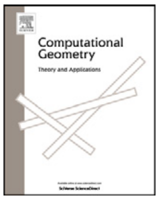
Computational Geometry ••• (••••) •••–•••

Contents lists available at SciVerse ScienceDirect

Computational Geometry: Theory and Applications

www.elsevier.com/locate/comgeo



Approximation algorithms for free-label maximization

Mark de Berg, Dirk H.P. Gerrits [*]

Department of Mathematics and Computing Science, TU Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

A R T I C L E   I N F O

Article history:
Received 27 April 2011
Accepted 19 October 2011
Available online xxxx
Communicated by D. Wagner

Keywords:
Map labeling
Moving objects
Free-label maximization

A B S T R A C T

Inspired by air-traffic control and other applications where moving objects have to be labeled, we consider the following (static) point-labeling problem: given a set P of n points in the plane and labels that are unit squares, place a label with each point in P in such a way that the number of free labels (labels not intersecting any other label) is maximized. We develop efficient constant-factor approximation algorithms for this problem, as well as PTASs, for various label-placement models.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

Air-traffic controllers have the important job of monitoring airplanes and warning pilots to change course to avoid any potential collision. They do this using computer screens that show each airplane as a moving point with an associated textual label. The labels hold important information (such as altitude and velocity) that needs to remain readable. As the airplanes move, however, labels may start to intersect. Currently, this means air-traffic controllers spend a lot of their time moving labels around by hand. So what we are interested in is to develop algorithms to automate this process.

Label models. A good labeling for a point set has legible labels, and an unambiguous association between the labels and the points. The latter puts restrictions on the shape of labels and the way they can be placed in relation to points. Various *label models* have been proposed, most often with labels assumed to be axis-aligned rectangles slightly larger than the text they contain.

In the *fixed-position models*, every point has a finite number of *label candidates* (often 4 or 8), each being a rectangle having the point on its boundary. In particular, in the 1-position model one designated corner of the label must coincide with the point. In the 2-position models there is a choice between two adjacent corners, and the 4-position model allows any corner of the label to coincide with the point. These models are illustrated in the upper-left 2×2 block in Fig. 1, where we use the following notation: 1P denotes the 1-position model, 2PH and 2PV denote 2-position models (where the H and V indicate whether the two designated corners are endpoints of the same horizontal or vertical edge), and 4P denotes the 4-position model.

The *slider models* introduced by Van Kreveld et al. [15] generalize the fixed-position models. In the 1-slider (1SH, 1SV) models one side of the label is designated, but the label may contain the point anywhere on this side. In the 2-slider (2SH, 2SV) models there is a choice between two opposite sides of the label, and in the 4-slider (4S) model the label can have the point anywhere on its boundary (see the last row and column in Fig. 1). Erlebach et al. [9] introduced terminology analogous to the slider models for fixed-position models with a non-constant number of positions (1MH, 1MV, 2MH, 2MV, 4M (not pictured)).

[*] Corresponding author.
E-mail addresses: mdberg@win.tue.nl (M. de Berg), dirk@dirkgerrits.com (D.H.P. Gerrits).

0925-7721/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.comgeo.2011.10.004

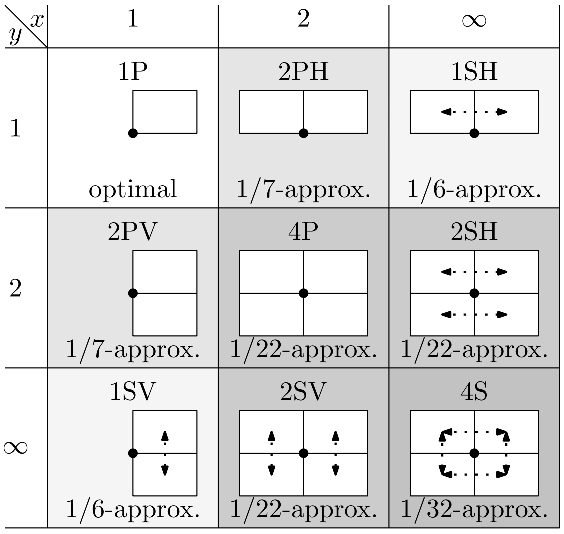


Fig. 1. The fixed-position and slider models, and our constant-factor approximations for the free-label-maximization problem (assuming unit-square labels) for each. The x-axis (y-axis) indicates the number of allowed horizontal (vertical) positions for a label.

Previous work. A lot of research has gone into labeling static points (as well as polylines and polygons) on cartographic maps. See, for instance, the on-line Map Labeling Bibliography [19], which currently contains 371 references. This research has focused mostly on two optimization problems. The *size-maximization problem* asks for a labeling of all the points with labels of the largest possible size under the condition that no two labels intersect. Here (and in the sequel) the labels are considered open sets, that is, two labels intersect if and only if their interiors intersect. The size-maximization problem is APX-hard (except in the 1P model), even for unit-square labels [10]. Constant-factor approximation algorithms exist for various label models [10,14]. The more widely studied *number-maximization problem* asks for a maximum-cardinality *subset* of the n points to be labeled with pairwise non-intersecting labels of *given* dimensions. Even if all labels are unit squares, this problem is known to be strongly NP-hard for the 1P [11], 4P [10,16], and 4S models [15]. A generalization of this problem concerns weighted points [17] and asks for a maximum-weight subset of the points to be labeled so that, for example, a large city is more likely to be labeled than a small town. For unit-height rectangular labels this problem admits a polynomial-time approximation scheme (PTAS) for static points in all fixed-position and slider models, both in the unweighted [3,15] and the weighted case [9,17]. For arbitrary rectangles in the unweighted case an $O(1/\log\log n)$-approximation algorithm is known for the fixed-position models [2], but the slider models, the weighted case, and the (non-)existence of a PTAS remain open problems.

Despite the large body of work on labeling static points, virtually no results have been published on labeling moving points. Been et al. [1] studied the unweighted number-maximization problem for static points under continuous zooming in and out by the viewer, which can be seen as points moving on a very specific kind of trajectories. Gemsa et al. [12] similarly studied static points under continuous rotation of the view. Rostamabadi and Ghodsi [18] studied how to quickly flip and scale the labels of static points to avoid one moving point.

Free-label maximization. As just discussed, previous work has focused on the size-maximization and number-maximization versions of the label-placement problem. By either shrinking the labels, or removing some of them, a labeling is produced without any intersections. However, European air-traffic safety regulations require all airplanes to be labeled at all times, with labels of fixed sizes [7]. Thus we must allow label intersections, and naturally want as few of them as possible. Or rather, we would like as many *free* labels—labels that are disjoint from all other labels—as possible. We call this problem the *free-label-maximization problem*. The free-label-maximization problem has not been studied previously. Hence, as a first step towards the automatic labeling of moving points in air-traffic control, we investigate the free-label-maximization problem for static points.

Our results. The decision problem of determining whether a labeling without intersections exists for a static point set is strongly NP-complete [10,16], even if all labels are unit squares. This immediately implies that finding a labeling with the maximum number of free labels is NP-hard. Hence, we have to settle for approximation algorithms.

For unit-square labels we describe a simple plane-sweep algorithm that runs in $O(n\log n)$ time and $O(n)$ space. (In fact, our algorithm works if all labels are translates of a fixed rectangle, since a suitable scaling can transform this case to the case of unit-square labels.) The algorithm gives a constant-factor approximation for the 2PH and 1SH models; by running the algorithm multiple time, each time with a different sweeping direction, we obtain constant-factor approximations for the other label models. Fig. 1 shows the approximation ratios we obtain for the various models.[1] We also present a (more involved) PTAS. The existence of a PTAS makes free-label maximization easier than size maximization, as the latter is APX-hard even for unit-square labels. In contrast, techniques used for (approximate) number maximization for unit-square labels easily extend to unit-height labels of differing widths, which does not seem to be the case for free-label maximization. Thus the complexity of free-label maximization seems to fall in between that of the size-maximization and number-maximization problems.

We present our constant-factor approximation algorithm in Section 2, and our PTAS in Section 3. The former's approximation guarantees for the various label models are listed in Fig. 1. We will only discuss the 2PH, 4P, 1SH, 2SV, and 4S label models; the algorithms and proofs for the other models are analogous. Throughout the paper we assume that no two points have the same x- or y-coordinate. This assumption is not essential, but it makes our exposition simpler.

2. Constant-factor approximations for unit squares

Let P be a set of n points in the plane. We wish to label each point in P with a label that is a unit square in such a way that the number of free labels is maximized. Next we describe a generic algorithm for this problem. It can be applied to any label model, but the approximation factor and implementation details depend on the model used.

The algorithm, which we call GreedySweep, processes the points from left to right, labeling them one-by-one. Whenever possible it produces a "freeable" label. We call a label or label candidate L *freeable* if it is free *at the time* and we know it will *remain* free as we continue labeling the remaining points. More formally, suppose GreedySweep has labeled points $p_1, \dots, p_{i-1} \in P$ so far, and has now arrived at point $p_i \in P$. We denote the set of all label candidates for $p_i \in P$ according to the label model being used by $\textsc{Cand}(p_i)$. A label candidate $L \in \textsc{Cand}(p_i)$ is freeable if none of the previously placed labels intersect L, and every point still to be labeled has at least one label candidate that intersects neither L nor any previously placed freeable label. A label is freeable simply if it was the result of picking a freeable label candidate as that point's label. We denote by $\textsc{gs}(p_i)$ the label assigned to point $p_i \in P$ by the algorithm.

```
Algorithm GreedySweep($P$, Cand)
1. Let $p_1, \dots, p_n$ be the points in $P$, sorted in order of increasing $x$-coordinate.
2. for $i \leftarrow 1$ to $n$
3.     do if $p_i$ has a freeable label candidate
4.          then Let $L \in \textsc{Cand}(p_i)$ be a leftmost freeable label candidate, with ties broken arbitrarily, and set $\textsc{gs}(p_i) \leftarrow L$.
5.          else Let $L \in \textsc{Cand}(p_i)$ be a leftmost label candidate (with ties broken arbitrarily) that does not intersect any
               freeable labels that have already been placed. Note that such an $L$ always exists, by definition of freeable.
               Set $\textsc{gs}(p_i) \leftarrow L$.
```

Later we will show how to implement the algorithm in an efficient manner. First, however, we analyze its approximation ratio for the 2PH label model and for the 1SH label model.

2.1. Approximation ratio for the 2PH and 1SH models

Lemma 1. *For the free-label-maximization problem with unit-square labels, the* GreedySweep *algorithm gives a* $1/6$*-approximation for the 1SH model and a* $1/7$*-approximation for the 2PH model, and both ratios are tight.*

Proof. Consider an optimal solution (for the given label model) and let $\textsc{opt}(p_i)$ denote the label assigned to point p_i in the optimal solution. Now suppose $\textsc{opt}(p_i)$ is free in the optimal solution. We will charge $\textsc{opt}(p_i)$ to a freeable label $\textsc{gs}(p_j)$ placed by GreedySweep, with $j \leqslant i$. Freeable labels remain free throughout the execution of GreedySweep, so this charges each free label in the optimal solution to a free label in the computed solution. We will first describe how the charging is done, and then argue that no label in the computed solution is charged more than a few times.

- Suppose $\textsc{opt}(p_i)$ is a freeable label candidate at the time p_i is being processed. Then $\textsc{gs}(p_i)$ is necessarily a freeable label—although possibly distinct from $\textsc{opt}(p_i)$—and we charge $\textsc{opt}(p_i)$ to $\textsc{gs}(p_i)$. We call $\textsc{opt}(p_i)$ a *type-*0 *charge* to $\textsc{gs}(p_i)$—see Fig. 2(a).
- Suppose $\textsc{opt}(p_i)$ is *not* a freeable label candidate at the time p_i is being processed. We will now argue that there must still be a "nearby" freeable label to which we can charge.
  Consider an unprocessed point p_k—that is, a point p_k with $k > i$—and suppose we label it with its rightmost label candidate. Then this label cannot intersect $\textsc{opt}(p_i)$. Otherwise all other label candidates for p_k—which lie farther to the left—would intersect $\textsc{opt}(p_i)$ as well, contradicting that $\textsc{opt}(p_i)$ is free. In the same way, p_k's rightmost label candidate cannot intersect previously placed freeable labels. Hence the points still to be labeled cannot be the cause that $\textsc{opt}(p_i)$ is not freeable. Thus there must be a point p_j with $j < i$ which already has a label $\textsc{gs}(p_j)$ that intersects $\textsc{opt}(p_i)$.

[1] In an earlier version [6] we erroneously claimed slightly better bounds.

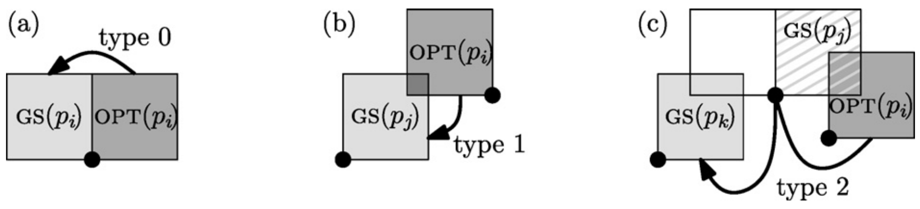


Fig. 2. The three types of charges used in proving the approximation ratio of GreedySweep for the 2PH and 1SH models.

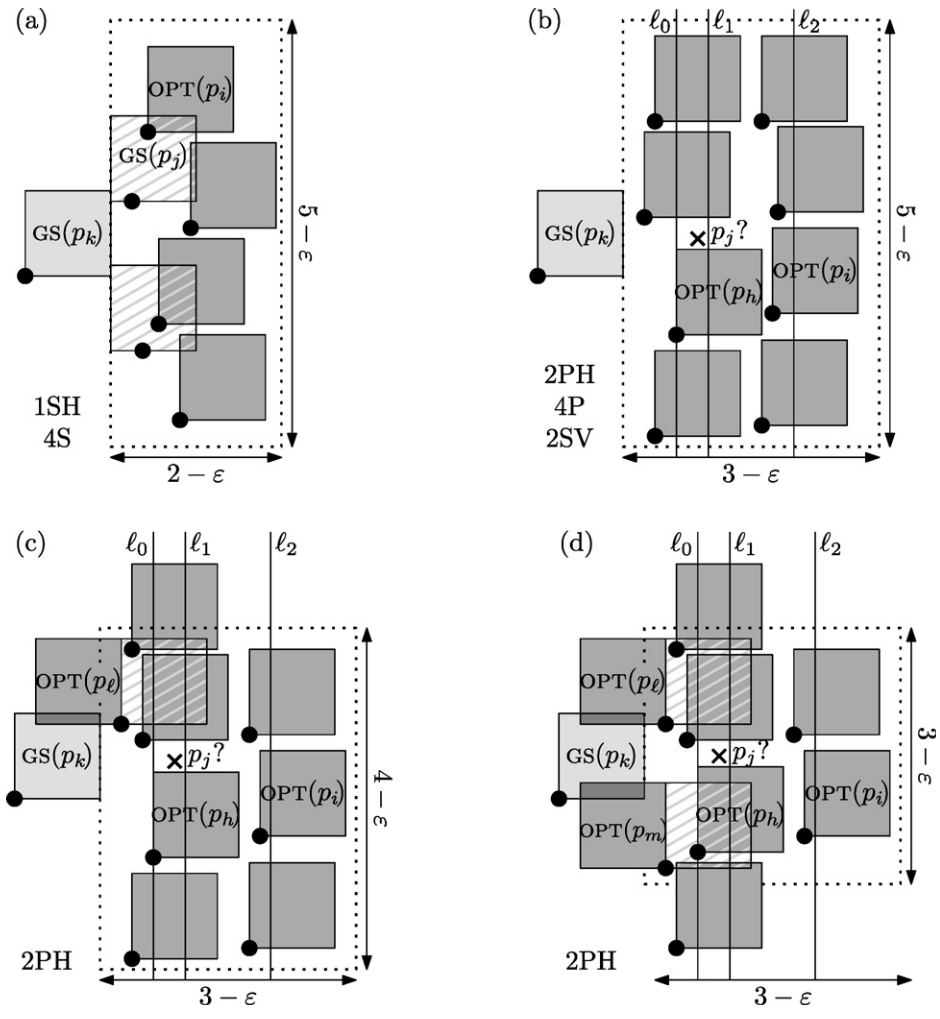


Fig. 3. (a) In the 1SH model there can be four type-2 charges to a label. (b), (c), (d) When there are zero, one, or two type-1 charges, respectively, the total number of type-1 and type-2 charges is always at most seven.

– Suppose GS(p_j) is freeable. Then we charge OPT(p_i) to GS(p_j) and call OPT(p_i) a *type*-1 *charge* to GS(p_j)—see Fig. 2(b).
– Suppose GS(p_j) is *not* freeable. Note that GS(p_j) cannot be the leftmost label candidate for p_j, otherwise all of p_j's label candidates intersect OPT(p_i). But if GS(p_j) is not leftmost, then as we move GS(p_j) to the left it must hit a freeable label before becoming leftmost. Let GS(p_k) be the first freeable label hit, where $k < j$. Charge OPT(p_i) to GS(p_k), and call OPT(p_i) a *type*-2 *charge* to GS(p_k) through p_j—see Fig. 2(c).

To finish the proof of the stated approximation ratios, we will now give tight upper bounds for the number of charges of each type to a freeable label.

• *Type* 0. Consider a label GS(p_i) that receives a type-0 charge OPT(p_i). Trivially, this must be the only type-0 charge to GS(p_i). Note that GS(p_i) must be at least as far to the left as OPT(p_i), otherwise GreedySweep would have picked

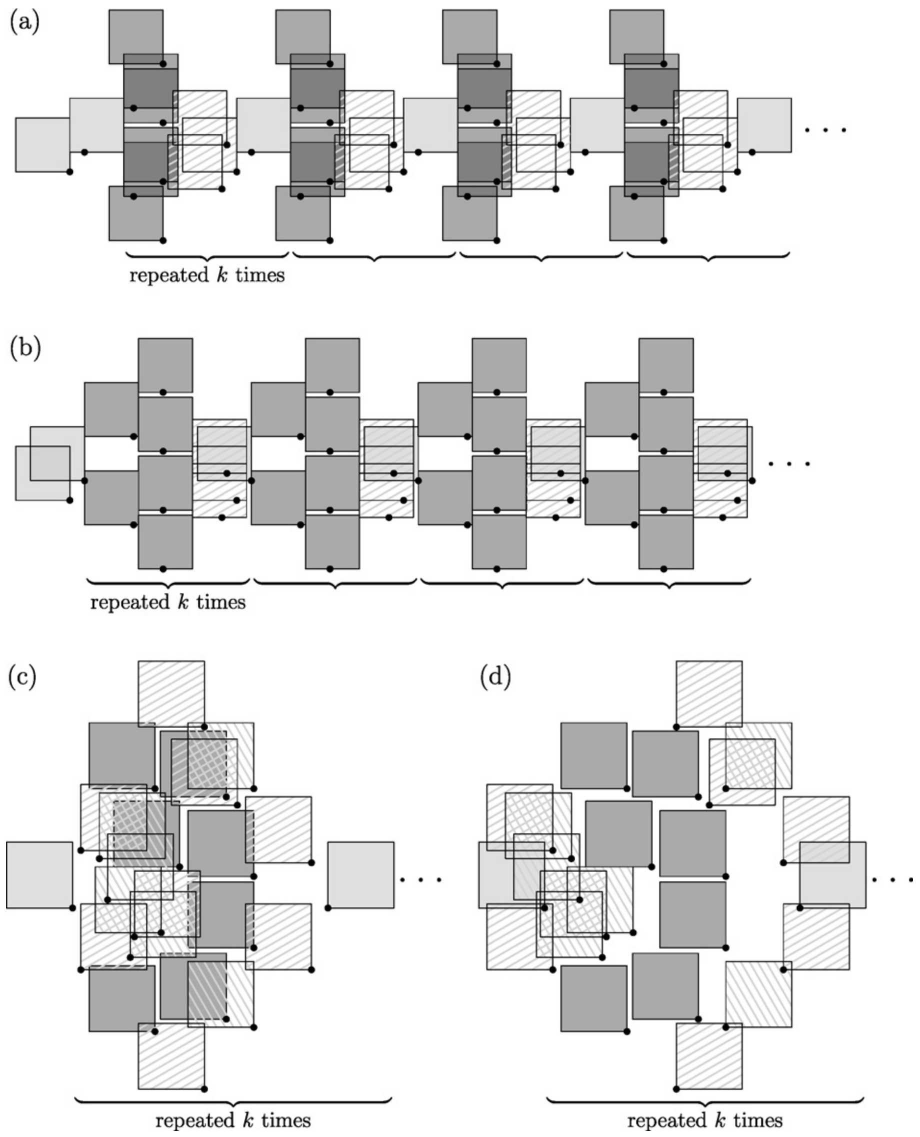


Fig. 4. (a) A point set with its GreedySweep labeling for the 1SH model; the $k+2$ light gray labels are free. (b) A different solution for the same instance where the $6k$ dark gray labels are free. (c) Another point set with its GreedySweep labeling for the 2PH model; the $k+1$ light gray labels are free. (d) A different solution for the same instance where the $7k$ dark gray labels are free.

opt(p_i) over gs(p_i). Thus any label for a point p_j to the right of p_i that intersects gs(p_i) must also intersect opt(p_i), contradicting that opt(p_i) is free. This implies that opt(p_i) is not just the only type-0 charge to gs(p_i), but the only charge of *any* type.

- *Type* 1. Consider a label gs(p_j) that receives a type-1 charge opt(p_i). Note that opt(p_i) must contain a corner of gs(p_j). If opt(p_i) contains the top-left or bottom-left corner of gs(p_j) then all of p_i's label candidates intersect gs(p_j). Thus opt(p_i) must contain the top-right or bottom-right corner of gs(p_j). Hence there can be at most two type-1 charges to gs(p_j).
- *Type* 2. Consider a label gs(p_k) that receives a type-2 charge opt(p_i) through some point p_j. Note that gs(p_j) will have been placed as far to the left as possible without intersecting freeable labels. Hence, in the 1SH model, gs(p_j) touches gs(p_k). In turn, opt(p_i) intersects gs(p_j). Therefore opt(p_i) must be fully contained in a rectangle of size $(2-\varepsilon) \times (5-\varepsilon)$, for some $\varepsilon > 0$, whose left side contains the right side of gs(p_k). In Fig. 3(a) this rectangle is shown dotted. The optimal solution can have at most four free labels completely contained in this rectangle, so gs(p_k) can be charged at most four times in this manner. Together with two type-1 charges, there are at most six charges to gs(p_k) in the 1SH model.

  In the 2PH model, gs(p_j) in general does not touch gs(p_k). Therefore the rectangle fully containing opt(p_i) is of size $(3-\varepsilon) \times (5-\varepsilon)$, as seen in Fig. 3(b). This would imply that gs(p_k) can receive eight type-2 charges, giving a total of ten together with two type-1 charges. By examining the cases with zero, one, and two type-1 charges separately, we shall reduce this total to eight. Then, by analyzing the type-2 charges more carefully, we reduce this further to a bound of seven, which is tight in the worst case.

If there are no type-1 charges to $\text{GS}(p_k)$, then obviously it receives at most eight charges in total, all of type 2. So suppose there is one type-1 charge to $\text{GS}(p_k)$, say $\text{OPT}(p_\ell)$, for some $\ell > k$. Then $\text{OPT}(p_\ell)$ must be leftmost and $\text{GS}(p_\ell)$ must be rightmost. There can be up to two type-2 charges to $\text{GS}(p_k)$ through p_ℓ: one intersecting the top-right corner of $\text{GS}(p_\ell)$ and one intersecting its bottom-right corner. All other type-2 charges must go through points distinct from p_ℓ. Say p_m is such a point. Then $\text{OPT}(p_m)$ is leftmost and intersects $\text{GS}(p_k)$ but not $\text{OPT}(p_\ell)$. If $\text{OPT}(p_\ell)$ contains the top-right corner of $\text{GS}(p_k)$—the case where it contains the bottom-right corner is symmetric—, then $\text{OPT}(p_m)$ must be below $\text{OPT}(p_\ell)$. Further, $\text{GS}(p_m)$ has the same y-coordinate as $\text{OPT}(p_m)$, so all labels with a type-2 charge through p_m must be below the horizontal line through the top edge of $\text{OPT}(p_\ell)$. All type-2 charges to $\text{GS}(p_k)$ must therefore fit in a smaller $(3-\varepsilon)\times(4-\varepsilon)$ rectangle, except possibly the topmost type-2 charge through p_ℓ—see Fig. 3(c). At most six type-2 charges will fit in the rectangle, yielding a total of seven type-2 charges. Together with the type-1 charge $\text{OPT}(p_\ell)$ that gives a total of eight charges in this case. Lastly, suppose there are two type-1 charges to $\text{GS}(p_k)$. Then we combine the arguments for the case where there is a single type-1 charge intersecting the top-right corner of $\text{GS}(p_k)$ and the case where there is a single type-1 charge intersecting the bottom-right corner of $\text{GS}(p_k)$. This gives that all type-2 charges must fit in a $(3-\varepsilon)\times(3-\varepsilon)$ rectangle, with the possible exception of two type-2 charges—see Fig. 3(d). At most four type-2 charges will fit in the rectangle, for a total of six type-2 charges. Together with the two type-1 charges this again gives a total of eight charges.

To reduce the total to seven, we first stab the dotted rectangles in Fig. 3(b)–(d) with two vertical lines ℓ_1 and ℓ_2, with ℓ_1 at distance 1 from the left edge, and ℓ_2 at distance 1 from the right edge. Among the type-2 charges within the rectangle we call the ones intersecting ℓ_1 the *left column*, and the ones intersecting ℓ_2 the *right column*. Note that every type-2 charge in the rectangle must be in one of the columns, and no type-2 charge can be in both columns if we are to have eight charges in total. Let $\text{OPT}(p_h)$ be the rightmost label in the left column, and note that there must be some label in the right column, say $\text{OPT}(p_i)$, that lies completely to the right of $\text{OPT}(p_h)$. For this label $\text{OPT}(p_i)$ to be charged to $\text{GS}(p_k)$ there must be some point p_j with a leftmost label candidate intersecting $\text{GS}(p_k)$ and a rightmost label candidate intersecting $\text{OPT}(p_i)$. This means that the horizontal distances between $\text{GS}(p_k)$ and p_j, and between p_j and $\text{OPT}(p_i)$ must be less than 1. Thus p_j lies between the vertical line ℓ_1 and the vertical line ℓ_0 through the left edge of $\text{OPT}(p_h)$. Furthermore, p_j can be neither above the top label in the left column, nor below the bottom label in the left column, as the vertical distance to $\text{GS}(p_k)$ is then greater than 1. Obviously p_j cannot be inside a free label, so it must be between two labels of the left column. However, the vertical distance between two such labels is necessarily less than 1, so one of them must be intersected by p_j's label. This contradicts that these labels were free in OPT. Hence having eight charges is impossible, and there can be at most seven.

We conclude that the approximation ratios for the 1SH model and the 2PH model are at best 1/6 and 1/7, respectively. Fig. 4 shows that these ratios are tight. □

2.2. Approximation ratio for the 4P, 2SV, and 4S models

Already for the 4P model, GreedySweep can be as bad as an $O(1/\sqrt{n})$-approximation. To see this, first divide the plane into identical, quadrilateral "tiles" by two sets of parallel lines. We "paint" each tile with the same pattern of one black and four white points. The shape of the tiles and the position of the points shown in Fig. 5(a) is such that all label candidates contain points except the bottom-left label candidate for each black point. Thus no white point can have a free label. However, every white point can be labeled to avoid the bottom-left label candidates for the black points, so those can indeed all be free in a solution. For some white points this requires using one of its two rightmost label candidates, however. This is key, as GreedySweep prefers leftmost label candidates, all other things being equal.

When we tile not the plane, but some finite area $\mathcal{A}$ in the above way, it becomes possible to assign free labels to some of the leftmost white points in each "row". GreedySweep will do so, thereby preventing the leftmost black point in the row from getting a free label—see Fig. 5(b). Its non-free label will have to be in a rightmost position, thereby blocking the next black point in the row from getting a free label. With no free labels "nearby", GreedySweep has no reason to prefer one non-free label candidate over another for the white points. It will then always pick one of the leftmost ones, say the top-left one, which blocks all remaining black points from getting free labels. The end result is that GreedySweep only has free labels along the perimeter of $\mathcal{A}$ while an optimal solution has a free label at every black point in the interior of $\mathcal{A}$.

We can remedy the situation by running GreedySweep several times with different sweep directions and taking the best of the computed solutions.

- For the 2SV model we do one left-to-right sweep (as before) and one right-to-left sweep. In the latter sweep we modify the algorithm so that it prefers rightmost candidates rather than leftmost candidates.
- For the 2SH model we do one top-to-bottom sweep (preferring topmost label candidates) and one bottom-to-top sweep (preferring bottommost label candidates).

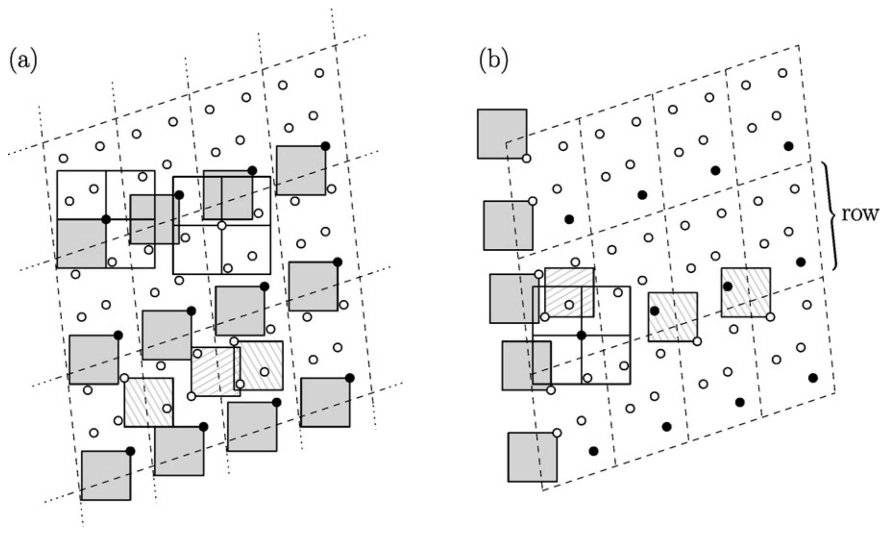


Fig. 5. The approximation factor of GreedySweep can be $O(1/\sqrt{n})$ for the 4P model.

- For the 4P model we use the same two sweeps as for the 2SV model (but using the sweeps for the 2SH model would work as well).
- For the 4S models we sweep in all four directions.

Next we show that this always yields a constant-factor approximation.

Lemma 2. *The approximation factors of the algorithms described are* $1/22$ *for the 4P and 2SV models, and* $1/32$ *for the 4S model.*

Proof. Consider an optimal solution OPT and the solution GS computed by the left-to-right application of GreedySweep. Because of the above discussion we cannot hope to charge each free label in OPT to one in GS in a way that there are few charges to each label. However, we *can* charge the free *rightmost* labels in OPT in such a way. If most free labels in OPT turn out not to be rightmost (as above), then we can symmetrically consider the leftmost free labels in the right-to-left sweep, the bottommost free labels in the top-to-bottom sweep, or the topmost free labels in the bottom-to-top sweep. In this way, at least half the free labels in OPT can be charged for the 4P and 2SV models, and at least a quarter for the 4S model. The charging scheme is as before, but with one extra type of charge. Let p_i be a point with a free rightmost label in OPT.

- Suppose OPT(p_i) is a freeable label candidate at the time p_i is being processed. We charge OPT(p_i) to GS(p_i) (which must also be freeable) and call OPT(p_i) a *type*-0 *charge* to GS(p_i)—see Fig. 6(a).
- Suppose OPT(p_i) is a non-freeable label candidate (at the time p_i is being processed) because it is intersected by a label GS(p_j) that has already been placed—that is, $j < i$.
  - If GS(p_j) is freeable, then we charge OPT(p_i) to GS(p_j) and call OPT(p_i) a *type*-1 *charge* to GS(p_j)—see Fig. 6(b).
  - Suppose GS(p_j) is *not* freeable. Note that since OPT(p_i) is rightmost, and $j < i$, GS(p_j) cannot be leftmost. But if GS(p_j) is not leftmost, then as we move GS(p_j) to the left it must hit a freeable label before it has become leftmost. Let GS(p_k) be the first freeable label hit, where $k < j$. Charge OPT(p_i) to GS(p_k), and call OPT(p_i) a *type*-2 *charge* to GS(p_k) through p_j—see Fig. 6(c).
- Suppose OPT(p_i) is a non-freeable label candidate because it will inevitably be intersected by a label GS(p_j) still to be placed—that is, $j > i$. (Note that this case could not occur in the 2PH and 1SH models.) This means that p_j has some label candidates that intersect OPT(p_i), some label candidates that intersect one or more previously placed freeable labels, and no other label candidates. Since p_i and all points that have received a freeable label lie to the left of p_j, the same must be true of p_j's *rightmost* label candidates. In other words, some of p_j's rightmost candidates intersect OPT(p_i), and some intersect previously placed freeable labels. Let GS(p_k) be such a previously placed freeable label. Charge OPT(p_i) to GS(p_k), and call OPT(p_i) a *type*-3 *charge* to GS(p_k) through p_j—see Fig. 6(d).

To finish the proof of the stated approximation ratios, we will now give upper bounds for the number of charges of each type to a freeable label.

- *Type* 0. Consider a label GS(p_i) that receives a type-0 charge OPT(p_i). Trivially, this must be the only type-0 charge to GS(p_i). If OPT(p_i) and GS(p_i) differ in y-coordinate, then there can still be additional charges of other types to GS(p_i). However, OPT(p_i) must contain (on its boundary if not in its interior) either the top-right or the bottom-right corner

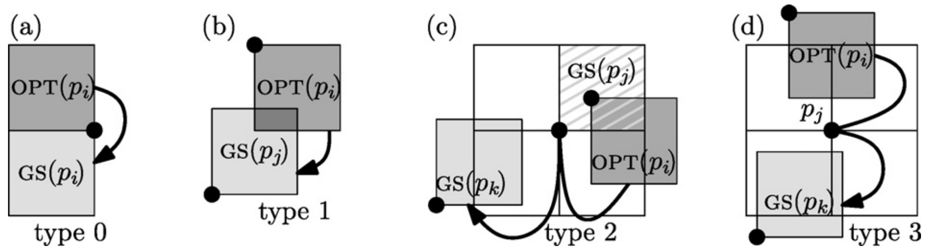


Fig. 6. The four types of charges used in proving the approximation ratio of GreedySweep for the 4P, 2SV, and 4S models.

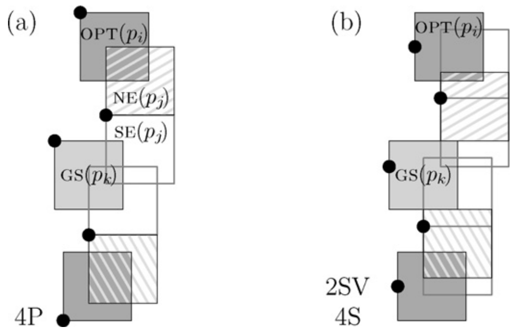


Fig. 7. There can be at most two type-3 charges to a label in (a) the 4P model and (b) the 2SV and 4S models.

of GS(p_i). That corner cannot be intersected by another free label of OPT, so by "adding" a type-0 charge to GS(p_i) we "lose" one type-1 charge.

- *Type* 1. Consider a label GS(p_j) that receives a type-1 charge OPT(p_i). Note that OPT(p_i) must contain a corner of GS(p_j). Since OPT(p_i) is rightmost, and $j < i$, OPT(p_i) cannot contain the top-left or bottom-left corner of GS(p_j). Thus OPT(p_i) must contain the top-right or bottom-right corner of GS(p_j), and hence there can be at most two type-1 charges to GS(p_j).
- *Type* 2. Consider a label GS(p_k) that receives a type-2 charge OPT(p_i) through some point p_j. Note that GS(p_j) will have been placed as far to the left as possible without intersecting freeable labels. Like for the 1SH model, this means that GS(p_j) touches GS(p_k) in the 4S model. In turn, OPT(p_i) intersects GS(p_j). Therefore OPT(p_i) must be fully contained in a rectangle of size $(2-\varepsilon) \times (5-\varepsilon)$, for some $\varepsilon > 0$, whose left side contains the right side of GS(p_k). This was illustrated for the 1SH model in Fig. 3(a). The optimal solution can have at most four free labels completely contained in this rectangle, so GS(p_k) can be charged at most four times in this manner.
In the 4P and 2SV models, as in the 2PH model, GS(p_j) in general does not touch GS(p_k). Therefore the rectangle fully containing OPT(p_k) is of size $(3-\varepsilon) \times (5-\varepsilon)$, as was illustrated for the 2PH model in Fig. 3(b). This would imply that GS(p_k) can receive eight type-2 charges. We reduced this number to seven for the 2PH model, and the same argument applies for the 4P and 2SV models. We also argued for the 2PH model that the total number of type-1 and type-2 charges combined was no more than seven. It is not clear how to apply that reasoning to the 4P and 2SV models, however.
- *Type* 3. Consider a label GS(p_k) that receives a type-3 charge OPT(p_i) through some point p_j. Then some rightmost label candidates for p_j intersect OPT(p_i), and some intersect GS(p_k) or other previously placed freeable labels. In particular, consider the top-rightmost candidate NE(p_j) and the bottom-rightmost candidate SE(p_j). OPT(p_i) cannot intersect both NE(p_j) and SE(p_j), for then OPT(p_i) contains p_j. On the other hand, OPT(p_i) must intersect one of NE(p_j) and SE(p_j). The same reasoning applies to GS(p_k). So either OPT(p_i) intersects NE(p_j) and GS(p_k) intersects SE(p_j), or the other way around. In the former case OPT(p_i) is above GS(p_k), in the latter case OPT(p_i) is below GS(p_k). This leads to the two type-3 charges to GS(p_k) depicted in Fig. 7, and we claim that this is the maximum possible. To see this, consider the type-3 charge with the highest y-coordinate, say OPT(p_i), and suppose it charges through the point p_j. Then other type-3 charges above GS(p_k) with lower y-coordinates than OPT(p_i) must contain p_j, a contradiction. A symmetric argument holds for the type-3 charges below GS(p_k).

Summarizing, the maximum number of charges to a freeable label is eight for the 4S model (two of type 1, four of type 2, two of type 3) and eleven for the 4P and 2SV models (two of type 1, seven of type 2, two of type 3). Recall that we charge at least a quarter of the free labels of OPT in the 4S model, leading to an approximation ratio of 1/32. In the 4P and 2SV models we charge at least half the free labels of OPT, leading to an approximation ratio of 1/22.

We do not claim that these bounds are tight. Even if the bounds on the number of charges to a freeable label prove to be tight, that would not imply tightness of the approximation ratios. For example, a "bad" problem instance for the left-to-right sweep may turn out to be "good" for the right-to-left sweep. □

The following theorem states the main results of this section.

Theorem 1. *There are $O(n \log n)$-time and $O(n)$-space algorithms for free-label maximization on n points with unit-square labels, having the following approximation ratios*: $1/6$ (*tight*) *for the 1SH model,* $1/7$ (*tight*) *for the 2PH model,* $1/22$ *for the 4P and 2SV models, and* $1/32$ *for the 4S model.*

The approximation factors have been proved in Lemmas 1 and 2. In the next subsection we will describe how to implement GreedySweep to run in $O(n \log n)$ time and $O(n)$ space.

2.3. *Efficient implementation*

Recall our constraints on the labels that we place: 1) no label intersects a previously placed *freeable* label, 2) a freeable label does not intersect *any* previously placed label, and 3) every unprocessed point has at least one label candidate not intersecting a previously placed freeable label. For each of these conditions we will maintain a data structure that we can efficiently query for the set of label candidates satisfying the condition. More precisely, we will use:

1. A structure FREE storing the union of the *freeable* labels placed so far, supporting two operations:
   - AddLabel(FREE, L), which updates FREE for a newly placed freeable label L.
   - NonIntersecting(FREE, S), which returns, for a given set of label candidates S, the subset $S' \subseteq S$ of label candidates that do not intersect the labels in FREE.
2. An analogous structure ALL with operations AddLabel(ALL, L) and NonIntersecting(ALL, S) storing the union of *all* labels placed so far.
3. A structure R storing the "remaining" rightmost label candidates $\mathrm{R}(p) \subseteq \mathrm{Cand}(p)$ for every unprocessed point p, that is, the rightmost label candidates not intersecting previously placed freeable labels. (Note that we ignore any non-rightmost remaining label candidates. We will argue later that this is justified.) The structure supports the following three operations:
   - UpdateForFreeLabel(R, L), which removes from R all label candidates intersecting the newly placed freeable label L.
   - LeftmostNonBlocking(R, S), which returns the leftmost label candidates in S which do not "block" the remaining points from being labeled. A label candidate $L \in S$ blocks an unprocessed point p from being labeled if every remaining label candidate $L' \in \mathrm{R}(p)$ of p intersects L.
   - RemoveCandidates(R, p), which removes all the label candidates for point p from R.

With these data structures, the GreedySweep algorithm becomes as follows. The temporary variables $\mathrm{LF}(p_i)$, $\mathrm{PF}(p_i)$, and $\mathrm{NF}(p_i)$, with $\mathrm{LF}(p_i) \subseteq \mathrm{PF}(p_i) \subseteq \mathrm{NF}(p_i) \subseteq \mathrm{Cand}(p_i)$, are mnemonic and denote the leftmost freeable label candidates, the "potentially" freeable label candidates, and the non-freeable label candidates, respectively.

Algorithm GreedySweep(P, Cand)
```
1.  Let $p_1, \ldots, p_n$ be the points in $P$, sorted in order of increasing $x$-coordinate.
2.  Initialize the structures FREE, ALL, and R.
3.  for $i \leftarrow 1$ to $n$
4.      do RemoveCandidates(R, $p_i$)  (* $p_i$ can no longer be blocked *)
5.         PF($p_i$) ← NonIntersecting(ALL, Cand($p_i$))
6.         LF($p_i$) ← LeftmostNonBlocking(R, PF($p_i$))
7.         if LF($p_i$) $\neq \emptyset$  (* $p_i$ has a freeable label candidate *)
8.           then Set $L$ to be any label candidate in LF($p_i$). (All lie equally far to the left.)
9.                AddLabel(FREE, $L$)
10.               UpdateForFreeLabel(R, $L$)
11.          else NF($p_i$) ← NonIntersecting(FREE, Cand($p_i$))
12.               Set $L$ to be the leftmost label candidate in NF($p_i$), with ties broken arbitrarily.
13.        AddLabel(ALL, $L$)
14.        Set GS($p_i$) ← $L$.
```

It remains to demonstrate $O(n)$-space implementations of the proposed data structures with $O(\log n)$ amortized execution time for their operations.

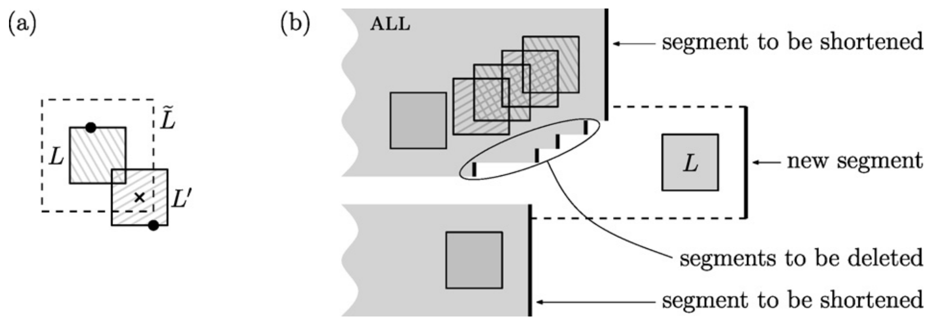


Fig. 8. (a) L' intersects L if and only if its center is contained in the extended label $\widetilde{L}$. (b) The ADDLABEL operation inserts one new segment, shortens at most two existing segments, but may delete many segments.

2.3.1. *Data structures for labels*

To represent FREE and ALL it is not necessary to store the exact union of the labels in question; their right envelope will suffice. This is because points are processed from left to right, and labels touch the points to which they are attached. Such a right envelope can be maintained as a *red–black tree* [4, Ch. 13] storing its vertical segments. We order the tree nodes by y-coordinate, and *augment* [4, Ch. 14] each node v with a field storing the maximum x-coordinate in the subtree rooted at v. For easier queries we will not store the right envelope of the original labels, but of the *extended labels*. For a label L we define its extended label $\widetilde{L}$ to be the 2×2 square with the same center as L. A label L' then intersects L if and only if the center of L' lies inside $\widetilde{L}$—see Fig. 8(a).

We now identify each label (candidate) with its center point. This turns CAND(p_i) into a set of $O(1)$ points for the fixed-position models. For each such point p representing a label candidate, we can query the described red–black tree to determine in $O(\log n)$ time whether p lies to the right of the right envelope maintained in the tree. For the slider models this transformation turns CAND(p_i) into a set of one, two, or four axis-parallel line segments. We can similarly query the red–black tree in $O(\log n)$ time with each such segment s to determine the part of s that lies to the right of the right envelope (if any). The result of each such query is either the empty set, a single point, or a (possibly shorter) line segment. Taking the union of these $O(1)$ query results completes the NONINTERSECTING($\cdot$, CAND(p_i)) operation.

The ADDLABEL($\cdot$, L) operation inserts one new segment, and shortens at most two existing segments, but may delete many segments—see Fig. 8(b). However, only $O(n)$ segment deletions will happen in total. Hence the operation can be performed in amortized $O(\log n)$ time, as required.

2.3.2. *Data structures for unlabeled points*

The structure R is used to maintain the invariant that every unlabeled point has label candidates available to it that do not intersect previously placed freeable labels. Note that if an unprocessed point p_j has any such label candidate, it must have one that is also a rightmost label candidate for p_j. (We elaborated on this when discussing type-3 charges in the proof of Lemma 2.) Thus we defined R(p_j) as the rightmost label candidates of an unprocessed point p_j that do not intersect any previously placed freeable labels. Depending on the label model, R(p_j) is either a set of $O(1)$ points lying on a common vertical line, or a single vertical line segment.

LEFTMOSTNONBLOCKING *for fixed-position models.* The query LEFTMOSTNONBLOCKING(R, PF(p_i)) boils down to determining for which label candidates $L \in$ PF(p_i) there exists no unprocessed point p_j with R(p_j) fully contained in the extended label $\widetilde{L}$. To this end we maintain two *priority search trees* [5, Ch. 10] $\mathcal{P}_b$ and $\mathcal{P}_t$. In $\mathcal{P}_b$ we store the bottommost point of R(p_j) for all p_j, and in $\mathcal{P}_t$ the topmost point. For a single label candidate $L \in$ PF(p_i) with center (x, y) we query $\mathcal{P}_b$ with the region $(-\infty, x+1) \times (y-1, y)$, and $\mathcal{P}_t$ with the region $(-\infty, x+1) \times (y, y+1)$. If the bottommost or topmost point of R(p_j) is found with these queries then $\widetilde{L}$ contains all of R(p_j), otherwise L is freeable—see Fig. 9(a). This is because R(p_j) spans at most 1 unit of vertical space. Such queries can be performed in $O(\log n)$ time, so we can also execute LEFTMOSTNON-BLOCKING(R, PF(p_i)) in $O(\log n)$ time when PF(p_i) is a set of $O(1)$ points. That covers the fixed-position models.

LEFTMOSTNONBLOCKING *for slider models.* For the slider models, let PF$_\ell$(p_i), PF$_r$(p_i), PF$_t$(p_i), and PF$_b$(p_i) denote the leftmost, rightmost, topmost, and bottommost label candidates of PF(p_i), respectively. Since we identify a label candidate with its center point, each of these is either the empty set—so we can ignore it—, or an axis-parallel line segment. The vertical segment PF$_\ell$(p_i) is trivial to handle. It represents leftmost label candidates for the point p_i being processed, which can never intersect any of the rightmost label candidates R(p_j) for a point p_j still to be processed. So if PF$_\ell$(p_i) $\neq \emptyset$, then PF$_\ell$(p_i) contains exactly the leftmost freeable label candidates, and we are done. Otherwise, we look at the horizontal segments PF$_t$(p_i) and PF$_b$(p_i). If the left endpoint of such a segment is not freeable, then no other point on the segment is either. Thus we simply query $\mathcal{P}_b$ and $\mathcal{P}_t$ with the left endpoints of PF$_t$(p_i) and PF$_b$(p_i) in the same way as above. If neither left endpoint represents a freeable label candidate we have to look at the vertical segment PF$_r$(p_i). This case is more complicated.

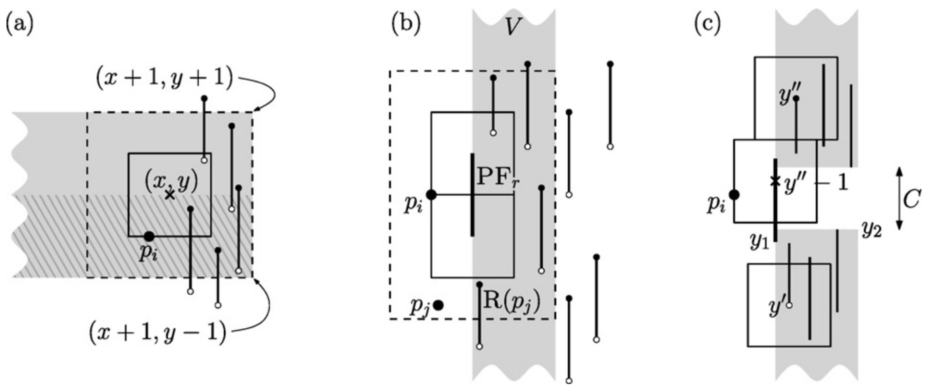


Fig. 9. (a) $\mathrm{R}(p_j)$ is contained in $\widetilde{L}$ if and only if its bottommost point (∘) is contained in the hatched area, or its topmost point (•) in the unhatched area. (b) Only the segments in V can prevent label candidates of $\mathrm{PF}_r(p_i)$ from being freeable. (c) The freeable label candidates for p_i are in the interval $[y'+1, y''-1] \cap C$.

Let x be the x-coordinate of the vertical segment $\mathrm{PF}_r(p_i)$, and let V be the vertical strip between x-coordinates x and $x+1$. Note that only the vertical segments of R which lie inside V can cause points on $\mathrm{PF}_r(p_i)$ to be non-freeable—see Fig. 9(b). We denote these segments by R_V. Within V we are essentially left with a 1-dimensional problem on intervals along the y-axis. We wish to determine the points y on interval $\mathrm{PF}_r(p_i)$ such that the interval $(y-1, y+1)$ does not fully contain any interval of R_V. Note that such a point y cannot be contained in $\mathrm{R}(p_j)$ for any p_j, because $\mathrm{R}(p_j)$ has at most length 1 and would in turn be fully contained in $(y-1, y+1)$. Thus we need to look for such points y in the complement of the union of the intervals R_V. Consider one component C of the complement, and partition the intervals R_V into the set R_b below C and the set R_t above C. Let y' be the topmost bottom endpoint among the intervals in R_b, and let y'' be the bottommost top endpoint among the intervals in R_t. The interval $[y'+1, y''-1] \cap C$, if non-empty, then consists of all freeable label candidates in C—see Fig. 9(c). Our strategy is to loop over the components C overlapping $\mathrm{PF}_r(p_i)$, and determining the points y' and y'' for each of them.

In order to determine y' we build a red–black tree $\mathcal{T}_b$ on the y-coordinates of the bottom endpoints of the intervals in R_V. For determining y'' we build a similar tree $\mathcal{T}_t$ on the top endpoints. By performing searches in these two trees we can find y' and y'' in $O(\log n)$ time.

To find the components C, first note that there is at most one such component to be found. Otherwise there would be an interval $\mathrm{R}(p_j)$ fully contained in $\mathrm{PF}_r(p_i)$. The interval $\mathrm{R}(p_j)$ must have a length less than 1, so its top or bottom endpoint touches the right envelope maintained by FREE. Denoting the area to the left of this right envelope by $\mathcal{H}(\text{FREE})$, this means $\mathrm{PF}_r(p_i)$ is intersected by $\mathcal{H}(\text{FREE})$, and therefore by $\mathcal{H}(\text{ALL}) \supseteq \mathcal{H}(\text{FREE})$. However, that is impossible because $\mathrm{PF}(p_i) = \mathrm{CAND}(p_i) \setminus \mathcal{H}(\text{ALL})$.

In order to find the component C we build an *interval tree* [5, Ch. 10] $\mathcal{I}$ on the intervals R_V. Let y_1 denote the bottom endpoint of $\mathrm{PF}_r(p_i)$. Using $\mathcal{I}$ we can determine in $O(\log n)$ time whether y_1 is contained in any interval of R_V. If not, then $y_1 \in C$. Otherwise we can determine, in the same time bound, the maximum top endpoint y_2 among the intervals in which y_1 is contained. We then query $\mathcal{I}$ again to determine if y_2 is contained in any intervals. If not, we have $y_2 \in C$. Otherwise the intervals containing y_1 and the intervals containing y_2 together cover $\mathrm{PF}_r(p_i)$, so no suitable C exists.

Building the trees $\mathcal{T}_t$, $\mathcal{T}_b$, and $\mathcal{I}$ from scratch for each point would be too slow. However, as we process points from left to right the strip V moves monotonically from left to right. When a segment $\mathrm{R}(p_j)$ enters the strip we insert it into the three trees, and when it leaves the strip we remove it from the trees again, both in $O(\log n)$ time. As each point enters and leaves the strip at most once, this allows us to maintain these trees in amortized $O(\log n)$ time per point. This concludes the implementation details of the LEFTMOSTNONBLOCKING operation.

Other operations on R. For the operation UPDATEFORFREELABEL(R, L) we first find all segments in R with an endpoint in $\widetilde{L}$. Using $\mathcal{P}_t$ and $\mathcal{P}_b$ this can be done in $O(\log n + k)$ time, where k is the number of segments found. We then cut off the parts of these segments in the interior of $\widetilde{L}$, and update all five aforementioned trees ($\mathcal{P}_t$, $\mathcal{P}_b$, $\mathcal{T}_t$, $\mathcal{T}_b$, and $\mathcal{I}$) for the shortened segments in $O(k \log n)$ time. In the worst case $k = \Omega(n)$. However, each segment can be shortened at most twice: once when its top endpoint falls within an extended freeable label, and once when its bottom endpoint does. It cannot be shortened more often as freeable labels do not intersect each other. Thus the amortized time needed for UPDATEFORFREELABEL(R, L) is $O(\log n)$.

The last operation, REMOVECANDIDATES(R, p_i), simply removes $\mathrm{R}(p_i)$ from all five trees in $O(\log n)$ time. Thus we have shown an implementation of GREEDYSWEEP using $O(n \log n)$ time and $O(n)$ space, completing the proof of Theorem 1.

3. PTASs for unit squares

We can obtain a PTAS for the case of unit-square labels by applying the “shifting technique” of Hochbaum and Maass [13]. Imagine a grid of unit squares overlaying the plane such that no point is on a grid line and call this the 1-*grid*. If, for some

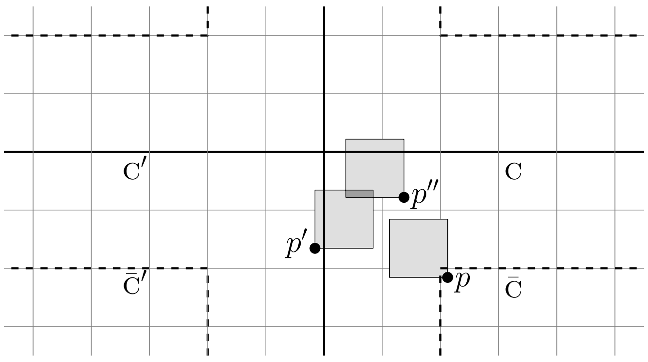


Fig. 10. The cells (bold lines) and inner cells (dashed bold lines) of a k-grid. The underlying 1-grid is shown in thin gray lines.

integer $k > 4$ to be specified later, we leave out all but every kth horizontal and vertical grid line this forms a coarser k-*grid*. By varying the offsets at which we start counting the lines, we can form k^2 different k-grids $G_1, \dots, G_{k^2}$ out of the 1-grid. Consider one of them, say G_i. For any $k \times k$ square cell $\mathrm{c} \in G_i$, let $\bar{\mathrm{c}} \subset \mathrm{c}$ be the smaller $(k-4) \times (k-4)$ square with the same center as c—see Fig. 10. We call $\bar{\mathrm{c}}$ the *inner cell* of c. For a given set P of n points, let $P_{\mathrm{c}} := \mathrm{c} \cap P$ and $P_{\bar{\mathrm{c}}} := \bar{\mathrm{c}} \cap P$. Furthermore, define $P_{\mathrm{in}}(G_i) := \bigcup_{\mathrm{c} \in G_i} P_{\bar{\mathrm{c}}}$. We call a labeling $\mathcal{L}$ for P *inner-optimal* with respect to G_i if $\mathcal{L}$ maximizes the number of points in $P_{\mathrm{in}}(G_i)$ that get a free label. Note that if $\mathrm{c}, \mathrm{c}' \in G_i$ are distinct cells, then a point $p \in \bar{\mathrm{c}}$ can never have a label intersecting the label for a point $p' \in \mathrm{c}'$—see Fig. 10. Hence an inner-optimal labeling for P can be obtained by computing an inner-optimal labeling on P_{c} independently for each cell $\mathrm{c} \in G_i$. We will show below how to do this in time polynomial in n (but exponential in k). By itself this does not help us, as any particular k-grid G_i may have many points that lie outside of inner cells. We claim, however, that computing an inner-optimal labeling for all k-grids $G_1, \dots, G_{k^2}$ and then taking the best one still yields a $(1-\varepsilon)$-approximation for suitably chosen k:

Lemma 3. *For all fixed-position and slider models, the best inner-optimal labeling for a set P of n points in the plane with respect to all k^2 different k-grids $G_1, \dots, G_{k^2}$ yields a $(1-\varepsilon)$-approximation to free-label maximization with unit-square labels if $k \geqslant 8/\varepsilon$.*

Proof. Let OPT be some optimal solution, and let $F \subseteq P$ be the set of points with a free label in OPT. Let $f := |F|$. In any k-grid the inner cells are separated from each other by horizontal and vertical strips with a width of four 1-grid cells—see Fig. 10. Thus any point in F lies in an inner cell for $(k-4)^2$ of the k^2 different k-grids. By the pigeon-hole principle, there must be a k-grid G_i for which

$$\left|F \cap P_{\mathrm{in}}(G_i)\right| \geqslant (k-4)^2/k^2 \cdot f = (1-4/k)^2 \cdot f.$$

An inner-optimal labeling for P with respect to G_i will have at least $|F \cap P_{\mathrm{in}}(G_i)|$ free labels. Hence we get a $(1-\varepsilon)$-approximation if

$$(1-4/k)^2 = 1 - 8/k + 16/k^2 \geqslant 1 - \varepsilon,$$

which is satisfied if $k \geqslant 8/\varepsilon$. □

To complete the PTAS we need to show how to compute an inner-optimal labeling for the set P_{c} of points inside a $k \times k$ cell c. We say that a subset $F \subseteq P_{\bar{\mathrm{c}}}$ is *freeable* if we can label the points in P_{c} such that all points in F get a free label. The key insight is that, by a packing argument, not too many of the points $P_{\bar{\mathrm{c}}}$ in the inner cell $\bar{\mathrm{c}}$ can get a free label. Thus there is a limited number of freeable subsets. We first bound the number of potentially freeable subsets that we need to consider, and then show how to test each one for feasibility.

In many applications, there will not be too many points that are very close together (with respect to the label sizes). To take this into account, we will not just use the total number of points (n) in our analysis, but also their "density" (Δ). More precisely, let $\Delta \leqslant n$ denote the maximum number of points in P contained in any unit square. If $\Delta = 1$ then labeling every point with its top-left label candidate, say, yields a solution where all labels are free. So we assume $\Delta \geqslant 2$ from now on.

Lemma 4. *Let* c *be a cell in a k-grid and $\bar{\mathrm{c}}$ be its inner cell, and let P_{c} and $P_{\bar{\mathrm{c}}}$ be the respective subsets of points inside these cells. We can compute in $O(\sum_{F \in \mathcal{F}} |F|)$ time a collection $\mathcal{F}$ of subsets of $P_{\bar{\mathrm{c}}}$ such that for any freeable subset $F \subseteq P_{\bar{\mathrm{c}}}$ we have $F \in \mathcal{F}$. The collection $\mathcal{F}$ satisfies*:

- $|\mathcal{F}| \leqslant \Delta^{2(k-4)^2}$ *for the 2PH and 1SH models,*
- $|\mathcal{F}| \leqslant \Delta^{4(k-4)^2}$ *for the 4P, 2SV, and 4S models, and*
- $|F| \leqslant (k-3)^2$ *for all $F \in \mathcal{F}$.*

Proof. A 1-grid cell contains at most Δ points, and $\bar{\mathrm{c}}$ consists of $(k-4)^2$ cells of the 1-grid. In the 2PH and 1SH models, no more than two points from the same 1-grid cell can be simultaneously labeled with non-intersecting labels. Thus any freeable subset $F \subseteq P_{\bar{\mathrm{c}}}$ can be constructed by taking at most two points from each 1-grid cell. Hence, there are at most

$$\left(\binom{\Delta}{0}+\binom{\Delta}{1}+\binom{\Delta}{2}\right)^{(k-4)^2} \leqslant \Delta^{2(k-4)^2}$$

potentially freeable subsets F, where the inequality follows from the assumption that $\Delta \geqslant 2$. Similarly, no more than four points from the same 1-grid cell can be simultaneously labeled with non-intersecting labels in the 4P, 2SV, and 4S models, leading to at most

$$\left(\binom{\Delta}{0}+\binom{\Delta}{1}+\binom{\Delta}{2}+\binom{\Delta}{3}+\binom{\Delta}{4}\right)^{(k-4)^2} \leqslant \Delta^{4(k-4)^2}$$

potentially freeable subsets F.

In addition to limiting the number of points taken from each 1-grid cell, we can also limit the total number of points that we need to take. Since all labels are unit squares, the labels of the points in the $(k-4)\times(k-4)$ square $\bar{\mathrm{c}}$ must be fully contained in a slightly larger $(k-2-\delta)\times(k-2-\delta)$ square around $\bar{\mathrm{c}}$, for some $\delta > 0$. There can be at most $(k-3)^2$ free labels in this area, hence $|F| \leqslant (k-3)^2$ for all freeable subsets $F \subseteq P_{\bar{\mathrm{c}}}$. □

The previous lemma states that there are not too many potentially freeable subsets, and that each one is fairly small. The next two lemmas show, for different label models, how to test whether a potentially freeable subset is really freeable. We start with the 2PH, 1SH, and 4P models.

Lemma 5. *Let P_{c} be the set of all $n_{\mathrm{c}} \leqslant \min(k^2\Delta, n)$ points contained in a k-grid cell* c, *and let $F \subseteq P_{\bar{\mathrm{c}}}$ be a subset of those points. Let $f := |F|$. Then deciding whether there exists a labeling $\mathcal{L}$ for P_{c} where all points in F have a free label, and if so producing $\mathcal{L}$, can be done*

(i) *in $O(n_{\mathrm{c}} \log n_{\mathrm{c}})$ time for the 2PH and 1SH models, and*
(ii) *in $O((n_{\mathrm{c}} - f)4^f)$ time for the 4P model.*

Proof. (i) Go through the points from left to right and label them one-by-one. For every point $p \in F$ we pick the leftmost label candidate that does not intersect a previously placed label, and for every point $p \in P_{\mathrm{c}} \setminus F$ we pick the leftmost label candidate that does not intersect a previously placed label for a point in F. If we can process all points in P_{c} in this way then clearly we have found a suitable labeling $\mathcal{L}$. If we instead encounter a point p for which no label candidate can be chosen, then we report that no such labeling $\mathcal{L}$ exists. This is correct, because the partial labeling constructed by this algorithm has all labels at least as far to the left as $\mathcal{L}$ would have, so p cannot be correctly labeled in $\mathcal{L}$ either. The above is simply a somewhat simplified version of the GreedySweep algorithm from Section 2, and can be implemented to run in $O(n_{\mathrm{c}} \log n_{\mathrm{c}})$ time.

(ii) Enumerate all 4^f labelings of the points in F, and check for each such labeling $\mathcal{L}'$ whether it can be extended into a labeling $\mathcal{L}$ for all points in P_{c}. This entails checking whether each point $p \in P_{\mathrm{c}} \setminus F$ has a label candidate that does not intersect any label of $\mathcal{L}'$. For this we only need to look at labels for points $p' \in F$ that lie in the 3×3 square of 1-grid cells centered at the 1-grid cell containing p. We can access these points efficiently by populating in advance a $(k-4)\times(k-4)$ array of "buckets" storing the points of F contained in each 1-grid cell. Since each 1-grid cell contains at most four points of F, performing the above check can then be done in $O(1)$ time for each of the $n_{\mathrm{c}} - f$ points in $P_{\mathrm{c}} \setminus F$. □

For the 2SV model we can neither use the greedy labeling of F (as for the 2PH and 1SH models), nor try all labelings (as for the 4P model). Instead we proceed as follows. Try all 2^f ways of restricting the labels for the points in F to be either leftmost or rightmost. The problem is then to decide whether F can be labeled with free labels in the 1SV model, while labeling $P_{\mathrm{c}} \setminus F$ with (free and/or non-free) labels in the 2SV model. The position of a label along a 1-slider can be modeled as a number between 0 and 1, making the configuration space $\mathcal{C}$ of possible labelings for F the f-dimensional unit hypercube. Let $\mathcal{C}_{\text{nonint}} \subseteq \mathcal{C}$ be the subspace of labelings for F where the labels of the points in F are disjoint. For any point $p \in P_{\mathrm{c}} \setminus F$ let $\mathcal{C}_p \subseteq \mathcal{C}$ be the subspace of labelings $\mathcal{L}' \in \mathcal{C}_p$ for F where p can still get a label without intersecting a label in $\mathcal{L}'$. We then need to decide whether $\mathcal{C}_{\text{free}} := \mathcal{C}_{\text{nonint}} \cap (\bigcap_{p \in P_{\mathrm{c}} \setminus F} \mathcal{C}_p)$ is non-empty, and if so construct a feasible labeling $\mathcal{L}' \in \mathcal{C}_{\text{free}}$ for F and extend it into a labeling $\mathcal{L}$ for P_{c}. We will show how this can be done using an arrangement of $O(n_{\mathrm{c}})$ hyperplanes in $\mathcal{C}$.

Lemma 6. *Let P_{c} be the set of all $n_{\mathrm{c}} \leqslant \min(k^2\Delta, n)$ points contained in a k-grid cell* c, *and let $F \subseteq P_{\bar{\mathrm{c}}}$ be a subset of those points. Let $f := |F|$. Then deciding whether there exists a labeling $\mathcal{L}$ for P_{c} where all points in F have a free label, and if so producing $\mathcal{L}$, can be done in $O(n_{\mathrm{c}})^f$ time for the 2SV and 4S models.*

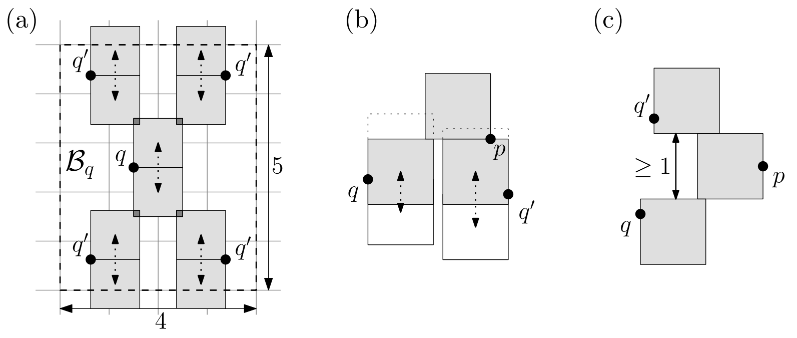


Fig. 11. (a) Any point q' whose label intersects the label of a point q must lie in a region $\mathcal{B}_q$ of 20 cells of the 1-grid. (b) The labels of points $q, q' \in F$ on one side of p may not cross the horizontal line through p. (c) The labels of points $q, q' \in F$ on opposite sides of p need to be at least 1 unit apart vertically.

Proof. We discuss only the 2SV model; the proof for the 4S model is analogous. For each point $p \in F$, we restrict its labeling to be either leftmost or rightmost—this can be done in 2^f ways—and from now on we consider this restriction to be fixed.

If two points $q, q' \in F$ have intersecting labels, then they must be fairly close. Specifically, there is a 5×4 rectangle $\mathcal{B}_q$ around q, consisting of 20 cells of the 1-grid, such that $\mathcal{B}_q$ contains q'—see Fig. 11(a). Assuming it is possible for q and q' to have intersecting labels in the first place, preventing that from happening introduces a linear constraint on the slider coordinates of q and q'. Since F has at most four points in any 1-grid cell, $\mathcal{B}_q$ contains at most 80 points of F (including q itself). Hence $\mathcal{C}_{\text{nonint}}$ is the intersection of at most $f \cdot (80-1) \cdot 1/2 = 79f/2$ half-spaces. (The factor $1/2$ is because the half-space for a pair q, q' is counted twice, once for q and once for q'.)

For any point $p \in P_c \setminus F$, let $\mathcal{C}_p^\ell \subseteq \mathcal{C}$ be the subspace of labelings for F which still allow p to get a leftmost label. Now consider a labeling for F that is *not* in $\mathcal{C}_p^\ell$. Any leftmost label for p will intersect the label of at least one point of F in such a labeling. We claim (and will argue later) that there must then exist a subset $F' \subseteq F$ with $|F'| \leqslant 2$ such that every leftmost label of p intersects a label of a point in F'. Hence, $\mathcal{C}_p^\ell$ can be constructed as $\mathcal{C}_p^\ell = \bigcap_{F' \subseteq F,\, |F'| \leqslant 2} \mathcal{C}_p^\ell(F')$, where $\mathcal{C}_p^\ell(F')$ is the subspace of labelings for F where p has at least one leftmost label candidate not intersecting the labels for F'. Since $F' \subseteq \mathcal{B}_p$, there are at most $\binom{80}{1} + \binom{80}{2} = 3240$ sets F' to be considered. For $q \in F$ the subspace $\mathcal{C}_p^\ell(\{q\})$ takes on one of two shapes:

- If no label candidate of q by itself intersects all leftmost label candidates of p, then $\mathcal{C}_p^\ell(\{q\})$ is the full hypercube.
- Otherwise $\mathcal{C}_p^\ell(\{q\})$ is a half-space defined by a linear constraint on q's slider coordinate, giving it a minimum or maximum value—see Fig. 11(b).

For $q, q' \in F$ the subspace $\mathcal{C}_p^\ell(\{q, q'\})$ takes on one of three shapes:

- If there is no pair of label candidates for q and q' that together intersect all leftmost label candidates of p, then $\mathcal{C}_p^\ell(\{q, q'\})$ is the full hypercube.
- Otherwise, suppose p is vertically in between q and q', say q is below p and p is below q'. Then $\mathcal{C}_p^\ell(\{q, q'\})$ is a half-space defined by the linear constraint that the vertical distance between the labels of q and q' should be at least 1—see Fig. 11(c).
- Lastly, if q and q' are on the same side (vertically) of p, then $\mathcal{C}_p^\ell(\{q, q'\}) = \mathcal{C}_p^\ell(\{q\}) \cap \mathcal{C}_p^\ell(\{q'\})$—see Fig. 11(b).

Hence, $\mathcal{C}_p^\ell$ can be constructed as the intersection of at most 3240 half-spaces. The same is true for $\mathcal{C}_p^r$, the subspace of labelings for F which allow p to get a rightmost label. Since $\mathcal{C}_p = \mathcal{C}_p^\ell \cup \mathcal{C}_p^r$, we can find $\mathcal{C}_{\text{free}} = \mathcal{C}_{\text{nonint}} \cap (\bigcap_{p \in P_c \setminus F} \mathcal{C}_p)$ as the union of some of the cells in an arrangement of (at most) $h := 79f/2 + 2 \cdot 3240 n_c = O(n_c)$ hyperplanes. We can construct this arrangement in $O(h^f)$ time [8]. In the same time we can test whether $\mathcal{C}_{\text{free}}$ is non-empty and if so construct a labeling $\mathcal{L}' \in \mathcal{C}_{\text{free}}$ for F. Greedily extending the labeling $\mathcal{L}'$ for F into a labeling $\mathcal{L}$ for $P_c \supseteq F$ does not increase the running time.

Recall that we have to do all this for each of the 2^f ways of restricting the labeling of the points in F to leftmost or rightmost. Hence, the overall running time is $2^f \cdot O(n_c)^f = O(n_c)^f$.

It remains to prove the claim that we can ignore sets $F' \subseteq F$ with three or more elements. To this end, consider a labeling $\mathcal{L}'$ for F which intersects all leftmost label candidates for p. Let L be the topmost label in $\mathcal{L}'$ that intersects p's bottom-leftmost label candidate, and let L' be the bottommost label in $\mathcal{L}'$ that intersects p's top-leftmost label candidate (possibly with $L' = L$). Then L and L' together must intersect all leftmost label candidates for p, otherwise p would have a free leftmost label candidate vertically between L and L'. □

Putting together the above lemmas yields the following result.

Theorem 2. *For any fixed $\varepsilon > 0$, and for each of the fixed-position and slider models, there exists a polynomial-time algorithm that computes a $(1-\varepsilon)$-approximation for free-label maximization with unit-square labels.*

The worst-case running time of the algorithm is $n^{O(1/\varepsilon^2)}$. If the density Δ of the point set is $O(1)$—that is, any unit square contains $O(1)$ points—then the running time improves to $O(n\log n) + n2^{O(1/\varepsilon^2)}$ for the 2PH, 4P, and 1SH models, and to $O(n\log n) + n2^{O(1/\varepsilon^2\log(1/\varepsilon))}$ for the 2SV and 4S models.

Proof. Compute a 1-grid in $O(n\log n)$ time [13]. Let $k = \lceil 8/\varepsilon \rceil$ and generate all k^2 possible k-grids $G_1, \ldots, G_{k^2}$ out of the 1-grid. For each k-grid G_i, we compute an inner-optimal labeling for the (at most n) cells containing points. This is done for a cell $c \in G_i$ by enumerating the potentially freeable subsets F of $P_{\bar{c}}$ (Lemma 4), and checking for each subset F whether P_c can be labeled so that all points in F have a free label (Lemmas 5 and 6). The best out of these k^2 solutions is a $(1-\varepsilon)$-approximation (Lemma 3). Let $n_c \leqslant \min(\Delta k^2, n)$ be the maximum number of points in any k-grid cell. The running time is then

$$O(n\log n) + nk^2 \cdot \begin{cases} \Delta^{2(k-4)^2} \cdot O(n_c \log(n_c)) & \text{for the 2PH and 1SH models,} \\ \Delta^{4(k-4)^2} \cdot O(n_c 4^{(k-3)^2}) & \text{for the 4P model,} \\ \Delta^{4(k-4)^2} \cdot O(n_c)^{(k-3)^2} & \text{for the 2SV and 4S models.} \end{cases}$$

For $\Delta = O(n)$ all these running times are $\lceil 8/\varepsilon \rceil^2 n^{O(1/\varepsilon^2)}$, which simplifies to $n^{O(1/\varepsilon^2)}$ (assuming $n > 1$). For $\Delta = O(1)$ this improves to $O(n\log n) + n2^{O(1/\varepsilon^2)}$ for the 2PH, 4P, and 1SH models, and to $O(n\log n) + n2^{O(1/\varepsilon^2\log(1/\varepsilon))}$ for the 2SV and 4S models. □

4. Conclusion

Air-traffic controllers monitor airplanes on computer screens as moving points with associated textual labels, and warn pilots to change course on potential collisions. Currently they spend a lot of their time moving labels around by hand to prevent labels from intersecting one another and becoming unreadable. Algorithms from the cartographic map labeling literature do not apply, as these solve a different problem. To this end we have introduced the *free-label-maximization problem* as a new variant of the labeling problem, and have studied it for static points as a first step. In free-label maximization we must label all points with labels of fixed dimensions and seek to maximize the number of *free* labels (labels that do not intersect other labels). We have presented a simple and efficient constant-factor approximation algorithm, as well as a PTAS, for free-label maximization under the commonly assumed model that labels are directly attached to their points. In air-traffic control, however, labels are usually connected to their point by means of a short line segment (a *leader*). Our constant-factor approximation can be extended to this case, although the approximation factors get worse. We believe the PTAS may be extendable as well, although this involves some more technical details.

Our algorithms work if all labels are unit squares (or, equivalently, all labels are translates of a fixed rectangle). The cases of labels being unit-height rectangles or arbitrary rectangles are still open. For the number-maximization problem these cases allow, respectively, a PTAS [3] and an $O(1/\log\log n)$-approximation [2]. The former achieves a $(1-1/k)$-approximation to number maximization in only $O(n\log n + n\Delta^{k-1})$ time, while the running time of our PTAS for free-label maximization is completely impractical. It would be interesting to see if these results for number maximization can be matched for free-label maximization. If not, then free-label maximization is strictly harder than number maximization, while easier than size maximization. The weighted version of the free-label-maximization problem is another interesting direction for future research.

The most important area for future research, however, is the labeling of moving points. Even outside of air-traffic control applications, we believe that free-label maximization is a better model for this than the size- and number-maximization problems. Continuously scaling labels under size maximization would be hard to read, and the (dis)appearance of a label under number maximization is an inherently discrete event which can be disturbing for the viewer. It is fairly simple to kinetically maintain the labeling of our constant-factor approximation algorithm as the points move. This is not enough to obtain a good result, however, as labels will sometimes “jump” from place to place. We would prefer to “smooth out” the label trajectories so that labels move continuously at finite speeds, but it is not yet clear how to do this.

References

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

[2] P. Chalermsook, J. Chuzhoy, Maximum independent set of rectangles, in: C. Mathieu (Ed.), Proc. 20th ACM–SIAM Sympos. on Discrete Algorithms (SODA'09), New York, 2009, pp. 892–901.

[3] T.M. Chan, A note on maximum independent set in rectangle intersection graphs, Inform. Process. Lett. 89 (2004) 19–23.

[4] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, 3rd ed., MIT Press, 2009.

[5] M. de Berg, O. Cheong, M. van Kreveld, M. Overmars, Computational Geometry: Algorithms and Applications, 3rd ed., Springer-Verlag, 2008.

[6] M. de Berg, D.H.P. Gerrits, Approximation algorithms for free-label maximization, in: H. Kaplan (Ed.), Proc. 12th Scandinavian Sympos. and Workshops on Algorithm Theory (SWAT'10), Bergen, 2010, pp. 297–308.

[7] A. Dorbes, Requirements for the implementation of automatic and manual label anti-overlap functions, EEC Note No. 21/00, EUROCONTROL Experimental Centre, 2000.
[8] H. Edelsbrunner, J. O'Rourke, R. Seidel, Constructing arrangements of lines and hyperplanes with applications, SIAM J. Comput. 15 (2) (1986) 341–363.
[9] T. Erlebach, T. Hagerup, K. Jansen, M. Minzlaff, A. Wolff, Trimming of graphs, with an application to point labeling, in: S. Albers, P. Weil (Eds.), Proc. 25th Internat. Sympos. Theoretical Aspects Comput. Sci. (STACS'08), Bordeaux, 2008, pp. 265–276.
[10] M. Formann, F. Wagner, A packing problem with applications to lettering of maps, in: Proc. 7th Annu. ACM Sympos. Comput. Geom. (SoCG'91), North Conway, 1991, pp. 281–288.
[11] R.J. Fowler, M.S. Paterson, S.L. Tanimoto, Optimal packing and covering in the plane are NP-complete, Inform. Process. Lett. 12 (1981) 133–137.
[12] A. Gemsa, M. Nöllenburg, I. Rutter, Consistent labeling of rotating maps, in: F. Dehne, J. Iacono, J.-R. Sack (Eds.), Proc. 11th Internat. Sympos. Algorithms and Data Structures (WADS'11), New York, USA, 2011, pp. 451–462.
[13] D.S. Hochbaum, W. Maass, Approximation schemes for covering and packing problems in image processing and VLSI, J. ACM 32 (1) (1985) 130–136.
[14] M. Jiang, S. Bereg, Z. Qin, B. Zhu, New bounds on map labeling with circular labels, in: Proc. 15th Annu. Internat. Sympos. Algorithms Comput. (ISAAC'04), 2004, pp. 606–617.
[15] M. van Kreveld, T. Strijk, A. Wolff, Point labeling with sliding labels, Comput. Geom. Theory Appl. 13 (1999) 21–47.
[16] J. Marks, S. Shieber, The computational complexity of cartographic label placement, Technical Report TR-05-91, Harvard, CS, 1991.
[17] S.-H. Poon, C.-S. Shin, T. Strijk, T. Uno, A. Wolff, Labeling points with weights, Algorithmica 38 (2) (2003) 341–362.
[18] F. Rostamabadi, M. Ghodsi, A fast algorithm for updating a labeling to avoid a moving point, in: Proc. 16th Canadian Conf. Comput. Geom. (CCCG'04), 2004, pp. 204–208.
[19] A. Wolff, T. Strijk, The Map Labeling Bibliography, http://liinwww.ira.uka.de/bibliography/Theory/map.labeling.html, 2009.