Optimizing Active Ranges for Consistent Dynamic Map Labeling

Ken Been
Computer Science Dept.
Yeshiva University
New York, NY, USA
kbeen@yu.edu

Martin Nöllenburg[*]
Faculty of Informatics
Karlsruhe University
Germany
noellenburg@iti.uka.de

Sheung-Hung Poon
Dept. Computer Science
National Tsing Hua University
Hsin-Chu, Taiwan
spoon@cs.nthu.edu.tw

Alexander Wolff
Faculteit Wiskunde en Informatica
TU Eindhoven
The Netherlands
www.win.tue.nl/~awolff

ABSTRACT

Map labeling encounters unique issues in the context of dynamic maps with continuous zooming and panning—an application with increasing practical importance. In *consistent* dynamic map labeling, distracting behavior such as popping and jumping is avoided. In the model for consistent dynamic labeling that we use, a label becomes a 3d-solid, with scale as the third dimension. Each solid can be truncated to a *single* scale interval, called its *active range*, corresponding to the scales at which the label will be selected. The *active range optimization (ARO)* problem is to select active ranges so that no two truncated solids overlap and the sum of the heights of the active ranges is maximized. The *simple* ARO problem is a variant in which the active ranges are restricted so that a label is never deselected when zooming in. We investigate both the general and simple variants, for 1d- as well as 2d-maps. The 1d-problem can be seen as a scheduling problem with geometric constraints, and is also closely related to geometric maximum independent set problems.

Different label shapes define different ARO variants. We show that 2d-ARO and general 1d-ARO are NP-complete, even for quite simple shapes. We solve simple 1d-ARO optimally with dynamic programming, and present a toolbox of algorithms that yield constant-factor approximations for a number of 1d- and 2d-variants.

Categories and Subject Descriptors

I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling

[*]Supported by the German Research Foundation (DFG) under grant WO 758/4-3.

General Terms

Algorithms, Theory

Keywords

Dynamic map labeling, approximation algorithms, NP-hardness

1. INTRODUCTION

Recent years have seen tremendous improvements in Internet-based, geographic visualization systems that provide continuous zooming and panning (e.g., Google Earth), but relatively little attention has been paid to special issues faced by map labeling in such contexts. In addition to the need for interactive speed, several desiderata for a *consistent dynamic labeling* were identified by Been et al. [2]: labels do not pop in and out or jump (suddenly change position or size) during panning and zooming, and the labeling does not depend on the user's navigation history. As an example, Google Earth does not currently satisfy these desiderata.

Model.

We adapt the labeling model of Been et al. [2] as follows. In *static labeling* the key operations are selection and placement—select a subset of the labels that can be placed without overlap. Let each label L be defined in its own label coordinates. A *static placement* of L is its image $\hat{L}$ in world coordinates under a transformation composed of translation, rotation, and dilation (see Figure 1a). A further transformation takes a portion of world coordinates to the screen, dilating by factor $1/s$; we define s to be the scale. Note that s is the inverse of cartographic scale.

In *dynamic labeling* we select *at each scale* a subset of labels that can be placed without overlap. To meet the desiderata for consistent dynamic labeling we (1) define a *dynamic placement* of L to be a function that assigns a static placement $\hat{L}^s$ to each scale $s \geq 0$; (2) require that each dynamic placement be continuous with scale; (3) define *dynamic selection* to be a Boolean function of scale; and (4) require that each label L_i, $1 \leq i \leq n$, be selected precisely on a single interval of scales, $[a_i, A_i]$, which is called the *active range* of L_i.

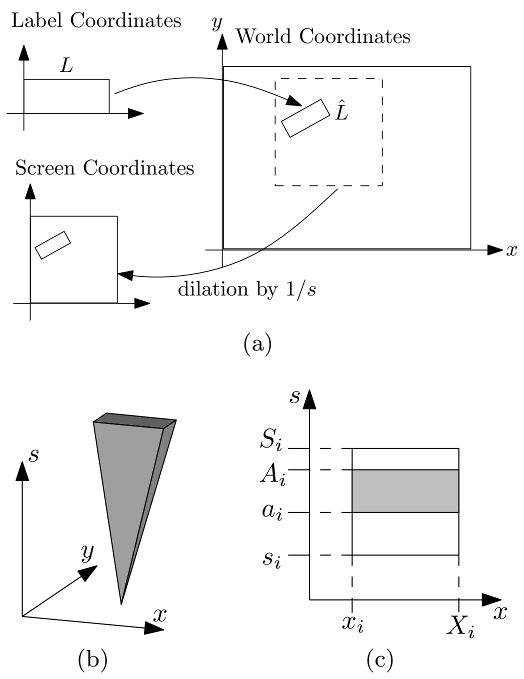


Figure 1: (a) Label, world and screen coordinates. (b) A dynamic placement of a 2d-label is a solid in extended world coordinates. Here: with invariant point placement and proportional dilation. (c) A 1d-label with constant dilation, available range $[s_i, S_i]$, and active range $[a_i, A_i]$.

Define *extended world coordinates* by adding a scale dimension to world coordinates. We can think of dynamic placement as mapping a label L into extended world coordinates such that the cross section of the image of L at scale s is the static placement $\hat{L}^s$. Let $S_{\max}$ be a universal maximum scale for all labels. We define the *available range* of L_i to be an interval of scales, $[s_i, S_i] \subseteq [0, S_{\max}]$, in which label L_i "wants" to be selected. For example, street labels are available at low scales and country labels at high scales. We require $[a_i, A_i] \subseteq [s_i, S_i]$. Let $E_i = \bigcup_{s\in[s_i,S_i]} \hat{L}_i^s$. Since dynamic placement is continuous with scale, E_i is a solid defined by sweeping the label shape along a continuous curve that is monotonic in scale, with the rotation and dilation factors at each scale given by continuous functions as in Fig. 1b. Let T_i be the restriction of E_i to $s \in [a_i, A_i]$. We call E_i the *extrusion* of L_i and T_i its *truncated extrusion*.

The extrusion shapes are determined by the label shape and the translation, rotation and dilation functions that compose the dynamic placement. We restrict our attention to certain classes of extrusions. Our 2d-labels are rectangular; we also consider 1d-labels, which are segments $[x_i, X_i]$ on the x-axis—see Fig. 1c. For translation, we consider only *invariant point placements*, in which a particular point on the label always maps to the same location in world coordinates, so the label never "slides". Our rotation functions are constant, and yield axis-aligned labels.

Let D^L be the dilation function embedded in the dynamic placement of L. We consider three classes of such functions. If $D^L(s) = c$ for a constant $c > 0$, then label size is fixed in world coordinates and inversely proportional to scale on screen, like the geographic features. The solid is then a "straight" extrusion. If $D^L(s) = bs$ for a constant $b > 0$, then label size is fixed on screen and proportional to scale in world coordinates. The solid is then a label-shaped cone with apex at $s = 0$ as in Fig. 1b. With invariant point placements, the cone contains the vertical line through its apex. The cone might be symmetric to that line (e.g., for labeling a region) or might have a vertical side incident to it (e.g., for labeling a point). Finally, we also consider, in a more general setting, functions of the form $D^L(s) = bs + c$ for constants $b > 0$ and $c \neq 0$. The solid in this case is a portion of a cone with apex at $-c/b$.

Objective.

Let $\mathcal{E}$ denote the set of all extrusions, and assume we are given an available range for each. For a set $\mathcal{T}$ of truncated extrusions, define $H(\mathcal{T}) = \sum_{i=1}^{n}(A_i - a_i)$ to be the *total active range height*. This is the same as integrating over all scales the function $f(s)$ that counts the number of labels selected at scale s. The *(general) active range optimization (ARO)* problem is to choose the active ranges so as to maximize H, subject to the constraint that no two truncated extrusions overlap. This is the dynamic analogue of placing the maximum number of labels without overlap in the static case. We concentrate solely on maximizing H since alternative optimization criteria like maximizing the minimum active range height do not make much sense in practice, e.g., if there is a label with a very small available range. We call any set of active ranges that correspond to non-overlapping truncated extrusions a *solution*. It is of theoretical and practical interest to also consider a version of the problem in which all labels are available at all scales and a label is never deselected when zooming in—i.e., $[s_i, S_i] = [0, S_{\max}]$ and $a_i = 0$ for all i. We call this variant of ARO *simple*. Note that the 1d-version of ARO can be seen as a scheduling problem with geometric constraints, and is also closely related to geometric maximum independent set problems.

Previous Work.

Map labeling has been identified as an important application area by the ACM Computational Geometry Impact Task Force [3], and has been the focus of extensive algorithmic investigation [14]. The vast majority of research on this topic covers *static* labeling. A typical goal is to select and place labels without overlap while optimizing an objective function. The objective function might be simply the number of labels [1, 13], or it might incorporate multiple cartographic criteria [5]. There are many variations possible, and most have been shown to be NP-hard [6, 8, 13].

For *dynamic* labeling, Petzold et al. [10, 11] use a preprocessing phase to generate a data structure that is searched during interaction to produce a labeling for the current scale and view area. Poon and Shin [12] build a hierarchy of precomputed solutions, and interpolation between these produces a solution for any scale. Neither of these approaches satisfies the consistency desiderata. In addition to introducing consistency for dynamic map labeling, Been et al. [2] show that simple 2d-ARO is NP-complete for arbitrary star-shaped labels, and implement a simple heuristic solution.

Outline.

We investigate the complexity of ARO in Section 2. We prove that general 1d-ARO with constant dilation is NP-complete, even if all extrusions are squares, and that simple

d	extrusion shape – technique	ARO	dilation	NP-compl.	approx.	running time	see
1	triangles – dynamic program	simple	bs	no	optimal	$O(n^3)$	Thm. 3
	unit squares – line stabbing	general	c	?	2/3	$O(n \log n)$	Thm. 6
	unit-height rect. – MIS		c	yes	1/3	$O(n \log n)$	Thm. 4
	unit-width rect. – line stabbing		c	yes	1/2	$O(n \log n)$	Thm. 5
	rectangles – divide & conquer		c	yes	$1/\log n$	$O(n \log n)$	Thm. 7
	segments of congruent triangles		bs	?	1/2	$O((k+n) \log n)$	Thm. 10
	congruent trapezoids		$bs + c$	?	1/2	$O(k \log n + n \log^2 n)$	Thm. 8
2	congruent square cones	simple	bs	yes	1/4	$O((k+n) \log^2 n)$	Cor. 1
	congruent square cones		bs	yes	1/8	$O(n \log^3 n)$	Cor. 2
	arbitrary square cones		bs	yes	1/24	$O(n \log^3 n)$	Thm. 12
	segments of congruent square cones	general	bs	yes	1/4	$O((k+n) \log^2 n)$	Thm. 11
	congruent frusta		$bs + c$	yes	$1/(4W)$	$O(n^4)$	Thm. 9

Table 1: Results attained in this paper, where k is the number of pairwise intersections between extrusions, W is the width ratio of top over bottom side, and d is the dimension of the ARO problems.

2d-ARO with proportional dilation is NP-complete, even if all extrusions are congruent square cones. Both proofs are by reduction from PLANAR3SAT, the latter using 3d gadgets. We present an algorithmic study of ARO in Section 3 by developing an algorithmic toolbox containing both new techniques and new applications of known techniques to solve several variants of ARO problems. One of our algorithms is exact, the others yield approximations. Table 1 summarizes our results.

Since we have just started to investigate these new problems, many questions remain unsolved—most notably: does any of our problems have a polynomial-time approximation scheme (PTAS)?—and require future effort, see Section 4.

2. COMPLEXITY

In this section, we prove that two variants of ARO are NP-complete. Both proofs use a reduction from the NP-hard problem PLANAR3SAT [9]. An instance of PLANAR3SAT is a 3SAT formula φ whose variable-clause graph G_φ is planar. Note that G_φ can be laid out such that all variables correspond to points on the x-axis and clauses to non-crossing three-legged "combs" above or below the x-axis [8].

THEOREM 1. *General 1d-ARO with constant dilation is NP-complete; i.e., given a set $\mathcal{E} = \{E_1, \dots, E_n\}$ of axis-aligned rectangular extrusions in the plane and a real $K > 0$, it is NP-complete to decide whether there is a set of pairwise disjoint truncated extrusions $\mathcal{T} = \{T_1, \dots, T_n\}$ with $T_1 \subseteq E_1, \dots, T_n \subseteq E_n$ and $H(\mathcal{T}) \geq K$. The problem remains NP-complete when restricted to instances where all extrusions are squares and each has one of three sizes, all extrusions are unit-width rectangles and each has one of two heights, or all extrusions are unit-height rectangles and each has one of two widths.*

PROOF. For membership in $\mathcal{NP}$, decompose each E_i into $O(n)$ horizontal strips determined by the lines $\{s = s_i, s = S_i \mid 1 \leq i \leq n\}$. Clearly there is an optimal solution that corresponds to a union of such strips. So we can guess a subset of the strips and then check in polynomial time whether (a) strips from the same square are consecutive, (b) no two strips overlap, and (c) their total height is at least K.

To show hardness, let φ be an instance of PLANAR3SAT. We first treat square extrusions; i.e., we construct a set $\mathcal{E}_\varphi$ of squares as illustrated in Fig. 2 and fix a threshold $K > 0$ such that $H(\mathcal{S}(\mathcal{E}_\varphi)) \geq K$ for an optimal solution $\mathcal{S}(\mathcal{E}_\varphi)$ if and only if φ is satisfiable.

The squares in $\mathcal{E}_\varphi$ have side lengths 1, 5, and 7. We refer to a square of side length j as a j-square. In Fig. 2 all 5-squares are highlighted by bold boundaries. Each 7-square contains a vertically and horizontally centered chain of five (unique) 1-squares. Thus the 7-square can contribute at most three units to H if the 1-squares contribute one unit each. Note that this is more than if the 7-square contributes seven units and all 1-squares contribute zero. The contribution of a 7-square is a (7×3)-rectangle that can either appear above or below the chain of 1-squares. We say that the 7-square is in upper or lower *state*, which gives us a means to encode Boolean values. The contributing part of each square is shaded in Fig. 2, and each type of square has its own degree of shading.

Each square in $\mathcal{E}_\varphi$ belongs to a variable gadget, a literal gadget, or a clause gadget. These gadgets correspond one-to-one to the n variables, $3m$ literals, and m clauses of φ, respectively.

The gadget of a variable x consists of a horizontal chain of n_x 7-squares (thus containing $5n_x$ 1-squares), where n_x is a constant that depends on the appearance frequency of x in φ. Adjacent 7-squares overlap so that their states must alternate. We let x being *true* correspond to the leftmost 7-square of the gadget being in upper state. It is clear that the gadget of x contributes at most $8n_x$ units to H.

The gadget of a literal λ in φ consists of a 5-square E_λ containing five 1-squares, vertically centered. Number the 7-squares in each variable gadget from left to right. If λ is negated, then E_λ intersects the top edge of an odd-numbered (or bottom-edge of an even-numbered) 7-square of the corresponding variable gadget. Otherwise parity flips; see the positions of E_y and $E_{\neg z}$ in Fig. 2. Each literal gadget contributes at most seven units to H.

The clause gadget forms the afore-mentioned three-legged comb, with a leg for each literal. Each leg has a vertical segment, and the left and right legs also have horizontal segments that contain an even number of 7-squares. The literal gadgets connect the variable to the clause leg, see Fig. 2. The leg of literal λ consists of a fixed number m_λ (depending on φ) of 7-squares and $5m_\lambda$ 1-squares, contributing at most $8m_\lambda$ units to H.

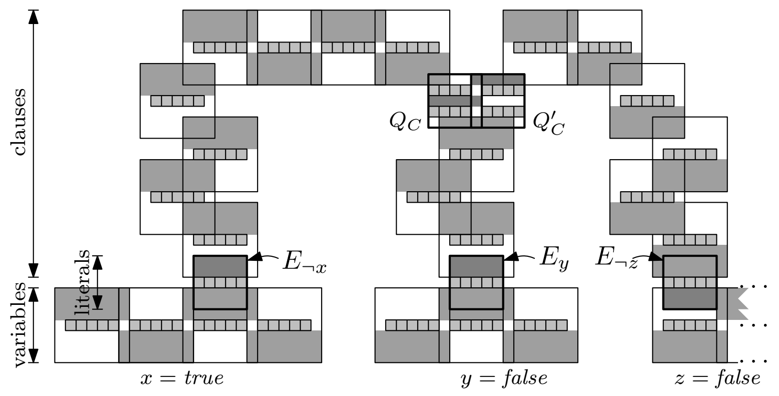


Figure 2: The gadgets of our reduction for the clause $C = (\neg x \vee y \vee \neg z)$.

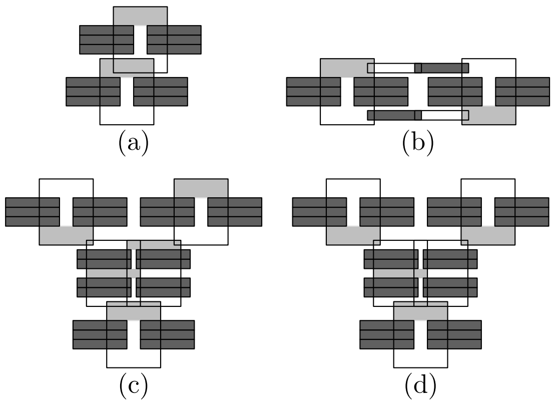


Figure 3: Gadgets for unit-width rectangles: vertical chain (a), horizontal chain (b), clauses that evaluate to *true* (c) and to *false* (d).

At the center of the gadget for clause C are two 5-squares Q_C and Q'_C, each containing eight 1-squares as depicted in Fig. 2. Let $Q \in \{Q_C, Q'_C\}$. Note that there are three ways that Q can contribute (one unit) to H, and that Q overlaps with the last squares of two legs corresponding to literals in C. Assume that the legs and the literal gadgets contribute maximally to H. Then, if the two literals corresponding to legs overlapping Q evaluate to *false*, only the middle unit-height strip of Q can contribute (one unit) to H. However, if all three literals evaluate to *false*, the two middle strips of Q_C and Q'_C *together* can contribute at most one unit since they overlap. In this case the clause gadget center (Q_C, Q'_C and the sixteen 1-squares in their union) contributes 17 units in total. On the other hand, if a literal in C evaluates to *true*, both Q_C and Q'_C contribute one unit, and the clause gadget center contributes 18 units.

To justify the assumption that the variable gadgets, literal gadgets, and clause legs contribute maximally to H, note that a gain of one unit in the clause gadget center would not pay for a loss of at least two units in any other location.

Let $K = (8\sum_{v\in\mathrm{Var}(\varphi)} n_v) + 7\cdot 3m + (8\sum_{\lambda\in\mathrm{Lit}(\varphi)} m_\lambda) + 18m$, where $\mathrm{Var}(\varphi)$ and $\mathrm{Lit}(\varphi)$ denote the variables and literals in φ, respectively. The summands of K correspond to the maximum contributions of all variable gadgets, literal gadgets, gadget legs, and clause gadget centers. A contribution to H of K can only be achieved if φ is satisfiable.

The set $\mathcal{E}_\varphi$ consists of $O(m^2)$ squares. The positions of all squares can be encoded in space quadratic in the length of an encoding of φ. The reduction can be performed in polynomial time.

The construction for unit-width rectangles is similar, see Fig. 3. Switching the roles of horizontal and vertical, the same construction holds for unit-height rectangles. $\square$

Surprisingly, in 2d even the *simple* ARO problem is hard.

THEOREM 2. *Simple 2d-ARO with proportional dilation is NP-complete; i.e., given a set $\{E_1, \ldots, E_n\}$ of congruent square cones and a real $K > 0$, it is NP-complete to decide whether there is a set of pairwise disjoint truncated extrusions $\mathcal{T} = \{T_1, \ldots, T_n\}$ with $T_1 \subseteq E_1, \ldots, T_n \subseteq E_n$ and $H(\mathcal{T}) \geq K$.*

PROOF. To see membership in $\mathcal{NP}$, note that in any optimal solution $\mathcal{S}$ all cones either reach $S_{\max}$ or touch another cone. Thus $\mathcal{S}$ can be constructed by guessing an order and inserting cones greedily. To show hardness, let φ be a planar 3SAT formula with m clauses. We construct a set $\mathcal{E}_\varphi$ of unit square cones and show that there is a threshold $K > 0$ such that $H(\mathcal{S}(\mathcal{E}_\varphi)) \geq K$ iff φ is satisfiable.

Similar to the 1d-case, we construct variable gadgets and three-legged combs connecting them to clause gadgets. Fig. 4 shows 3d-models of the gadgets; Fig. 5 shows 2d-projections. Each gadget consists of chains of cones. Their apices lie on a half-integer grid. Cones whose apices have L_∞-distance $1/2$ (1) touch at $S_{\max}/2$ $(S_{\max})$. Therefore, all cones are active and do not interfere in the range $[0, S_{\max}/2]$, but only every other cone in a chain can extend up to $S_{\max}$.

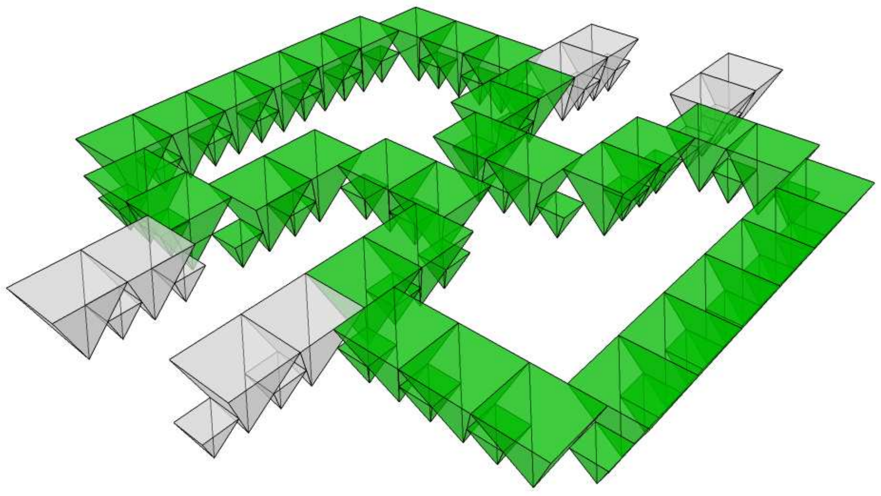
The variable gadget consists of an even cyclic chain. Numbering the cones clockwise starting at the top left, we denote the state where the odd (even) cones extend to the full height as *true* (*false*); see Fig. 4a, 5a, and 5b. At those positions where the cycle has an indentation we can connect literal pipes: a positive literal at the beginning (in clockwise order) of an indentation, and a negative literal at the end. Thus, if a literal evaluates to *false* (*true*), the corresponding literal pipe starts with a half (full) cone.

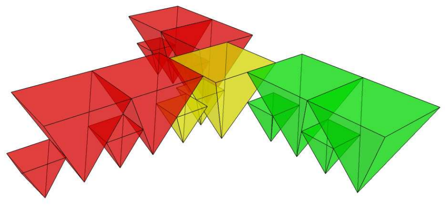
The clause gadget (Fig. 4b, 4c, 5c, 5d) consists of three pairwise adjacent cones so that at most one can be full size. Each of them is also adjacent to the last cone in a literal pipe. The literal pipes have an even number of cones, so the pipe for a *true* (*false*) literal ends in a half (full) cone. Thus, none of the clause cones can be full size iff all literals are *false*.

By construction, the total active range H of the variable and literal gadgets is independent of their truth values. A clause gadget contributes $2\,S_{\max}$ $(1.5\,S_{\max})$ to H if the corresponding clause is *true* (*false*). Thus φ is satisfiable iff the clause cones in total contribute $2m\,S_{\max}$ to H. $\square$

3. ALGORITHMIC TOOLBOX

We give a toolbox of six different algorithms to tackle several variants of 1d- and 2d-ARO problems. Some are only briefly covered since they are based on well-known techniques: dynamic programming, a left-to-right greedy algo-


(a) Variable gadget for $x = \mathit{true}$.


(b) One literal is *true* (green).

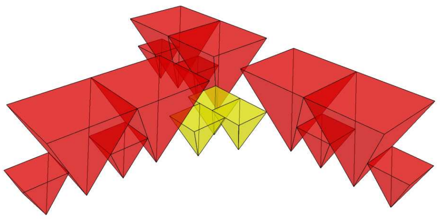
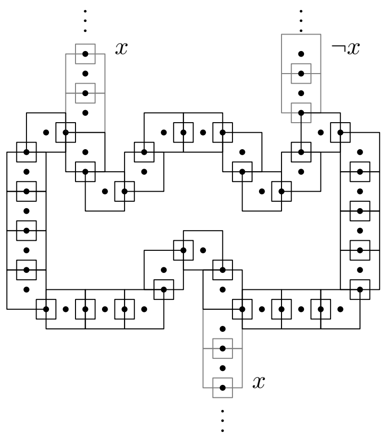
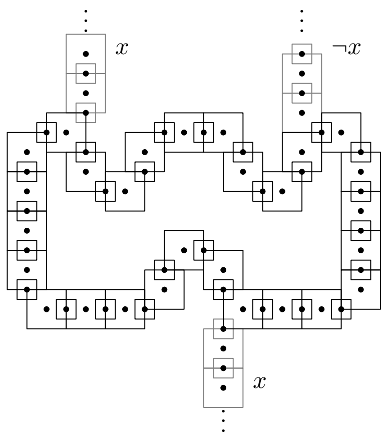

(c) All literals are *false* (red).

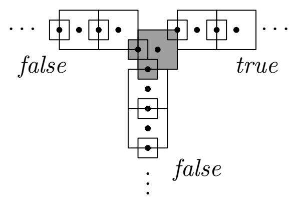
Figure 4: 3d-models of the variable and clause gadgets in Theorem 2 (with partial literal gadgets).



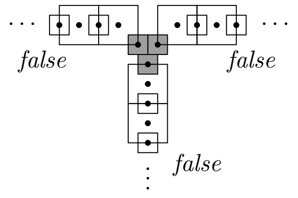
(a) Variable $x = \mathit{true}$.



(b) Variable $x = \mathit{false}$.



(c) One literal is *true*.



(d) All literals are *false*.

Figure 5: 2d-projections of the variable and clause gadgets in Theorem 2 (with partial literal gadgets).

rithm, line stabbing, and divide and conquer. We concentrate on two algorithms that apply new techniques: a top-to-bottom fill-down sweep, and a level-based small-to-large greedy algorithm.

3.1 Dynamic programming

Triangles.

We start by considering simple 1d-ARO with proportional dilation: each extrusion E_i is a triangle with apex on the x-axis and top side on the horizontal line $s = S_{\max}$. The truncated extrusions T_i differ only by having (possibly) lower top sides. Observe that in an optimal solution at least one T_i has height $S_{\max}$, and thus divides the problem into two independent subproblems. This is the essence of the dynamic programming solution.

THEOREM 3. *Simple 1d-ARO with proportional dilation can be solved in $O(n^3)$ time.*

PROOF. Let p_i be the apex of wedge-shaped extrusion E_i on the x-axis. For ease of notation define dummy wedges E_0 and E_{n+1} with apexes p_0 and p_{n+1}, and assume that $p_0, \ldots, p_{n+1}$ are sorted from left to right. For $i < j$, define the *free space* $\Delta(i, j)$ between p_i and p_j to be the triangular or trapezoidal space enclosed by $s = 0$, the right side of E_i, the left side of E_j, and possibly $s = S_{\max}$. Let $\mathcal{A}[i, j]$ be the optimal solution for $p_{i+1}, \ldots, p_{j-1}$ in $\Delta(i, j)$. In $\mathcal{A}[i, j]$, at least one of $T_{i+1}, \ldots, T_{j-1}$ must touch a non-bottom side of $\Delta(i, j)$, thus dividing the problem into two independent subproblems. For each $k = i+1, \ldots, j-1$, we denote by h_k the scale at which T_k first reaches a non-bottom side of $\Delta(i, j)$. Then $\mathcal{A}[i, j] = \max_{k=i+1}^{j-1}(\mathcal{A}[i, k] + h_k + \mathcal{A}[k, j])$, and the optimal solution for our problem is $\mathcal{A}[0, n+1]$. Each of the $O(n^2)$ entries in the dynamic programming table is computed in $O(n)$ time, giving $O(n^3)$ time in total. $\square$

3.2 Left-to-right greedy algorithm

Unit-height rectangles.

Van Kreveld et al. [13] presented the following greedy algorithm for maximum independent set (MIS) among axis-aligned rectangles of unit height. We are given a set $\mathcal{E}$ of unit-height rectangles. Until $\mathcal{E}$ is empty, repeatedly select the rectangle $E \in \mathcal{E}$ with leftmost right edge, and remove from $\mathcal{E}$ all remaining rectangles intersecting E. This takes $O(n \log n)$ time, and is a (1/2)-approximation for MIS [13]. It is not hard to see that the same algorithm yields a (1/3)-approximation for 1d-ARO.

THEOREM 4. *The maximum total active range height for a set of n rectangular extrusions of unit height can be approximated by a factor of $1/3$ in $O(n \log n)$ time.*

3.3 Line stabbing

We use line stabbing for unit squares and unit-width rectangles, i.e., general 1d-ARO with constant dilation and equal-size labels. Line stabbing is a special case of the shifting technique by Hochbaum and Maass [7]. The idea is to stab all extrusions with vertical lines of distance at least 1 such that each extrusion is stabbed by exactly one line and each line stabs at least one extrusion. The stabbing lines are numbered l_1 to l_k from left to right. Since all extrusions have unit width, those intersecting l_i do not overlap those intersecting l_{i+2}. Our approximate solutions make use of the *optimal* solutions for either the extrusions stabbed by a single line or those stabbed by two adjacent lines.

Unit-width rectangles.

For unit-width rectangles we partition the vertical stabbing lines into sets Λ_1 and Λ_2, containing all the stabbing lines with odd and even indices, respectively. A simple greedy algorithm computes the maximum total active range height for the rectangles intersecting a single stabbing line. Thus the solution $\mathcal{A}_i$ for all rectangles intersecting lines in Λ_i can be computed optimally. By the pigeon-hole principle it is clear that the solution $\mathcal{A}_i$ maximizing H is a (1/2)-approximation. It can be computed in $O(n \log n)$ time. Thus we obtain the following.

THEOREM 5. *The maximum total active range height for a set of n rectangular extrusions of unit width can be approximated by a factor of $1/2$ in $O(n \log n)$ time.*

Unit squares.

To obtain a (2/3)-approximation for this case, we partition the vertical stabbing lines into *three* sets, $\Lambda_i = \{l_j \mid j = i \pmod 3\}$ for $i = 1, 2, 3$. Deleting all squares stabbed by one of the sets Λ_i divides the problem into independent subproblems defined by two consecutive stabbing lines each. A greedy sweep-line algorithm finds the optimal solution for each of these subproblems as follows. Sweep a line from bottom to top stopping at event points s_i and S_i. We maintain an active pair of at most two independent squares that contains at each scale the leftmost square on the left stabbing line and the rightmost square on the right line if they are independent. If they overlap, then whichever was made active first remains active. For a square E_i, set a_i to the scale at which E_i enters the active pair, and set A_i to the scale at which it leaves the active pair.

LEMMA 1. *The above algorithm computes, in $O(n \log n)$ time, the maximum active range height of a set of n unit squares stabbed by two vertical lines of distance at least 1.*

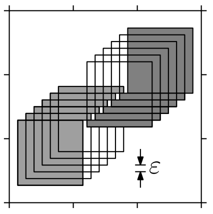


Figure 6: Instance with n squares in a (3×3)-block. In the optimal solution (shaded) each square has an active range.

PROOF. Obviously the algorithm activates the maximum number of independent squares (one or two) at each scale. It remains to show that the active range is set at most once for each square. If a square E_i leaves the active list before the sweep line reaches S_i, it must be replaced by a more extremal (left or right) square E_j. But clearly $S_j > S_i$, so E_i will never again be extremal. It is not hard to see that this can be implemented in $O(n \log n)$ time. $\square$

Using this method, we optimally solve the three subproblems created by removing the squares stabbed by, respectively, one of the sets Λ_i. By the pigeon-hole principle, the subsolution that maximizes H is a (2/3)-approximation. This yields the following theorem.

THEOREM 6. *The maximum total active range height for a set of n unit-square extrusions can be approximated by a factor of $2/3$ in $O(n \log n)$ time.*

Note that the above technique cannot be easily extended to a PTAS by defining sets $\Lambda_1, \dots, \Lambda_k$ for some $k > 3$ since it is not clear how to solve the remaining $(k-1)$-line subproblems (near-) optimally in polynomial time. Unlike the situation in Lemma 1, greedily activating the maximum number of independent squares at each scale for $k > 3$ can result in a square being activated more than once, i.e., its active range is not a contiguous interval. Partitioning the plane into square blocks (as Hochbaum and Maass [7] do) instead of vertical strips does not help either since any optimal solution inside a $(t \times t)$-block may contain active ranges from an arbitrary number of unit squares; see Figure 6. Any k-element subset $\mathcal{E}'$ of the depicted n-square instance $\mathcal{E}$ has $H(\mathcal{E}') \approx 3 + k/n$, whereas $H(\mathcal{E}) = 4 - \Theta(\varepsilon)$.

3.4 Divide and conquer

Arbitrary rectangles.

Agarwal et al. [1] gave an $O(n \log n)$-time divide-and-conquer algorithm to compute a $(1/\log n)$-approximation for MIS among axis-aligned rectangles. Their algorithm readily adapts to ARO. First, the given set $\mathcal{E}$ of rectangles is split at the vertical line $g : x = x_{\text{med}}$, where x_{med} is the median of the at most $2n$ abscissae of the left and right sides of the given rectangles. This yields the three disjoint subsets $\mathcal{E}_1$, $\mathcal{E}_2$, and $\mathcal{E}_{12}$ of the rectangles in $\mathcal{E}$ lying left of g, lying right of g, and intersecting g, respectively. The solutions $\mathcal{A}_1$ and $\mathcal{A}_2$ for $\mathcal{E}_1$ and $\mathcal{E}_2$ are computed recursively. The solution $\mathcal{A}_{12}$ for $\mathcal{E}_{12}$ can be solved optimally by a simple greedy algorithm. Finally, among the two sets $\mathcal{A}_1 \cup \mathcal{A}_2$ and $\mathcal{A}_{12}$, the one with the larger value of the objective function is returned. The proof of the following theorem is similar to that in [1].

THEOREM 7. *The divide-and-conquer algorithm computes in $O(n \log n)$ time a $(1/\log n)$-approximation to the maximum total active range height for a set of n rectangles.*

3.5 Top-to-bottom fill-down sweep

A number of variants of 1d- and 2d-ARO are approximated by a constant factor with Algorithm 1, below. The idea is to sweep down over the extrusions in $\mathcal{E}$, and if $E_i \in \mathcal{E}$ is selected at scale s, we "fill" E_i from s down to its bottom—i.e., we set $[a_i, A_i] = [s_i, s]$. Thus we have $a_i = s_i$ for every E_i that contributes to the objective function H at all.

Say that E_i is *available* if its available range includes the current sweep scale s, and *active* if its active range has already been set and covers s. We are interested in event points at which the conflict graph over the available extrusions changes. This happens at each S_i and s_i, and with some extrusion shapes it also happens at additional scales. If E_i and E_j are both available at s and at $s' > s$, and they intersect at s' but not at s, then let s_{ij} refer to the lowest scale at which they intersect. Let k be the number of s_{ij} events over $\mathcal{E}$. We make use of a subroutine, "try to pick" E_i, which means, "if E_i does not intersect the interior of any extrusion already chosen to be active at the current sweep scale s, then make E_i active and set $[a_i, A_i] = [s_i, s]$".

ALGORITHM 1. *Top-to-bottom sweep algorithm.*

> Sweep a line or plane from top to bottom. At each event point of type S_i, s_i, or s_{ij}, try to pick each available but inactive extrusion E_j, in non-increasing order of S_j.

The following lemma will help prove the approximation factors. Let $\mathcal{A} = \{(a_i, A_i)\}$ be the solution computed by Algorithm 1. Say that E_j *blocks* E_i at scale s under a given solution if E_i and E_j overlap (i.e., their interiors intersect) at s and $s \in [a_j, A_j]$. Note that this implies that $s \notin [a_i, A_i]$. Say that two extrusions are *independent at* s if their restrictions to the horizontal plane at scale s are non-overlapping.

LEMMA 2. *If, for any $E \in \mathcal{E}$ and $s \geq 0$, E can block no more than c pairwise independent extrusions at s, then $\mathcal{A}$ is a $(1/c)$-approximation for the maximum total active range height of $\mathcal{E}$.*

PROOF. Suppose that $E \in \mathcal{E}$ is inactive at scale s under $\mathcal{A}$. Then E must be blocked at the nearest event point above (or at) s since otherwise it would be picked by Algorithm 1. Since the extrusion conflict graph only changes at event points, E is blocked at s. Thus, in $\mathcal{A}$, if E is inactive at any scale s then E is blocked at s.

If at any scale no extrusion can block more than c mutually independent extrusions, and in $\mathcal{A}$ every inactive extrusion is blocked, then at any scale the number of active extrusions in an optimal solution can be no more than c times the number in $\mathcal{A}$. Integrating over all scales proves the lemma. $\square$

For each of the extrusion shapes covered in this section we determine a value for c, usually 2 or 4. For example, it is easy to see that $c = 2$ for unit-width rectangles (or, more generally, for any set of rectangles where the x-order of the left edges is the same as the x-order of the right edges), so Algorithm 1 yields a (1/2)-approximation. It runs in $O(n \log n)$ time. (Compare Theorem 5.)

Congruent trapezoids.

The top-to-bottom nature of Algorithm 1 ensures that if E_j blocks E_i at scale s then E_i intersects at least one side edge of E_j at s. This implies $c = 2$ in Lemma 2.

THEOREM 8. *Algorithm 1 computes a $(1/2)$-approximation for the maximum total active range height of a set of n congruent trapezoids in $O(k \log n + n \log^2 n)$ time, where k is the number of pairs of intersecting trapezoids.*

PROOF. We first show that with congruent trapezoids, if E_j blocks E_i at scale s under solution $\mathcal{A}$, then E_i must intersect at least one side edge of E_j at scale s. This implies $c = 2$ in Lemma 2, and thus the $1/2$ approximation factor.

Suppose E_j blocks E_i at s. If $S_i \leq S_j$ then E_i is at least as wide as E_j at s, so it must intersect a side edge of E_j. If $S_i > S_j$ then E_i is also available at scale A_j, when Algorithm 1 selects E_j, and since the trapezoids are wider at higher scales, E_i and E_j also intersect at A_j. But E_i must also be blocked by another trapezoid at A_j since otherwise Algorithm 1 would choose it over E_j. Thus, E_i intersects a side edge of E_j at A_j, and since they are congruent, it must also at s.

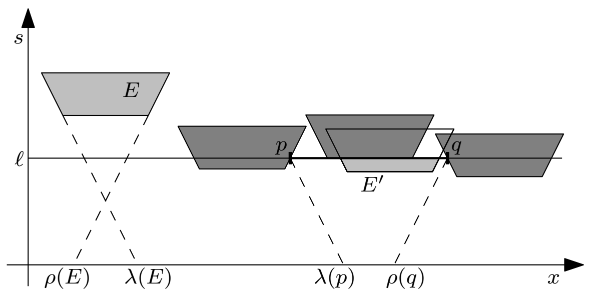
It remains to justify the time complexity. Initially we use a simple plane-sweep algorithm to find the s_{ij} events in $O((k+n)\log n)$ time. Then we create the sorted event list with s_i, S_i, and s_{ij} events. If multiple events occur at the same scale, then say that the s_i events come first, then the s_{ij} events, and finally the S_i events.

We use the fact that all trapezoids are congruent to transform them into a dual space. For a trapezoid E denote the abscissae of the intersections of the x-axis and the downward extensions of the left and right edges of E by $\lambda(E)$ and $\rho(E)$, respectively. (Notice that all trapezoids lie above the x-axis.) This defines a point $\delta(E) = (\lambda(E), \rho(E))$ in the dual space. Similarly, an arbitrary point p above the x-axis is mapped to a point $\delta(p)$ in the dual space using lines with the same slopes as the left and right edges of the trapezoids. See Fig. 7a. Most importantly, note that we can activate the trapezoid E' in the free space defined by the line segment $[p, q]$ if $\lambda(E') \geq \lambda(p)$ and $\rho(E') \leq \rho(q)$.

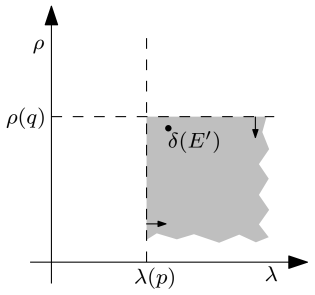
We use a balanced binary search tree for the active list, where the active trapezoids are ordered left to right by their segment intersections with the sweep line. For the available list, we use a 2-dimensional "dynamic priority range tree", which stores $\delta(E_i)$ for each available but inactive trapezoid E_i. A query on this structure asks for the point with maximum S_i among those in a rectangular region of (λ, ρ) space. Specifically, the query region $[\lambda(p), \infty) \times (-\infty, \rho(q)]$ returns the highest trapezoid that can be made active in the free space defined by line segment $[p, q]$.

We modify the usual plane sweep algorithm as follows: if multiple s_i events, or multiple s_{ij} events, share the same scale, then we consider the group of events together.

For one or more s_i events at the current sweep scale, remove each E_i from the active list, maintaining a new list of $[p, q]$ intervals freed by each. If any free interval contains a previously found interval, retain only the larger one. Then we traverse the interval list until it is empty. For each $[p, q]$ on the interval list, query the available list for the topmost trapezoid E_j that can be made active in $[p, q]$. If an E_j is found, add it to the active list. This splits $[p, q]$ into two parts $[p, p']$ and $[q', q]$ to the left and right of E_j. Append $[p, p']$ and $[q', q]$ to the interval list.



(a) The (λ, ρ)-coordinates of a trapezoid E. In the free interval $[p, q]$ on ℓ the trapezoid E' can be activated.



(b) Query region for finding $\delta(E')$ in (λ, ρ)-space.

Figure 7: Transformation to (λ, ρ)-coordinates.

For each of one or more s_{ij} events at the current sweep scale, if either of E_i or E_j is active then add the inactive trapezoid to a candidate list $\mathcal{C}$ sorted non-increasingly by S_i. Once the group of s_{ij} events is exhausted, try to pick each extrusion in $\mathcal{C}$ in order. For each S_i event, try to pick E_i.

Creating the event list requires $O((k+n)\log n)$ time. The s_i events require $O(n \log^2 n)$ time in total since there will be $O(1)$ searches on the available list for each trapezoid. Events of type s_{ij} require $O(k \log n)$ time in total for k searches of the active list. Events of type S_i require $O(n \log n)$ time in total for n searches of the active list. Thus the time complexity of the whole algorithm is $O(k \log n + n \log^2 n)$. ☐

Congruent frusta.

With congruent frusta, a blocked frustum must intersect a side *face* of its blocker. The number of independent frusta that can intersect a single face depends on W, the ratio of the side length of the (square) top side of each frustum to that of the bottom.

THEOREM 9. *Algorithm 1 computes a $1/(4W)$-approximation for the maximum total active range height of a set of n congruent frusta in $O(n^4)$ time.*

PROOF. We first show that with congruent frusta, if E_j blocks E_i at scale s under solution $\mathcal{A}$, then E_i must intersect at least one side face of E_j at scale s. Suppose E_j blocks E_i at s. If $S_i \leq S_j$ then E_i is at least as large as E_j at s, so it must intersect a side face of E_j. If $S_i > S_j$ then E_i is also available at scale A_j, when Algorithm 1 selects E_j, and since the frusta are larger at higher scales, E_i and E_j also intersect at A_j. But E_i must also be blocked by another frustum at A_j since otherwise Algorithm 1 would choose it over E_j. Thus, E_i intersects a side face of E_j at A_j, and since they are congruent, it must also at s.

This implies that $E \in \mathcal{E}$ can block at most $4W$ independent extrusions at any scale s, and the $1/4W$ approximation factor follows from Lemma 2. (See Fig. 9 for an example with $W = 3$.)

At each of $O(n^2)$ event points we try to pick each of $O(n)$ extrusions, and the try-to-pick routine traverses $O(n)$ active extrusions, giving $O(n^4)$ time. ☐

Trapezoidal segments of congruent triangles.

Here we consider as extrusions a set of trapezoidal segments of congruent underlying triangles with their apexes on a horizontal line.

THEOREM 10. *Given a set of n trapezoidal segments of congruent triangles, Algorithm 1 computes a $(1/2)$-approximation for the maximum total active range height in $O((n+k)\log n)$ time.*

PROOF. At each scale the width of every trapezoid is the same, which implies that any trapezoid blocked by $E \in \mathcal{E}$ intersects a side edge of E. Thus, at most two extrusions blocked by E at s can be independent at s, and the approximation factor 1/2 follows from Lemma 2.

We have a natural order on the extrusions given by the x-value of the apexes of the underlying cones. We can easily determine the interval on the x-axis containing cones that are within a given free interval at the current sweep scale.

We use a priority search tree for the available list, with S_i values as the priority. The active list will be a balanced binary search tree. The event heap is initialized with all S_i and s_i events in $O(n \log n)$ time.

At event points of type S_i we try to pick E_i, and search the available list for events q_{ij}, which are then added to the event list. In total, the S_i events require $O(n \log n)$ time for searching and updating the active list, $O(k + n \log n)$ time for searching the available list, and $O(k \log n)$ time for adding the q_{ij} events.

At event points of type s_i we remove E_i from the available list. If E_i was active, then remove it from the active list and compute the free interval determined by its predecessor and successor. Query the available list for the extrusion with the highest S_i in this free interval. If one is found, then also query the two new intervals on either side of it. At most two extrusions will be added to the active list. The s_i events require $O(n \log n)$ time in total.

At event points of type q_{ij}, if E_i or E_j is active then try to pick the other. These events total $O(k \log n)$ time.

In summary, the time complexity is $O((k + n)\log n)$. ☐

Frustal segments of congruent square cones.

This is the 2d-equivalent of the trapezoidal segments of congruent triangles in the previous paragraph.

THEOREM 11. *Given a set of n frustal segments of axis-aligned unit square cones, Algorithm 1 computes a $(1/4)$-approximation for the maximum total active range height in $O((n + k)\log^2 n)$ time.*

PROOF. Any extrusion blocked by an extrusion E at scale s must intersect one of the four corner edges of E at s, so at most four such extrusions can be independent. Thus the approximation factor 1/4 follows from Lemma 2.

In the implementation we use a range tree for the active extrusions, a range tree for the available extrusions, and a heap for the event list. The event list initially has all S_i and s_i events. The range trees store the apex locations in the (x, y) plane of the underlying cones. We can easily compute a range in the (x, y) plane containing cones that intersect a given cone at a particular scale.

At event S_i we first try to pick E_i. This requires $O(n \log^2 n)$ time over all S_i events. Next we compute s_{ij} for each E_j in the available list that intersects E_i. This requires $O(k' + \log^2 n)$ time for each E_i, or $O(k + n \log^2 n)$ overall. Each computed s_{ij} is added to the event list, requiring $O(k \log n)$ time in total. Finally, E_i is added to the available list, requiring $O(n \log^2 n)$ time in total.

At event s_i we first remove E_i from the available list, and from the active list if it is active, requiring $O(n \log^2 n)$ time in total. If E_i has been active we search the available list for extrusions that intersect E_i at s_i. This requires $O(k' + \log^2 n)$, or $O(k + n \log^2 n)$ time overall. We can sort the extrusions found in $O(k' \log k')$ time, or $O(k \log n)$ overall, and then try to pick each one in order of their heights, which will require $O(k \log^2 n)$ time overall.

At event s_{ij}, if one of E_i and E_j is active, then try to pick the other. This requires $O(k \log^2 n)$ time overall.

Altogether, the time complexity is $O((n + k) \log^2 n)$. □

Simple ARO with congruent square cones is a special case of the above, with each $[s_i, S_i] = [0, S_{\max}]$, so we immediately get the following corollary.

COROLLARY 1. *Given a set of n congruent square cones, Algorithm 1 computes a* $(1/4)$*-approximation for the maximum total active range height in* $O((n + k) \log^2 n)$ *time.*

3.6 Level-based small-to-large greedy algorithm

In this section we give an algorithm for simple 2d-ARO with square cones. It computes a $(1/8)$-approximation when the cones are congruent, and a $(1/24)$-approximation otherwise. The algorithm intersects the given cones with $O(\log n)$ horizontal planes, starting at $S_{\max}$ and proceeding downward.

ALGORITHM 2. *Level-based algorithm for 3d-cones*

> Initially no extrusion is active. In phase i, $i = 0, \dots, \lceil \log n \rceil$, let π_i be the horizontal plane at scale $s = S_{\max}/2^i$. Let E_j^i be the intersection of extrusion E_j with π_i and call E_j^i *active* if E_j is already active. As long as there is an inactive object E_j^i that does not intersect any active object, choose the smallest such object $E_{j^\star}^i$ and make $E_{j^\star}$ (and $E_{j^\star}^i$) active by setting $A_{j^\star} = s$.

We first consider arbitrary square cones that are symmetric to the vertical axes passing through their apexes. When the algorithm terminates, all squares at level i that are not active must intersect an active square—they are *blocked.* We associate each blocked square E_j^i to one of the active squares in the following way: (i) If E_j^i was not blocked at the beginning of phase i but became blocked by a newly activated square E_k^i, then associate E_j^i to E_k^i. (ii) If E_j^i was blocked in the beginning of phase i then associate E_j^i to any of its blocking squares that were active at the beginning of phase i. Next, we show that the squares associated to an active square cannot be arbitrarily small.

LEMMA 3. *Let* E_j^i *be an active square at level i with side length* ℓ_j^i*. Then any square associated to* E_j^i *has side length at least* $\ell_j^i/3$ *and intersects the boundary of* E_j^i*.*

PROOF. Let E_k^i be associated to E_j^i with $\ell_k^i < \ell_j^i$. By the greedy choice of the algorithm, all squares associated to a newly active square are larger than it. This implies that E_j must have been activated at a higher level, and that E_k must have been reassigned to E_j at some level $h \le i$. Thus, at level $h-1$ square E_k^{h-1} was associated to another square E_l^{h-1}. Note that for this reassignment to take place at level h, E_j^{h-1} must have been active. Thus we know that E_j^{h-1} and E_l^{h-1} do not intersect, but they both intersect E_k^{h-1}; see Fig. 8a. At level h the reassignment takes place because E_k^h no longer intersects E_l^h but still intersects E_j^h; see Fig. 8b. Now suppose $\ell_k^h < \ell_j^h/3$. Then by going from level h to $h-1$ the side lengths of the squares are doubled and it is easy to verify that E_k^{h-1} would be contained in E_j^{h-1}, a contradiction to the fact that $E_k^{h-1} \cap E_l^{h-1} \ne \emptyset$. As $\ell_k^h \ge \ell_j^h/3$ this also holds for level i, and since E_k^{h-1} intersects the boundary of E_j^{h-1} this is also still true for level i. □

Define $\pi_{\lceil \log n \rceil + 1}$ as the plane $s = 0$. We denote the active segments of the extrusions between planes π_{i-1} and π_i in the optimal solution $\mathcal{S}$ by $\mathcal{S}_i$ and in the solution of our algorithm by $\mathcal{A}_i$, respectively. We charge the active range height $H(\mathcal{S}_i)$ to that of $H(\mathcal{A}_{i+1})$.

LEMMA 4. *For each level* $i \in 1, \dots, \lceil \log n \rceil - 1$ *it holds that* $H(\mathcal{A}_{i+1}) \ge 1/24\ H(\mathcal{S}_i)$*.*

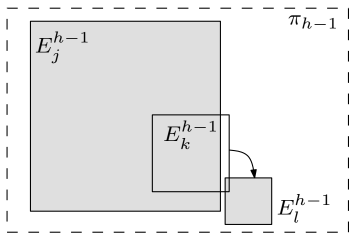
PROOF. Let square E_j^i be active in $\mathcal{A}$ and consider the set $D(E_j^i)$ of squares in π_i associated to it. The squares in $D(E_j^i)$ that correspond to active extrusions in $\mathcal{S}_i$ cannot intersect each other.

By Lemma 3, all squares in $D(E_j^i)$ have side length at least $\ell_j^i/3$ and intersect the boundary of E_j^i. Thus, at most 12 of those squares can be independent in π_i and hence active in $\mathcal{S}_i$ like in Fig. 9. Now the height between levels i and $i-1$ is twice the height between levels $i+1$ and i. Hence the active height of E_j in $\mathcal{A}_{i+1}$ is at least 1/24 times the sum of heights of active extrusions in $\mathcal{S}_i$ whose squares at level i are associated to E_j^i. It follows that $H(\mathcal{A}_{i+1}) \ge 1/24\ H(\mathcal{S}_i)$. □

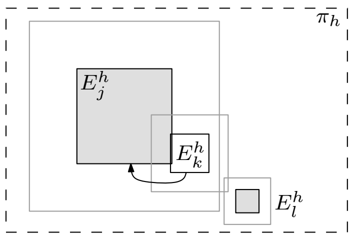
THEOREM 12. *Algorithm 2 computes a* $(1/24)$*-approximation to the maximum total active range height of a set of arbitrary square cones in* $O(n \log^3 n)$ *time.*

PROOF. It remains to compare $H(\mathcal{S}_{\lceil \log n \rceil}) + H(\mathcal{S}_{\lceil \log n \rceil + 1})$ to $H(\mathcal{A}_{\lceil \log n \rceil + 1}) + H(\mathcal{A}_1)$. The height of $\pi_{\lceil \log n \rceil - 1}$ is at most $2S_{\max}/n$ and obviously there are at most n active cone segments in $\mathcal{S}$ below $\pi_{\lceil \log n \rceil - 1}$, so their total active range height is at most $2S_{\max}$. On the other hand, there is at least one active cone segment in $\mathcal{A}_1$ of height $S_{\max}/2$. Together with Lemma 4 this implies the approximation factor 1/24.

For an efficient implementation of Algorithm 2 we store the squares in each level i in a two-dimensional segment tree τ_i, which supports deletion in $O(\log^2 n)$ time [4]. For each square E_j^i that has been activated at a previous level we



(a) Squares at level $h-1$.



(b) Squares at level h.

Figure 8: Intersection behavior of E_j, E_k, E_l at two consecutive levels.

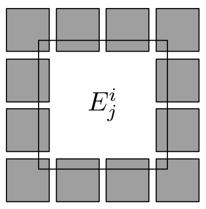


Figure 9: At most 12 pairwise independent squares intersect E_j^i.

delete all intersecting squares as follows. Place vertices at each corner of E_j^i and two vertices on each edge equidistantly. Query τ_i with each of these 12 vertices and delete the returned squares from τ_i. As the side length of intersecting squares is at least $\ell_j^i/3$ (see Lemma 3) these points suffice to find all intersecting squares. From the remaining squares the algorithm iteratively chooses the smallest one. By querying τ_i with the four corner points of the chosen square we identify and remove all intersecting squares, which are larger and thus must contain one of the corner points. Since a deletion takes $O(\log^2 n)$ time, Algorithm 2 needs $O(n \log^2 n)$ time per level.

There are $\log n$ levels and thus the total running time is $O(n \log^3 n)$. $\square$

With congruent square cones, all intersection squares on a plane are the same size, so at most four independent squares can intersect a given square. Thus a similar argument gives the following corollary.

Corollary 2. *Algorithm 2 computes a* $(1/8)$*-approximation to the maximum total active range height of a set of congruent square cones in* $O(n \log^3 n)$ *time.*

4. OPEN PROBLEMS

ARO is an exciting new problem inspired by interactive web-based mapping applications, and this is the first paper with an extensive, rigorous algorithmic study. We have described a number of approximation algorithms; an obvious question is whether any approximation factor can be improved, or whether any of the problems admits a PTAS. (Some obvious attempts for a PTAS do not work, see Section 3.3.) Furthermore, the complexity of general 1d-ARO is still unknown for regular shapes such as unit squares and congruent trapezoids.

Mapping applications in practice often need to work with different label models, such as labels of different lengths and fonts; non-axis-aligned labels; non-rectangular labels, such as a road label that follows a curvy road; and sliding labels—i.e., non-invariant point placements. Any of these raises a number of interesting theoretical questions.

5. 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. Been, E. Daiches, and C. Yap. Dynamic map labeling. *IEEE Trans. Visualization and Computer Graphics*, 12(5):773–780, 2006.

[3] B. Chazelle and 36 co-authors. The computational geometry impact task force report. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, volume 223, pages 407–463. AMS, 1999.

[4] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 3rd edition, 2008.

[5] J. Christensen, J. Marks, and S. Shieber. An empirical study of algorithms for point-feature label placement. *ACM Transactions on Graphics*, 14(3):203–232, 1995.

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

[7] D. S. Hochbaum and W. Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *J. ACM*, 32:130–136, 1985.

[8] D. E. Knuth and A. Raghunathan. The problem of compatible representatives. *SIAM J. Discr. Math.*, 5(3):422–427, 1992.

[9] D. Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.*, 11(2):329–343, 1982.

[10] 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.

[11] I. Petzold, L. Plümer, and M. Heber. Label placement for dynamically generated screen maps. In *Proc. 19th Internat. Cartographic Conf. (ICC'99)*, pages 893–903. Ottawa, Canada, 1999.

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

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

[14] A. Wolff and T. Strijk. The map-labeling bibliography. i11www.ira.uka.de/map-labeling/bibliography, 1996.