\ell_{i+1},\ldots,\ell_{j-1}$ within the free space $\Delta(i,j,k_i,k_j)$ and let $H(\mathcal{B})$ be its value. Let further T_l be a tallest truncated extrusion with shift position k_l in $\mathcal{B}$. Obviously T_l must have height h_{l,k_l}^{i,j,k_i,k_j} if $\mathcal{B}$ is optimal. Since our algorithm explicitly considers all labels and all shift positions as candidates for the tallest truncated extrusion, it also considers T_l and its shift position k_l, which splits the given instance into two independent subinstances with free spaces $\Delta(i,l,k_i,k_l)$ and $\Delta(l,j,k_l,k_j)$. Since $l-i<r$ and $j-l<r$ we know that $\mathcal{A}[i,j,k_i,k_j] \geq \mathcal{A}[i,l,k_i,k_l] + h_{l,k_l}^{i,j,k_i,k_j} + \mathcal{A}[l,j,k_l,k_j] = H(\mathcal{B})$.

Since the free space $\Delta(0,n+1,0,1)$ is chosen such that none of the truncated extrusions $T_1,\ldots,T_n$ can touch T_0 or T_{n+1}, $\mathcal{A}(0,n+1,0,1)$ indeed contains the value of an optimal solution to the k-position ARO problem. We can easily augment the algorithm to keep track of the pair (l,k_l) that achieved the maximum value in order to reconstruct the solution by backtracking from $\mathcal{A}(0,n+1,0,1)$. We summarize this result in the following theorem.

Theorem 1 *Given n points $P = \{p_1,\ldots,p_n\}$ on the x-axis, a label ℓ_i of base width w_i for each point p_i, and a set $\mathcal{S}_i \subset [0,1]$ of at most k shift positions for each label ℓ_i, we can compute an optimal solution to the k-position 1d ARO problem in $O(n^3k^3)$ time and $O(n^2k^2)$ space.*

We note that this algorithm generalizes the $O(n^3)$-time algorithm of Been et al. [4] for the 1-position 1d ARO problem, where each label has only a single available shift position.

4 An FPTAS for general 1d sliding ARO

In this section we present an FPTAS for approximating the optimal solution of the sliding 1d ARO problem within a factor of $(1-\varepsilon)$. The idea of the FPTAS is based on uniformly discretizing the interval $[0,1]$ of shift positions. Let $k>0$ be an integer and define the set of shift positions $\mathcal{S}^k = \{i/k \mid i \in \mathbb{Z},\ 0 \leq i \leq k\}$. For an instance I of the sliding 1d ARO problem consisting of a point set $P = \{p_1,\ldots,p_n\}$ and corresponding label set $\mathcal{L}$, we consider instead the $(k+1)$-position 1d ARO problem for the instance I' consisting of P, $\mathcal{L}$, and the shift position sets $\mathcal{S}_i = \mathcal{S}^k$ for $1 \leq i \leq n$. By Theorem 1 this instance I' can be solved in $O(n^3k^3)$ time and $O(n^2k^2)$ space using the dynamic programming algorithm of Section 3. In the following theorem we show that this approach gives an FPTAS for the original sliding 1d ARO problem.

Theorem 2 *Given n points $P = \{p_1,\ldots,p_n\}$ on the x-axis and a corresponding label set $\mathcal{L} = \{\ell_1,\ldots,\ell_n\}$, where label ℓ_i has base width w_i, we can compute a $(1-\varepsilon)$-approximate solution to the sliding 1d ARO problem in $O(n^3(1/\varepsilon)^3)$ time and $O(n^2(1/\varepsilon)^2)$ space.*

Proof. We need to show that for a suitably chosen parameter $k = k(\varepsilon)$ the optimal solution for the $(k+1)$-position 1d ARO instance I' as defined above is actually a $(1-\varepsilon)$ approximate solution for the sliding 1d ARO instance I. Let us assume that we know an optimal solution $A^\star$ for I, i.e., a shift position $t_i \in [0,1]$ and an active range $[0,a_i) \subseteq [0,\mathrm{s}_{\max}]$ for each label ℓ_i. Since $A^\star$ is optimal, each truncated extrusion T_i has either height $a_i = \mathrm{s}_{\max}$ or touches the left or right supporting line of another, taller, truncated extrusion T_j. In the latter case a_i is the smallest scale factor, where the extrusions E_i and E_j intersect.

For proving the approximation factor we derive a discretized solution A' from $A^\star$, where each shift position $t_i' \in \mathcal{S}^k$ and the active ranges are shortened to $[0,a_i') \subseteq [0,a_i)$ as to satisfy the label disjointness property. For every shift position t_i in $A^\star$ we define the new shift position t_i' in A' as follows

$$t_i' = \begin{cases} \lfloor kt_i \rfloor/k & \text{if } t_i < 1/2 \\ \lceil kt_i \rceil/k & \text{if } t_i \geq 1/2. \end{cases}$$

In other words, we tilt T_i towards its "heavier" side until it reaches a shift position in the set $\mathcal{S}^k$. Due to the tilting the truncated extrusions are no longer necessarily disjoint and we need to shorten the active ranges for some labels. Figure 4 shows an example.

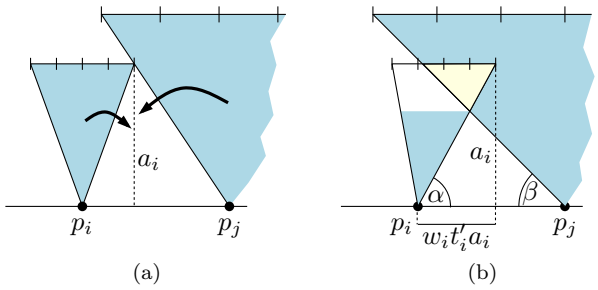

Figure 4: Discretizing the shift positions of two labels for $k = 4$.

Let T'_i and T'_j be two tilted truncated extrusions that intersect in their interior. Without loss of generality let T'_i be the smaller one such that, before the tilting, its top right corner was touching the left edge of T'_j as in Figure 4a. We first consider the case that T'_i and T'_j are tilted towards each other. Then the right edge of T'_i, the left edge of T'_j, and the horizontal line $y = a_i$ define a triangle D as in Figures 4b and 5. We decrease the active range of label ℓ_i to $[0, a'_i)$, where $a'_i = a_i - h$ for the height h of D. Obviously the truncated extrusions T'_i and T'_j no longer intersect in their interior for the new active range $[0, a'_i)$.

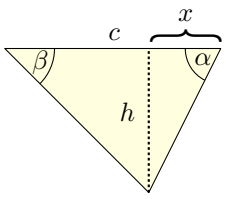

Figure 5: Intersection triangle D.

Next, we bound the height h of D. Let α be the angle between the right edge of T'_i and the x-axis. The same angle α is found at the top right corner of D. From Figure 4b we obtain that $\tan(\alpha) = a_i/(w_i t'_i a_i) = 1/(w_i t'_i)$ and from Figure 5 that $\tan(\alpha) = h/x$, where x is distance between the base point of the height h on the top side c and the top right corner. Since T'_i is tilted to the right and T'_j to the left we have $t'_i \geq 1/2$ and $(1 - t'_j) \geq 1/2$. By definition of the new shift positions t'_i and t'_j we know that the length of the top side c of D is at most $(w_i a_i + w_j a_i)/k$. This is because at scale factor s the tilt moves each truncated extrusion with base width w horizontally by at most a $1/k$ fraction of its width ws at s. With $x \leq c$ this yields

$$h = \frac{x}{w_i t'_i} \leq \frac{c}{w_i/2} \leq \frac{2a_i}{k} \frac{w_i + w_j}{w_i}.$$

Similar reasoning for the angle β in Figures 4b and 5 yields $\tan(\beta) = a_i/(w_j(1 - t'_j)a_i) = h/(c - x)$ and subsequently $h \leq 2a_i/k \cdot (w_i + w_j)/w_j$. Again without loss of generality we assume that $w_i \geq w_j$ and obtain $\min\{(w_i + w_j)/w_i, (w_i + w_j)/w_j\} = (w_i + w_j)/w_i \leq 2$. So we can finally bound the height of D by $h \leq 4a_i/k$.

We still need to consider the case that both truncated extrusions are tilted in the same direction, say to the left (the case that both are tilted to the right is symmetric). A conflict can still occur if T'_j is tilted further to the left than T'_i. The triangle D is defined as before, but now we know that the length of the side c is at most $w_j a_i/k$. Since T'_j is tilted to the left we still have $(1 - t'_j) \geq 1/2$. Now we argue about the angle β using the same identities as before and obtain $h \leq c/(w_j(1 - t'_j)) \leq 2a_i/k$.

In the case that T'_i and T'_j are tilted away from each other obviously no conflict can occur. Furthermore, the analysis still holds for conflicts involving the top left corner of T'_i instead of the top right corner.

So each truncated extrusion T_i of height a_i is shortened by at most $4a_i/k$ due to the discretization of the shift positions, or, equivalently, $a'_i \geq (1 - 4/k)a_i$. If we set $k = 4/\varepsilon$ we arrive at $\sum_{i=1}^n a'_i \geq (1 - \varepsilon) \sum_{i=1}^n a_i$. □

5 Discussion

In this paper we studied an extension of the initial ARO problem, introduced by Been et al. [4], where we additionally allow to slide the labels. Our dynamic programming approach for discrete k-position 1d ARO is a generalization of their approach to solve simple 1d ARO with proportional dilation. It shows that k-position 1d ARO can be solved in polynomial time.

Based on the dynamic program for k-position 1d ARO, we further derived an FPTAS for sliding 1d ARO by suitably discretizing the set of allowed shifts for the labels. While this shows that we can approximate the optimal value arbitrarily closely in polynomial time, the complexity of sliding 1d ARO is still open. The main difficulty in devising an NP-hardness proof is that the problem becomes efficiently solvable when every label has only a polynomial number of relevant sliding positions. It thus seems difficult to encode binary decisions as label positions.

Note that in our model the shift position of each label, once selected, remains fixed for all scales. For the k-position model this is actually required in order to avoid jumping labels, whereas in a more general sliding model we leave as an open problem to determine a continuous function that defines the label position for every scale. Here we might require that this function is monotone or that its slope is bounded.

In practice it is common that some points are more important than others and hence the active ranges of their labels should be more influential in the objective function. More precisely, let $\gamma_i > 0$ be a weight for each point p_i. We can then optimize the weighted total active range length $H_\gamma = \sum_{i=1}^{n} \gamma_i a_i$ instead of H. It is easy to see that both the dynamic programming algorithm and the FPTAS remain valid for optimizing H_γ.

Another problem variant is to use a non-linear objective function, motivated by the observation that H favors active labels at large values of s, i.e., in maps with small scales $1/s$. It might be reasonable in practice to choose a logarithmic function over a linear function for measuring the active ranges. Using the objective function $H_{\log} = \sum_{i=1}^{n} \log a_i$ instead of H has the effect that doubling the scale range at which a label is active has a fixed impact on the objective function regardless of the actual scale. The dynamic programming algorithm can immediately deal with $H_{\log}$. Even the approximation scheme of Section 4 remains an FPTAS under the mild additional assumptions that the minimum scale factor is 1 (instead of 0), that each $a_i \geq 2$, and that $\varepsilon \leq 1/2$.

Ultimately, the challenge in 2d dynamic map labeling is to consistently support multiple modes of interaction (zooming, panning, rotations) using a slider model for the labels. In this sense, our results are a first step towards consistent dynamic labeling of 2d zoomable maps with sliding labels. Unfortunately, our algorithms do not easily generalize to 2d point sets. In fact, it can be easily seen that k-position 2d ARO and sliding 2d ARO are both NP-hard. The result of Been et al. [4] essentially shows that 1-position 2d ARO, and thus also k-position 2d ARO is NP-hard. For the sliding 2d ARO problem deciding whether all labels may be active at all scales amounts to deciding whether all labels can be placed at scale $s_{\max}$. A slight modification of the NP-hardness proof for map labeling in the four-slider model [16] shows that this problem is NP-hard, even if all labels are unit squares.

Acknowledgments

We thank an anonymous reviewer for helpful suggestions. A. Gemsa and M. Nöllenburg are supported by the Concept for the Future of KIT under project YIG 10-209 within the framework of the German Excellence Initiative and by a Google Research Award.

References

[1] P. K. Agarwal, M. van Kreveld, and S. Suri. Label placement by maximum independent set in rectangles. *Comput. Geom. Theory Appl.*, 11:209–218, 1998.

[2] K. Ali, K. Hartmann, and T. Strothotte. Label layout for interactive 3D illustrations. *Journal of the WSCG*, 13(1):1–8, 2005.

[3] K. Been, E. Daiches, and C. Yap. Dynamic map labeling. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):773–780, 2006.

[4] K. Been, M. Nöllenburg, S.-H. Poon, and A. Wolff. Optimizing active ranges for consistent dynamic map labeling. *Comput. Geom. Theory Appl.*, 43(3):312–328, 2010.

[5] B. Bell, S. Feiner, and T. Höllerer. View management for virtual and augmented reality. In *ACM Sympos. on User Interface Software and Technology (UIST'01)*, pages 101–110, 2001.

[6] M. Formann and F. Wagner. A packing problem with applications to lettering of maps. In *Proc. 7th Annual ACM Sympos. on Computational Geometry (SoCG'91)*, pages 281–288, 1991.

[7] A. Gemsa, M. Nöllenburg, and I. Rutter. Consistent labeling of rotating maps. In *Proc. 12th Algorithms and Data Structures Symposium (WADS'11)*. To appear, 2011.

[8] E. Imhof. Positioning names on maps. *The American Cartographer*, 2(2):128–144, 1975.

[9] G. W. Klau and P. Mutzel. Optimal labeling of point features in rectangular labeling models. *Mathematical Programming (Series B)*, pages 435–458, 2003.

[10] K. D. Mote. Fast point-feature label placement for dynamic visualizations. *Information Visualization*, 6(4):249–260, 2007.

[11] M. Nöllenburg, V. Polishchuk, and M. Sysikaski. Dynamic one-sided boundary labeling. In *Proc. 18th ACM SIGSPATIAL Int'l Conf. Advances in Geographic Information Systems*, pages 310–319. ACM Press, 2010.

[12] K. Ooms, W. Kellens, and V. Fack. Dynamic map labeling for users. In *Proc. 24th Int'l Cartographic Conference (ICC'09)*, Santiago, Chile, 2009.

[13] I. Petzold, G. Gröger, and L. Plümer. Fast screen map labeling—data-structures and algorithms. In *Proc. 23rd Internat. Cartographic Conf. (ICC'03)*, pages 288–298, Durban, South Africa, 2003.

[14] S.-H. Poon and C.-S. Shin. Adaptive zooming in point set labeling. In *Proc. 15th Internat. Sympos. Fundam. Comput. Theory (FCT'05)*, volume 3623 of *Lecture Notes Comput. Sci.*, pages 233–244. Springer-Verlag, 2005.

[15] T. Strijk and M. van Kreveld. Practical extensions of point labeling in the slider model. *GeoInformatica*, 6(2):181–197, 2002.

[16] M. van Kreveld, T. Strijk, and A. Wolff. Point labeling with sliding labels. *Comput. Geom. Theory Appl.*, 13:21–47, 1999.

[17] F. Wagner and A. Wolff. A practical map labeling algorithm. *Comput. Geom. Theory Appl.*, 7:387–404, 1997.

[18] F. Wagner, A. Wolff, V. Kapoor, and T. Strijk. Three rules suffice for good label placement. *Algorithmica*, 30(2):334–349, 2001.

[19] A. Wolff and T. Strijk. The Map-Labeling Bibliography, 1996.