Labeling Points with Weights*

Sheung-Hung Poon† Chan-Su Shin‡ Tycho Strijk§
Takeaki Uno¶ Alexander Wolff‖

Submitted to Algorithmica Feb. 18, 2002. Revised Oct. 11, 2002

Abstract

Annotating maps, graphs, and diagrams with pieces of text is an important step in information visualization that is usually referred to as label placement. We define nine label-placement models for labeling points with axis-parallel rectangles given a weight for each point. There are two groups: fixed-position models and slider models. We aim to maximize the weight sum of those points that receive a label.

We first compare our models by giving bounds for the ratios between the weights of maximum-weight labelings in different models. Then we present algorithms for labeling n points with unit-height rectangles. We show how an $O(n \log n)$-time factor-2 approximation algorithm and a PTAS for fixed-position models can be extended to handle the weighted case. Our main contribution is the first algorithm for weighted sliding labels. Its approximation factor is $(2+\varepsilon)$, it runs in $O(n^2/\varepsilon)$ time and uses $O(n/\varepsilon)$ space. We show that other than for fixed-position models even the projection to one dimension remains NP-hard.

For slider models we also investigate some special cases, namely (a) the number of different point weights is bounded, (b) all labels are unit squares, and (c) the ratio between maximum and minimum label height is bounded.

Keywords

Computational geometry, GIS, label placement, sliding labels, combinatorial optimization, job scheduling, throughput maximization, maximum weight independent set.

*A preliminary version of this paper [PSSW01] was presented at ISAAC'01 with support of the German Research Foundation (DFG)

†Department of Computer Science, HKUST, Hong Kong. Email: hung@cs.ust.hk

‡School of Electronics and Information Engineering, Hankuk University of Foreign Studies, Korea. Email: cssin@hufs.ac.kr

§ASM Lithography, Veldhoven, The Netherlands. Email: t.strijk@freeler.nl

¶National Institute of Informatics, 2-1-2 Chiyoda-ku, Tokyo, 101-8430, Japan. Email: uno@nii.ac.jp

‖Faculty of Computer Science, Karlsruhe University, P.O. Box 6980, 76128 Karlsruhe, Germany. Internet: http://i11www.ira.uka.de/~awolff

Contents

1 Introduction

Label placement is one of the key tasks in the process of information visualization. In diagrams, maps, technical or graph drawings, features like points, lines, and polygons must be labeled to convey information. The interest in algorithms that automate this task has increased with the advance in type-setting technology and the amount of information to be visualized. Due to the NP-hardness of the general label-placement problem [FW91], cartographers, graph drawers, and computational geometers have suggested numerous approaches, such as expert systems [AF84], zero-one integer programming [Zor90], approximation algorithms [AvKS98, vKSW99, SvK02], simulated annealing [CMS95] and force-driven algorithms [Hir82] to name only a few. An extensive bibliography about label placement can be found in [WS96]. The ACM Computational Geometry Impact Task Force report [Cc99] identifies label placement as an important research area. Manually labeling a map is a tedious task that is estimated to take 50% of total map production time [Mor80].

This paper deals with one of the most common label-placement problems, namely labeling points with axis-parallel rectangles. With two exceptions this is the first paper that gives approximation algorithms for labeling *weighted* points. The aim is to maximize the sum of the weights of those points whose labels can be placed without intersection. Solving this problem is extremely important in praxis: on a map of Germany, e.g., attributing Berlin a higher priority (weight) than Wannsee ensures that in case of limited space the capital rather than one of its districts receives a label. The only two other approximation algorithms for weighted label placement are the following. First, Iturriaga [Itu99] showed how a factor-$O(\log n)$ approximation algorithm of Agarwal et al. [AvKS98] for maximum-independent set on rectangle-intersection graphs can be extended to handle weighted rectangles as well (n is the number of rectangles here). Second, Erlebach et al. [EJS01] recently improved these results for squares; they give a polynomial-time approximation scheme (PTAS) for the weighted case.

Van Kreveld et al. defined six point-labeling models [vKSW99]. They forged the term of *slider models* where a label can slide along one or several edges under the constraint that it touches the point it labels, see Figure 1. This is opposed to *fixed-position models* that allow only a constant number (like 4 or 8) of *label candidates* per point. Van Kreveld et al. compared three fixed-position (namely 1P, 2PH, and 4P in Figure 1) and three slider models (1SH, 2SH, and 4S) with respect to how many more points can be labeled in one model than in another using unit square labels [vKSW99]. Since we are considering labels with equal height but variable length, we need to classify more models. Figure 1 shows all nine fixed-position models and slider models that we consider in this paper. In that figure, each rectangle stands for a feasible label position. An arrow between two rectangles indicates that additionally all label positions are feasible that arise when moving one rectangle on a straight line onto the other. We refer the reader to [vKSW99] for a more formal definition.

For each of their six labeling models, van Kreveld et al. gave approximation algorithms for unit-height labels in the unweighted case. They also did an experimental comparison that showed that algorithms for sliding labels perform especially well on dense point sets such as scatterplots. Other applications with dense point sets include drill holes or electrophoresis gels.

We extend the results of van Kreveld et al. by taking weights into account. More specifically we present the following results. In all but the last section we assume unit-height labels. First, in Section 2, we compare our nine labeling models by giving bounds for the ratios between the weights of maximum-weight labelings in different models. In Section 3, we show how to extend an $O(n \log n)$-time factor-2 approximation algorithm and a PTAS for fixed-position models [AvKS98] to the

weighted case. Both rely on line stabbing, a technique that has already been used successfully for label placement, see [AvKS98, vKSW99]. The idea is to partition the two-dimensional problem into easier one-dimensional subproblems by stabbing the unit-height label candidates of the input points by horizontal lines of at least unit distance such that each label candidate is stabbed. If the resulting subproblems can be solved optimally, then the union of the subsolutions corresponding to either all odd or all even stabbing lines gives a factor-2 approximation for the original problem.

Other than for fixed-position models even the restriction to one dimension turns out to remain NP-hard for slider models. Our proof in Section 4 is by reduction from SubsetSum. This differs from most NP-hardness proofs in the label-placement literature, which are by reduction from 3sat or Planar3sat. For a review, see [vKSW99]. The only similar proof is by reduction from a special case of Partition [GIM+01].

In Section 5 we present the first approximation algorithm for weighted sliding labels, which is the main contribution of this paper besides the NP-hardness proof and the comparison of labeling models. Its approximation factor is $(2+\varepsilon)$, it runs in $O(n^2/\varepsilon)$ time and uses $O(n/\varepsilon)$ space. It is also based on line stabbing. Other than for fixed-position models the one-dimensional problem for sliding labels can only be approximated. For this purpose we use (and improve) a fully polynomial-time approximation scheme [BD00] for single-machine throughput maximization from the job-scheduling literature.

Sections 6 and 7 deal with two restrictions of the one-dimensional problem for sliding labels (i.e. intervals) that can be solved optimally. In Section 6, we consider the case when the number of different weights is bounded and receive a factor-2 approximation for all slider models. In Section 7 we restrict all intervals to unit length and combine the resulting exact one-dimensional algorithm with a dynamic-programming algorithm of Agarwal et al. [AvKS98] to construct a PTAS for labeling points with sliding unit-square labels. In Section 8 we finally drop the restriction on label heights and give algorithms with approximation factors of $3\lceil\log_2 \beta\rceil$ and $(3+\varepsilon)\lceil\log_2 \beta\rceil$ for fixed-position and slider models, respectively, where β is the ratio of maximum and minimum label height. Throughout this paper, we assume that labels are topologically open, i.e. they may touch.

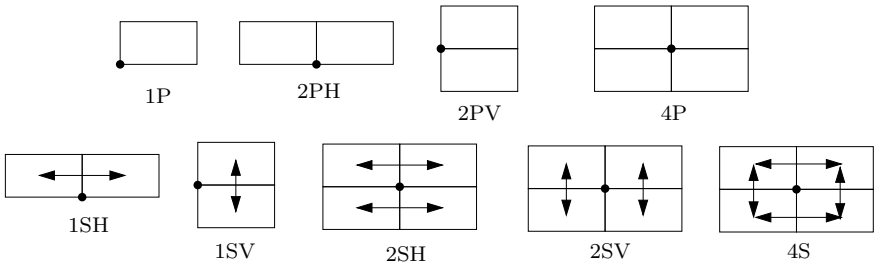

Figure 1: Each model has an abbreviation of the form xMD, where $M \in \{\mathrm{P}, \mathrm{S}\}$ stands for fixed-position model (P) or slider model (S), $x \in \{1, 2, 4\}$ refers to the number of fixed positions or sliding directions, and $D \in \{\emptyset, \mathrm{H}, \mathrm{V}\}$ indicates the horizontal or vertical direction in which fixed-position labels are arranged or labels can slide.

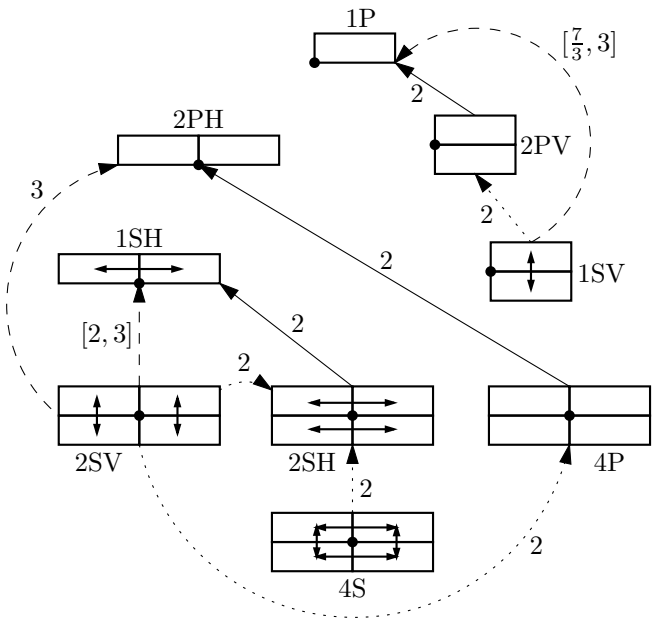

Figure 2: Constant ratios between different labeling models.

2 Comparing labeling models

Given a finite set P of points in the plane, where each point $p \in P$ is associated with a weight $w(p)$, let $W_M(P)$ denote the maximum sum of weights of points whose labels can be placed without intersections given labeling model M. If M_1 and M_2 are two different labeling models from Figure 1, the (M_1, M_2)-ratio is defined as

$$\Psi(M_1, M_2) = \lim_{n\to\infty} \Psi_n(M_1, M_2), \quad \text{where} \quad \Psi_n(M_1, M_2) = \max_{|P|=n} \frac{W_{M_1}(P)}{W_{M_2}(P)}.$$

In order to bound this ratio simultaneously for several pairs of labeling models with similar properties, we use definitions similar to those in [vKSW99].

Definition 1 *Let $v = (0, 1)$ be the unit vector parallel to the y-axis, directed upwards. We say that*

- *M_1 can be* y-flipped *into M_2 if any label position in M_1 that is not allowed in M_2 can be translated by v into a valid label position in M_2,*
- *M_1 can be* one-way y-slid *into M_2 if any label position in M_1 can be translated by σv into a valid label position in M_2 for some $\sigma \in [0, 1]$, and finally*
- *M_1 can be* two-way y-slid *into M_2 if any label position in M_1 can be translated by σv for some $\sigma \in [-1/2, 1/2]$ into a valid label position in M_2 such that a corner of the label coincides with the point to be labeled.*

Our bounds for ratios between different labeling models are summarized in Figures 2 and 4. The numbers that are attached to the arcs between two models M_1 and M_2 give the (M_1, M_2)-ratio; intervals specify lower and upper bounds. In Figure 2 the arcs additionally indicate whether one model can be y-flipped (solid arrows), one-way y-slid (dashed) or two-way y-slid (dotted) into the other. We will bound the ratios between such pairs of models in the next three subsections, and finally

investigate ratios that cannot be bounded by constants. These do not appear in [vKSW99] since there only square labels were taken into account. In the following we drop the prefix "*y*-" that was only meant for distinction with the definitions in [vKSW99].

2.1 Flipping

For models that can be flipped into each other, such as 2PV into 1P and 2SH into 1SH, we have the following result.

Lemma 1 *If M_1 can be flipped into M_2, then $\Psi(M_1, M_2) = 2$.*

Proof. The lower bound comes from the lower-bound example for the unweighted case, see Lemma 7 of [vKSW99].

In order to upper-bound the (M_1, M_2)-ratio, we consider an optimal M_1-labeling and flip each label by $v = (0, 1)$ if it is not valid in M_2. Clearly, neither the flipped labels nor those that have not been flipped intersect other flipped labels. Taking the subset with the larger weight sum yields $\Psi(M_1, M_2) \leq 2$. ◻

2.2 One-way sliding

For models that can be one-way slid into each other, such as 2SV into 2PH and 1SV into 1P, we have the following result.

Lemma 2 *If M_1 can be one-way slid into M_2, then $\Psi(M_1, M_2) \leq 3$.*

Proof. In order to upper-bound the (M_1, M_2)-ratio, we will employ *line stabbing*, a *shifting technique* [HM85] that has been widely used to design approximation algorithms, not exclusively for label placement [AvKS98, vKSW99]. We draw horizontal lines $h_0, \ldots, h_m$ (in top-to-bottom order) with unit spacing over an optimal M_1-labeling such that no line contains a point nor a horizontal edge of an M_1-label. Note that every M_1-label intersects exactly one horizontal line.

For $i \in \{0, 1, 2\}$ let L_i be the set of M_1-labels that intersect h_{i+3j} for any $0 \leq j \leq \lfloor m/3 \rfloor$. Next we translate each M_1-label in L_i into a valid M_2-position by a vector σv without causing intersections with any other labels in L_i as follows. As before, $\sigma \in [0, 1]$ is a real and $v = (0, 1)$ is the upwards pointing unit vector.

Let $l \in L_i$ be an M_1-label intersecting some h_j (without loss of generality $j \geq 2$) and let Δ_j be the open horizontal strip bounded by the two lines h_{j-2} and h_{j+1}. Clearly, $l \in \Delta_j$. Since l intersects h_j, the point of the label l lies below h_{j-1}, and no matter for which $\sigma \in [0, 1]$ the vector σv translates l upwards into a valid M_2-position, l remains below h_{j-2} and thus within Δ_j. Consider an M_1-label $l' \in L_i \backslash \{l\}$ intersecting some $h_{j'}$. If $j = j'$ then moving l and l' vertically will not cause them to intersect. On the other hand if $j \neq j'$ then $\Delta_j \cap \Delta_{j'} = \emptyset$. Since l remains in Δ_j and l' in $\Delta_{j'}$, the translated labels do not intersect in this case either. Thus we can translate all M_1-labels in L_i into a set of non-intersecting M_2-labels. Among the resulting three sets, we take the one with the largest weight sum. This yields $\Psi(M_1, M_2) \leq 3$. ◻

Figure 3 proves the lower bounds given in Figure 2 (see dashed arrows).

2.3 Two-way sliding

In [vKSW99], two-way sliding is called *corner sliding*, and it is shown that $\Psi(M_1, M_2)$ is at most 2 if M_1 can be corner slid into M_2. In fact, the same proof on upper and lower bounds also holds for the weighted case. Thus we have the following lemma.

Lemma 3 *If M_1 can be two-way slid into M_2, then $\Psi(M_1, M_2) = 2$.*

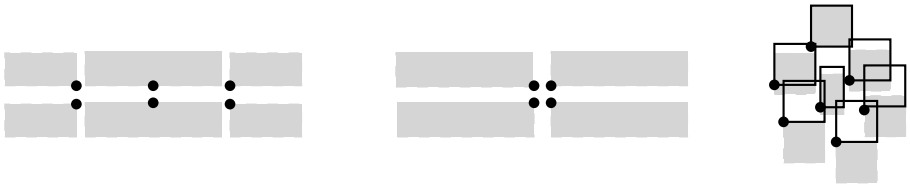

Figure 3: Lower-bound examples for one-way sliding.

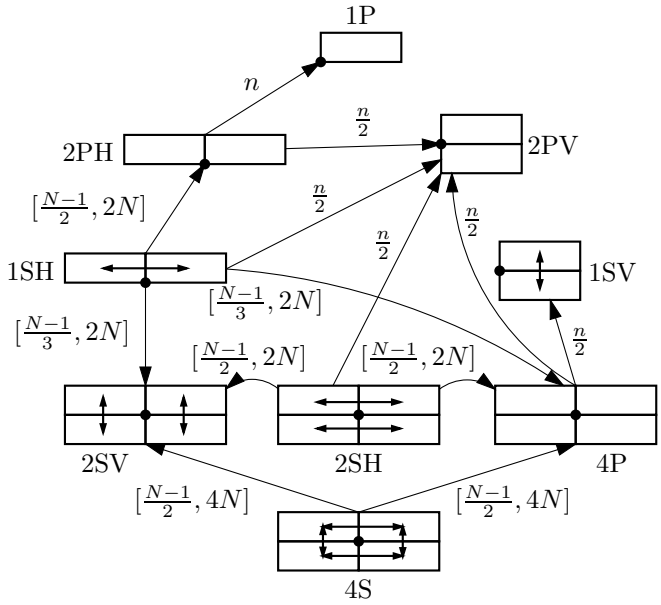

Figure 4: Ratios that cannot be bounded by constants. N is shorthand for $\log n$.

2.4 Other ratios

The three techniques stated above only work when flipping or sliding is performed parallel to the y-axis. The reason is that we require all labels to have the same height. Labels might have different length, so we cannot apply our techniques when flipping or sliding along the x-axis. In fact, in this case there are no constant ratios, see Figure 4. However obtaining sublinear bounds for some ratios is not trivial as we will show in the following.

First, it is easy to show $\Psi_n(2\text{PH}, 1\text{P}) \geq n$ by using labels of different length, say $1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots$ and the corresponding points on the x-axis at $0, 1-\frac{1}{2}, 1-\frac{1}{4}, 1-\frac{1}{8}, \ldots$. Van Kreveld et al. gave this example [vKSW99, Figure 3] as an argument for only considering square labels in their comparison. By placing a copy of the point set just below the original set, we get $\Psi_n(2\text{SH}, 2\text{PV}) \geq n/2$. The matching upper bounds are obvious. With the same arguments, we can achieve all the $n/2$-bounds shown in Figure 4.

To bound $\Psi_n(1\text{SH}, 2\text{PH})$, we consider two one-dimensional labeling problems that correspond to 1SH and 2PH. Given n points on the x-axis, each with a label length and a weight, find a feasible label placement that maximizes the sum of weights of the labeled points. We can interpret these labels as intervals on a line. In analogy to 1SH and 2PH we define two labeling models; a slider model 1d-1SH

where a label can be attached to its point anywhere between its endpoints and a fixed-position model 1d-2PH in which a label must be attached to its point at one of its two endpoints. We have the following result.

Lemma 4 $\frac{1}{2}\log n \leq \Psi_n(\text{1d-1SH}, \text{1d-2PH}) \leq \log n$.

Proof. Let P be a set of n points on the x-axis. For each point $p \in P$ we are given its position on the x-axis $x(p)$, its weight $w(p)$, and the length $\ell(p)$ of its label. If l is the label of p, then l must be placed within a "window" $[r, d]$ on the x-axis where $r = x(p) - \ell(p)$ and $d = x(p) + \ell(p)$. (The choice of the variable names is due to the similarity of our problem to scheduling problems which we exploit again in Section 5. In scheduling, each job has a release time r and a deadline d.)

For the upper bound we assume that $n = 2^k$ for some integer $k > 0$. The main observation that we use repeatedly below is the following. Consider a pair of adjacent labels $l = [a, b]$ and $l' = [a', b']$ of points p and p' in a fixed optimal 1d-1SH-labeling. Let l be to the left of l' (i.e. $b \leq a'$) and assume without loss of generality that l' is not shorter than l. Then the right endpoint d of the window of l must lie in the interval $[a, b']$. As a result, we can move (at least) l within $[a, b']$ to a valid 1d-2PH-position. Now l possibly intersects l', but l does not intersect any other 1d-1SH-labels because the translation is done only within $[a, b']$.

The overall translation is performed in k phases as follows. Let P_0 be the subset of points of P that are labeled in the optimal 1d-1SH-labeling. Number the 1d-1SH-labels from left to right starting at 0, and pair labels with the numbers $2i$ and $(2i+1)$. In phase 1, translate for each pair (at least) one label to a valid 1d-2PH-position as above. Denote by $P_1 \subseteq P_0$ the set of points whose labels have just been translated. Then $W_{\text{1d-1SH}}(P_1) = W_{\text{1d-2PH}}(P_1) \leq W_{\text{1d-2PH}}(P_0)$ and $|P_1| \geq |P_0|/2$. Recursively repeat the same process at phase j with $P_0 \setminus \bigcup_{i=1}^{j-1} P_i$ and set P_j to the subset whose labels are translated. After phase j we have that

$$W_{\text{1d-1SH}}(P_j) = W_{\text{1d-2PH}}(P_j) \leq W_{\text{1d-2PH}}(P_0)$$

and $|P_j| \geq |P_{j-1}|/2$. Due to the second inequality the process terminates after $k = \log n$ phases. Summing up the first inequality yields

$$\sum_{j=1}^{\log n} W_{\text{1d-1SH}}(P_j) \leq \log n \cdot W_{\text{1d-2PH}}(P_0).$$

Since the subsets P_j partition P_0 and P_0 is the subset of P that is labeled in the optimal 1d-1SH-labeling, the left term is equal to $W_{\text{1d-1SH}}(P_0) = W_{\text{1d-1SH}}(P)$. The right term is at most $\log n \cdot W_{\text{1d-2PH}}(P)$ since $P_0 \subseteq P$. Thus $\Psi_n(\text{1d-1SH}, \text{1d-2PH}) \leq \log n$.

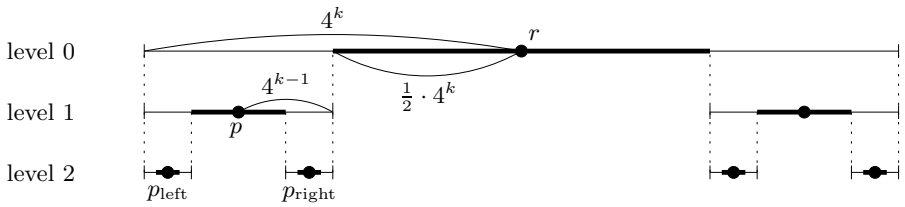

Figure 5: The lower bound construction for $\Psi_n(\text{1d-1SH}, \text{1d-2PH})$. The points are all meant to lie on the x-axis. Their windows are delimited by vertical strokes. The labels of an optimal 1d-1SH-labeling are indicated by the bold line segments.

For the lower bound consider a set P of n points, where we assume n to be $2^k - 1$ for convenience. The construction is similar to the recursive construction

of a complete binary tree of k levels, see Figure 5. At level 0 we place the root r at $x(r) = 2^{2k-1}$. At level i ($0 \le i \le k-1$) we place 2^i points, where each point p has weight $w(p) = 2^{k-i}$ and a label of length $\ell(p) = 4^{k-i}$. If $i < k-1$ then p has two children p_{left} and p_{right} that lie on level $(i+1)$ at $x(p_{\text{left}}) = x(p) - \frac{3}{4}\ell(p)$ and $x(p_{\text{right}}) = x(p) + \frac{3}{4}\ell(p)$. The window of p is $[x(p) - \ell(p), x(p) + \ell(p)]$. An optimal 1d-1SH-labeling labels all points in P by centering the label of each point p within its window, i.e. at $[x(p) - \ell(p)/2, x(p) + \ell(p)/2]$, see the bold line segments in Figure 5. Due to our construction no two labels intersect. Since the weights of the points at each level sum up to 2^k, the total sum is $W_{\text{1d-1SH}}(P) = k2^k \ge n \log n$.

However, any 1d-2PH-labeling can assign a label $l = [a, b]$ to a point p only such that either a or b coincides with $x(p)$. In either way, the points in one of the subtrees of p cannot be labeled because they lie entirely in l. We claim that the weight of an optimal 1d-2PH-labeling is at most $2(2^k - 1) = 2n$, which proves the lower bound. The proof is by induction on k, the number of levels of the tree. If $k = 1$, P consists only of one point whose weight is 2, so the claim clearly holds. Assume that for every tree with $i < k$ levels, the sum of weights of the points labeled is at most $2(2^i - 1)$. Now consider the tree T with k levels. This tree consists of a point at level 0 with weight 2^k and of two subtrees L and R with $k-1$ levels each. The weight $W(T)$ of an optimal 1d-2PH-labeling of T is at most $\max\{\max\{W(L), W(R)\} + 2^k, W(L) + W(R)\}$ because the 1d-2PH-labeling has the choice to assign a label to the point at level 0 or not. By our assumption $\max\{W(L), W(R)\} + 2^k \le 2(2^{k-1}-1) + 2^k = 2(2^k - 1)$ and $W(L) + W(R) \le 2(2^k - 2)$. Thus $W(T) \le 2(2^k - 1)$, which completes the proof. Our proof also shows that exactly one point per level is labeled in the optimal 1d-2PH-labeling. ◻

Lemma 5 $\frac{1}{2} \log n \le \Psi_n(\text{1SH}, \text{2PH}) \le 2 \log n$

Proof. The lower bound is a direct consequence of Lemma 4. The upper bound is obtained by reducing 2PH to two sets of one-dimensional problems with the help of line stabbing as in [vKSW99], and by then applying Lemma 4. ◻

In fact, Lemma 5 even holds for fixed-position models with any finite number of label positions. We now extend the arguments of Lemmas 4 and 5 to prove other $\Theta(\log n)$-bounds. For the sake of briefness we write $\{\text{4P}, \text{2SV}\}$ when we mean that a statement holds for both 4P and 2SV.

Lemma 6 $\frac{1}{2}(\log n) - \frac{1}{2} \le \Psi_n(\text{2SH}, \{\text{4P}, \text{2SV}\}) \le 2 \log n.$

Proof. For the lower bounds we construct an instance P that consists of two point sets with the tree-like structure used in Lemma 4. We place a set T of $n/2 = 2^t - 1$ points on the x-axis and a copy T' slightly above. This means that any 4P- or 2SV-labeling for T and T' cannot do better than 1d-2PH-labeling for T and T'. Thus $W_{\{\text{4P,2SV}\}}(P) = 2 \cdot W_{\text{1d-2PH}}(T) = 2 \cdot 2(2^t - 1) = 2n$. However, the optimal 2SH-labeling can label all points in P, so $W_{\text{2SH}}(P) = 2 \cdot t2^t$. Hence $\Psi_n(\text{2SH}, \{\text{4P}, \text{2SV}\}) = t/2 \cdot 2^t/(2^t - 1) \ge t/2 \ge \frac{1}{2}(\log n) - \frac{1}{2}$.

The upper bounds are achieved by the same argument as in Lemma 5. ◻

Lemma 7 $\frac{1}{3}(\log n) - \frac{1}{3} \le \Psi_n(\text{1SH}, \{\text{4P}, \text{2SV}\}) \le 2 \log n.$

Proof. The upper bound can be obtained as in the proof of Lemma 5. For the lower bound we split our point set P in two equal halves T and T' of $n/2 = 2^t - 1$ points as in the proof of Lemma 6. Again, both have the tree-like structure used in Lemma 4. Other than in Lemma 6, however, we place T' at a distance of 1 above T, see Figure 6. Thus all points can be 1SH-labeled and $W_{\text{1SH}}(P) = t2^{t+1}$.

Now we consider an optimal 4P-labeling. We split the available space into three regions: The space above T', the space below T, and the space between T and T'.

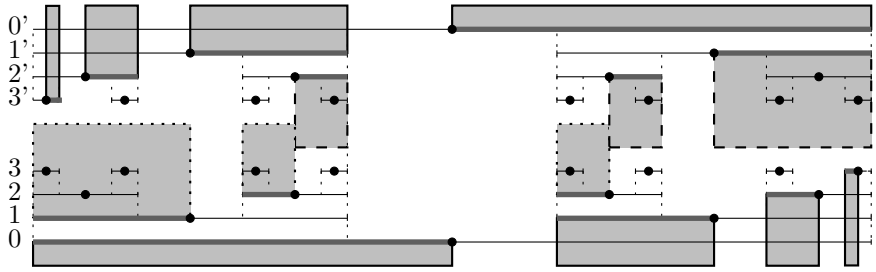

Figure 6: The lower bound construction for $\Psi_n(1\text{SH}, 4\text{P})$. The points in the upper half are meant to lie on the line $y = 1$, those in the lower half on the x-axis. The labels of an optimal 4P-labeling are indicated by rectangles (not drawn to scale).

The weight of the labels of an optimal labeling in each of these three areas is at most the weight of a labeling that is optimal with respect to that area. Lemma 4 says that an optimal labeling for the space above T' and the space below T each has at most weight $2(2^t - 1) \leq 2^{t+1}$. For the space in between we argue as follows. Let L be a label in that area. We claim that the weight of any labeling within L has at most the weight of L. This can be shown by induction over the level of L. By our claim the weight of two labels at level 0 is an upper bound for the weight of an optimal labeling that uses exclusively the space between T and T'. Thus $W_{4\text{P}}(P) \leq 2 \cdot 2^{t+1} + 2 \cdot 2^t = 3 \cdot 2^{t+1}$ and hence $\Psi_n(1\text{SH}, 4\text{P}) \geq t/3 \geq \frac{1}{3}(\log n) - \frac{1}{3}$. The case 2SV is analogous. ◻

Lemma 8 $\frac{1}{2}(\log n) - \frac{1}{2} \leq \Psi_n(4\text{S}, \{4\text{P}, 2\text{SV}\}) \leq 4 \log n$

Proof. The lower bounds come from the same argument as that in Lemma 6 since each 2SH-labeling is also a 4S-labeling. The upper bounds are obtained by first two-way sliding a 4S-labeling into a 2SH-labeling with a factor-2 loss and by then translating the 2SH-labeling as above into 4P- and 2SV-labelings with another loss of a factor of $2 \log n$. ◻

3 An approximation algorithm for fixed-position models

In this section we present approximation algorithms for unit-height labels under all labeling models shown in Figure 1. Our algorithms employ line stabbing, a technique that has been used before to tackle labeling problems with unit-height labels [AvKS98, vKSW99].

Consider the problem of finding a maximum weight independent set (MWIS) of n (topologically open) intervals on the x-axis. This problem is just slightly more general than the one-dimensional version 1d-1P of 1P where input points are assumed to be pairwise different, while this is not necessarily true for the left endpoints of the intervals in the corresponding MWIS problem. MWIS on interval intersection graphs can be solved in $O(n \log n)$ time by a simple dynamic programming algorithm [HTC92]. The one-dimensional version 1d-2PH corresponding to 2PH is as follows. Given a set of n points and for each an interval length, find a MWIS from the $2n$ intervals that are incident to one of the input points. We generally view intervals as topologically open but now make them intersect artificially if they belong to the same point. This can be achieved by a symbolic comparison rule, which allows us to use the above algorithm, although Hsiao et al. assume disjoint interval endpoints [HTC92].

We can generalize 1d-2PH to the problem 1d-kPH in which each input point p has at most k candidate intervals that all contain p. Applying the 1d-1P algorithm to the resulting collection of kn intervals gives rise to:

Lemma 9 *The problem* 1d-kPH *where each input point has at most k candidate intervals can be solved in $O(kn \log n)$ time.*

Combining line stabbing with the above lemma and with dynamic programming as in [AvKS98] yields the following result.

Theorem 1 *Weight maximization given fixed-position models (*1P, 2PH, 2PV*, and* 4P*) can be 2-approximated in $O(n \log n)$ time with linear space. These problems can be $(1 + 1/k)$-approximated in $O(n^{2k-1})$ time and space for any $k \geq 2$.*

Proof. We only sketch our algorithm for 4P, the most general problem among the four problems. Given 4P each point p in P has four label candidates, each of length $\ell(p)$ and unit height. This gives rise to a set R of $4n$ rectangles.

We draw horizontal lines of at least unit distance so that (i) each line intersects at least one rectangle, and (ii) each line contains neither points of P nor bottom and top edges of rectangles. Note that such lines can be drawn in linear time from top to bottom, provided that the y-coordinates of the rectangle edges have been sorted. The set R is partitioned into subsets R_i that consist of all rectangles that intersect line i. By Lemma 9 computing a MWIS for each R_i takes $O(n \log n)$ time in total. Clearly, the solutions for every second line do not intersect. Thus by the pigeonhole principle either the union of the solutions for the odd-numbered or for the even-numbered lines must have at least half the weight of an optimal solution for P.

Note that we never label a point with two of its four label candidates. The reason is that if both belong to the same set R_i, then the symbolic comparison rule for 1d-2PH makes them intersect, otherwise one label candidate belongs to the even and one to the odd lines.

In order to obtain a PTAS, we employ dynamic programming as in [AvKS98] with the only difference that an entry in the dynamic programming table is not the number of labels in the optimal subsolution constructed so far, but the sum of their weights. ◻

The above lemma also yields an $O(kn \log n)$-time factor-2 approximation algorithm for the two-dimensional analog kP of 1d-kPH.

4 The complexity of sliding in one dimension

Other than for fixed-position models even the restriction to one dimension turns out to remain NP-hard for slider models. In this problem, which we refer to as 1d-1SH, all points are given on the x-axis, and labels are (topologically open) intervals that must contain or touch the point they label. The aim is to maximize the weight sum of those points that can be labeled by intervals of the prescribed length such that no two intervals intersect. Our proof is by reduction from SubsetSum. This differs from most NP-hardness proofs in the label-placement literature, which are by reduction from 3Sat or Planar3Sat. For a review, see [vKSW99]. The only similar proof is by reduction from a special case of Partition [GIM+01].

For $P' \subseteq P$ let $w(P')$ be the sum of the weights of the points in P'.

Theorem 2 *The decision version of* 1d-1SH *is NP-complete, i.e.* 1d-1SH *is in $\mathcal{NP}$ and it is NP-hard to decide the following. Given a number W and a set P of points on the x-axis, each with a weight and a label length, is there a subset S of P with*

$w(S) \geq W$ such that each $p \in S$ can be labeled with a sliding label of the given length and no two labels intersect?

Proof. Clearly, 1d-1SH is in $\mathcal{NP}$: we can non-deterministically guess a subset S of P with $w(S) \geq W$ and then deterministically try to label it greedily from left to right in polynomial time.

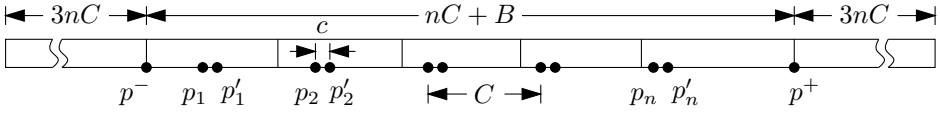

Figure 7: An instance of 1d-1SH to which an instance of SUBSETSUM is reduced.

To show the NP-hardness we reduce SUBSETSUM to 1d-1SH as follows. Given positive numbers $s_1, \ldots, s_n$ and a bound B, the problem SUBSETSUM is to decide whether there is a subset A of $\{1, \ldots, n\}$, such that $\sum_{i \in A} s_i = B$. SUBSETSUM is NP-complete [GJ79]. For our reduction to 1d-1SH we need a very large constant C and a very small constant c, e.g. $C = 1000 \sum_{1 \leq i \leq n} s_i$ and $c = \min_{1 \leq i \leq n} s_i/1000$. Now we construct (in polynomial time) an instance of 1d-1SH with $2n+2$ points as follows. See Figure 7 for an example. Let $P = \{p^-, p^+, p_1, \ldots, p_n, p'_1, \ldots, p'_n\}$, where p^- and p^+ are two "stoppers" with x-coordinates $x(p^-) = 0$ and $x(p^+) = nC + B$, and $p_1, \ldots, p_n, p'_1, \ldots, p'_n$ are $2n$ "normal" points with $x(p_i) = (i - \frac{1}{2})C$ and $x(p'_i) = x(p_i) + c$ for $i = 1, \ldots, n$. The corresponding weights are $w(p^+) = w(p^-) = 3nC$, $w(p_i) = C + s_i$ and $w(p'_i) = C$. Label lengths are equal to point weights. Let $W = 7nC + B$ be the threshold for 1d-1SH and let S be the subset of P with all points that receive a label in a fixed maximum-weight solution of 1d-1SH.

We show that the answer for SUBSETSUM is "yes" (i.e. there is a set $A \subseteq \{1, \ldots, n\}$ such that $\sum_A s_i = B$) if and only if the answer for 1d-1SH is "yes" (i.e. $w(S) \geq W$). We also show that given a 1d-1SH-labeling with weight at least W we can easily construct the subset A for SUBSETSUM.

Recall that in our instance label lengths equal point weights. Since the two stoppers p^- and p^+ must be in S due to their large weights, and the distance between p^- and p^+ is $nC + B$ we have that $w(S) \leq W$ independently of the SUBSETSUM instance.

On the one hand if there is a set $A \subseteq \{1, \ldots, n\}$ such that $\sum_A s_i = B$, then $w(S) \geq W$ since there is a labeling L of weight W, and S corresponds to a *maximum*-weight labeling. The labeling L is constructed as follows: label p^- leftmost, p^+ rightmost and then for $i = 1$ to n (i.e. from left to right) place a label touching its left neighbor to p_i if $i \in A$, otherwise to p'_i. Note that each label is attached to its point close to the label's center since $B \ll C$. The labeling L is "tight", i.e. each label touches its neighbors. Thus in our reduction a yes-instance of SUBSETSUM is reduced to a yes-instance of 1d-1SH.

On the other hand if the reduction yields a yes-instance of 1d-1SH, that is a point set with $w(S) \geq W$, then the original SUBSETSUM instance must have been a yes-instance as well: we claim that the set $A = \{i : p_i \in S\}$ fulfills $\sum_A s_i = B$. Since $w(S)$ is always at most W, we have $w(S) = W$ here. Therefore the weights of the labeled points between p^- and p^+ sum up to exactly $nC + B$. Since $B \ll C$ and $\sum_{1 \leq i \leq n} s_i \ll C$ this is only possible if exactly n of these points are labeled. Among those only the weights of the points of type p_i (whose indices are in A) exceed C. Thus $\sum_A s_i = B$.

This shows that a deterministic polynomial-time algorithm for 1d-1SH could be used to decide the NP-hard problem SUBSETSUM. ◻

Our proof also shows that 1d-1SH remains NP-hard even if we restrict the problem to instances where point weights equal the corresponding label lengths.

5 An approximation algorithm for slider models

Again we first tackle the corresponding one-dimensional problem. Other than in the case of fixed-position models we cannot hope to solve 1d-1SH exactly in polynomial time. Thus we are looking for an approximation algorithm or, even better, for a fast (i.e. fully polynomial-time) approximation scheme. Due to its close relationship to 1d-1SH we now state a scheduling problem, namely *single-machine throughput maximization*: Given a collection $\mathcal{J}$ of n *jobs* $J_1, \ldots, J_n$, each with a weight w_i, a *processing time* (or job length) ℓ_i, a *release time* r_i and a *deadline* d_i, find a schedule that maximizes the *throughput* on a single machine, i.e. find a maximum-weight subset $\mathcal{J}'$ of the jobs and for each job $J_i \in \mathcal{J}'$ an open interval I_i of length ℓ_i that is contained in the *execution window* $[r_i, d_i]$ of J_i such that no two intervals intersect.

Berman and DasGupta [BD00] have given a two-phase algorithm, ε-2PA, for single-machine throughput maximization if the maximum ratio between the size of the execution window and the processing time is bounded, i.e. if the so-called *stretch factor* $\alpha = \max_i\{(d_i - r_i)/\ell_i\}$ is bounded. Their algorithm has an approximation factor of $2/(1 + 1/(2^{\lfloor\alpha\rfloor+1} - 2 - \lfloor\alpha\rfloor)) + \varepsilon$ for any $\varepsilon > 0$ and runs in $O(n^2/\varepsilon)$ time. In the case $\alpha = 2$ this yields a factor-$(8/5 + \varepsilon)$ approximation. However, using a symbolic comparison rule as in Lemma 9, we get the same approximation factor as for $1 < \alpha < 2$, i.e. $(1 + \varepsilon)$.

Their algorithm uses $O(n^2/\varepsilon)$ storage. We simplify their algorithm and show how the storage consumption can be reduced to $O(n/\varepsilon)$ for $\alpha \leq 2$ assuming the above-mentioned symbolic comparison which ensures that all intervals of the same job intersect. In phase I, the *evaluation phase*, ε-2PA discretizes the problem depending on ε and on the job weights w_i. Intervals are generated in order of non-decreasing right endpoint and are put on a stack $\mathcal{S}$. In phase II, the *selection phase*, the intervals are successively taken off the stack and either put into the solution if they do not intersect any other interval there, or discarded otherwise.

The main idea behind the discretization is a *value* $v_I = w_I - \sum_{I' \cap I \neq \emptyset, I' \in \mathcal{S}} v_{I'}$ that is attributed to each interval I when it is pushed on $\mathcal{S}$. The weight w_I of I is the weight of the job to which I belongs. Phase I of ε-2PA consists of nothing but repeatedly determining the interval $I^\star$ whose right endpoint is leftmost among all intervals I with $v_I \geq \varepsilon w_I$ and then pushing $I^\star$ on $\mathcal{S}$. For an example, see Figure 8. Note that the number of intervals on $\mathcal{S}$ per job varies greatly.

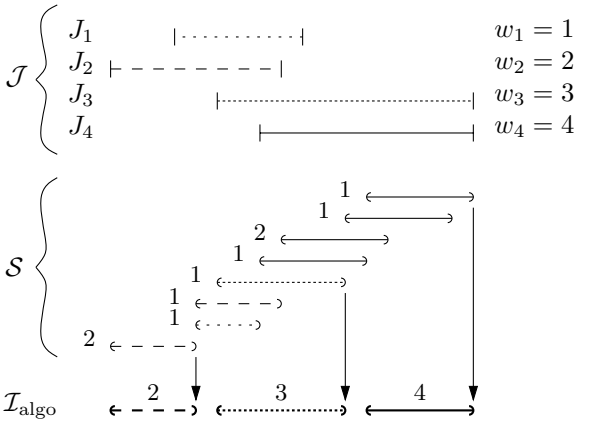

Figure 8: An example instance $\mathcal{J}$ with $\alpha = 2$ and $\varepsilon = 1$, the stack $\mathcal{S}$ after phase I of ε-2PA with the values of each interval, and the solution $\mathcal{I}_{\text{algo}}$ with the weights of the selected intervals.

Lemma 10 ([BD00]) *Given a collection $\mathcal{J}$ of n weighted jobs with stretch factor $\alpha \leq 2$, ε-2PA finds a schedule whose weight sum is at least $(1-\varepsilon)$ times the maximum weight.*

Proof. A proof of the approximation factor for any α can be found in [BD00]. We now give a much simpler proof for $\alpha \leq 2$ using our notation. Let S be the set of intervals on the stack $\mathcal{S}$ right after the end of phase I of the algorithm. For a subset S' of S, let $V(S')$ be the sum of the values of the intervals in S'. The proof consists of two steps. We first show that the weight sum W_{algo} of the intervals in the algorithm's solution $\mathcal{I}_{\text{algo}}$ equals $V(S)$. We then show that $V(S) > (1-\varepsilon)W_{\text{opt}}$, where W_{opt} is the weight sum of a fixed optimal solution $\mathcal{I}_{\text{opt}}$.

To see $W_{\text{algo}} = V(S)$ recall that the intervals are pushed on $\mathcal{S}$ in order of non-decreasing right endpoint. We put the intervals in S in groups as follows. The first group G_1 consists of the leftmost interval I_1 in $\mathcal{I}_{\text{algo}}$ and all intervals that have been pushed on $\mathcal{S}$ before I_1. The next group G_2 consists of the interval I_2 to the right of I_1 in $\mathcal{I}_{\text{algo}}$ and all intervals on $\mathcal{S}$ between I_1 and I_2, and so on. Note that each interval on $\mathcal{S}$ belongs to a group since the topmost interval on $\mathcal{S}$ is in $\mathcal{I}_{\text{algo}}$. The last interval I_i of group G_i always belongs to $\mathcal{I}_{\text{algo}}$. Recall that the value v_i of I_i equals the weight w_i of the corresponding job minus the values of all intervals on $\mathcal{S}$ below I_i that intersect I_i. Due to the way $\mathcal{I}_{\text{algo}}$ is constructed in phase II of the algorithm the first interval on $\mathcal{S}$ below I_i that does not intersect I_i is I_{i-1}. Thus $v_i = w_i - V(G_i \setminus \{I_i\})$. This means that $w_i = V(G_i)$. Summing up these equalities over all I_i in $\mathcal{I}_{\text{algo}}$ yields $W_{\text{algo}} = V(S)$.

To show $V(S) > (1-\varepsilon)W_{\text{opt}}$ we group the intervals on $\mathcal{S}$ differently. For each interval $I = (b, e)$ in $\mathcal{I}_{\text{opt}}$ the group G_I now contains all intervals on $\mathcal{S}$ whose right endpoint lies in $(b, e]$. Clearly, all groups are mutually exclusive, but their union is not necessarily S. If $I \in G_I$ then we have $V(G_I) \geq w_I$ as above. Otherwise consider the point of time $t^\star$ in phase I just before the first interval is pushed on $\mathcal{S}$ whose right endpoint lies to the right of I. At that moment I was not pushed on $\mathcal{S}$ since the threshold $v_I \geq \varepsilon w_I$ on its value was violated. Note that I would have been attributed the value $v_I = w_I - V(G_I)$ since G_I consists of all intervals on $\mathcal{S}$ at time $t^\star$ that intersect I. Thus $V(G_I) > (1-\varepsilon)w_I$, which yields $V(S) > (1-\varepsilon)W_{\text{opt}}$ after summing up over all I in $\mathcal{I}_{\text{opt}}$. ⊡

Lemma 11 *The algorithm ε-2PA can be implemented to run in $O(n^2/\varepsilon)$ time using $O(n/\varepsilon)$ storage.*

Proof. The threshold $v_I \geq \varepsilon w_I$ on the value v_I of an interval I to be pushed on $\mathcal{S}$ ensures that at most $1/\varepsilon$ intervals of one job are pushed on $\mathcal{S}$ since they all intersect each other in our case. Thus there are at most n/ε intervals on $\mathcal{S}$ at the end of phase I.

In order to determine the next interval to be pushed on $\mathcal{S}$ in phase I we maintain for each job J_i four parameters v_i, b_i, e_i, and s_i. The number v_i is the value that would be attributed to the interval $I_i = (b_i, e_i)$ of J_i that is in the leftmost position satisfying $v_i \geq \varepsilon w_i$. The pointer s_i refers to an interval on $\mathcal{S}$ whose right endpoint coincides with the left endpoint $e_i - l_i$ of I_i if such an interval exists. Otherwise $s_i = 0$.

Since $\mathcal{S}$ is empty in the beginning, we initially have $v_i = w_i$, $b_i = r_i$, $e_i = b_i + l_i$, and $s_i = 0$. We repeatedly push a copy of the interval $I_i = (b_i, e_i)$ on $\mathcal{S}$ whose right endpoint is leftmost, i.e. $e_i = \min\{e_1, \ldots, e_n\}$. After pushing I_i we must update our parameters for all intervals I_j in our data structure (not in $\mathcal{S}$!) that intersect I_i. For these we reduce v_j by v_i. If this results in $v_j < \varepsilon w_j$, we move I_j to the right as follows. First, if $s_j = 0$ we set s_j to the last interval on $\mathcal{S}$ that does not intersect I_j. Second, we repeatedly move s_j to the next interval I on $\mathcal{S}$, set the left

endpoint b_j of I_j to the right endpoint of I, set the right endpoint e_j of I_j to $b_j + l_j$, and increase v_j by the value of I since I does not intersect I_j any more. We keep moving s_j until either $v_j \geq \varepsilon w_j$ or $e_j > d_j$. In the latter case we do not have to consider job J_j any more. For an example, see Figure 9.

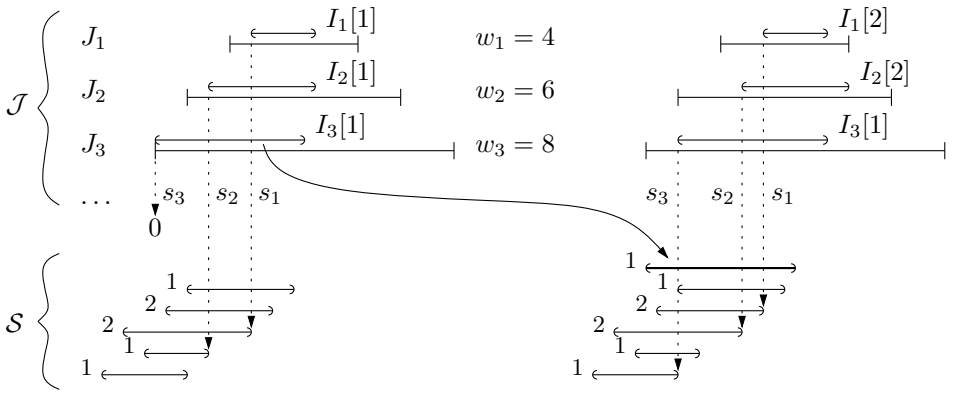

Figure 9: Our data structure during phase I of ε-2PA before and after a copy of I_3 is pushed on $\mathcal{S}$. For each interval I_i the number v_i is given in square brackets. Again $\alpha = 2$ and $\varepsilon = 1$. Note that not all intervals on $\mathcal{S}$ belong to jobs J_1 to J_3.

Clearly, each parameter can be maintained in time proportional to the size of $\mathcal{S}$ right before phase II, and the interval to be pushed on $\mathcal{S}$ can be determined in $O(n)$ time. Thus phase I takes $O(n^2/\varepsilon)$ time and uses $O(n/\varepsilon)$ space. Time and space consumptions of phase II are subsumed by those of phase I. ◻

The problem 1d-1SH is very closely related to throughput maximization with stretch factor 2: for each input point p_i of the labeling problem, we define a job J_i by setting its weight to that of p_i, its length to the interval length $\ell(p_i)$ of p_i, and its execution window to $[x(p_i) - \ell(p_i), x(p_i) + \ell(p_i)]$. Then the length of the execution window of each job is exactly twice the job length, i.e. $\alpha = 2$. Note that 1d-1SH and the scheduling problem are not completely equivalent. 1d-1SH is slightly more restrictive in that the input points are required to come from a set, i.e. they are pairwise different, while the corresponding midpoints of the job intervals can be arbitrary.

Theorem 3 *Weight maximization given slider models (*1SH, 1SV, 2SH, 2SV, 4S*) can be $(2+\varepsilon)$-approximated in $O(n^2/\varepsilon)$ time using $O(n/\varepsilon)$ space.*

Proof. Again we only consider the most general model 4S. For each input point p, we define two axis-parallel *candidate rectangles* of length $2\ell(p)$ and unit height. The upper (lower) rectangle touches p in the midpoint of its bottom (top) edge. A label of p is entirely contained in the union of its upper and lower rectangle. Let R denote the set of the resulting $2n$ rectangles.

As in Theorem 1, we draw horizontal lines of unit distance over R so that (i) each line intersects at least one rectangle, and (ii) each line contains neither points of P nor bottom and top edges of rectangles. Again R is partitioned into subsets R_i that consist of all rectangles that intersect line i. Let $W(L)$ be the weight sum of a labeling L. We use ε-2PA to compute a labeling L_i for each R_i. Lemma 10 guarantees that $W(L_i) \geq W_i/(1+\varepsilon/2)$, where W_i is the weight of a maximum-weight labeling of R_i. We denote by L_{even} (L_{odd}) the union of the sets L_i with i even (odd). We return a labeling with weight $\max\{W(L_{\text{even}}), W(L_{\text{odd}})\}$.

We will now prove that $W(L^\star) \leq (2+\varepsilon)\max\{W(L_{\text{even}}), W(L_{\text{odd}})\}$, where $L^\star$ is an optimal 4S-labeling for P. Partition $L^\star$ into three subsets $L^\star_{\text{even}}, L^\star_{\text{odd}}$, and $L^\star_{\text{mid}}$,

where $L^\star_{\text{even}}$ ($L^\star_{\text{odd}}$) consists of those rectangles in $L^\star$ that properly intersects an even (odd) line, and $L^\star_{\text{mid}}$ contains all rectangles in $L^\star$ that touch two consecutive lines with their top and bottom edges. Note that $L^\star_{\text{mid}}$ is empty in slider models that do not allow vertical sliding. Clearly, $W(L^\star_{\text{even}}) \leq (1+\varepsilon/2)W(L_{\text{even}})$ and $W(L^\star_{\text{odd}}) \leq (1+\varepsilon/2)W(L_{\text{odd}})$. We show that there is an optimal labeling $L^\star$ with $L^\star_{\text{mid}} = \emptyset$, which proves our claim.

Suppose to the contrary that there is no optimal labeling L' with $L'_{\text{mid}} = \emptyset$. Let $L^\star$ be any optimal labeling. Let G be an intersection graph whose nodes represent labels of $L^\star_{\text{mid}}$. Two nodes in G are adjacent if the corresponding labels touch. G is partitioned into a collection of maximally connected components. Consider a set Q that consists of the labels of some maximally connected component of G. Let δ be the maximum distance by which all labels in Q can be moved vertically without intersecting labels in $L^\star \setminus Q$. If $\delta > 0$, then we can move all labels in Q by a sufficiently small amount and get an optimal labeling L' with $L'_{\text{mid}} = \emptyset$.

If $\delta = 0$, then there must be a path $\Pi = (l_1, \ldots, l_k)$ in G such that the point p_1 of the topmost label l_1 lies on the bottom edge of l_1 and the point p_k of the bottommost label l_k lies on the top edge of l_k. Suppose that such a path does not exist. Then there are two labels in $L^\star \setminus Q$ that prevent us from moving Q vertically. These labels must touch some labels in Q, which is a contradiction to Q being a maximal connected component of G. Hence p_1 and p_k lie on the edges of l_1 and $l_k \in L^\star_{\text{mid}}$, and thus on two horizontal lines. This contradicts the way we placed the lines. ◻

6 An exact algorithm for a bounded number of different weights

The following two sections deal with two restrictions of the problem 1d-1SH that can be solved optimally. In this section we consider the case when the number of different weights is bounded. We state our result in the language of scheduling.

Lemma 12 *Let $\mathcal{J}$ be a collection of n jobs $J_1, \ldots, J_n$, where job J_i has weight w_i, release time r_i, deadline d_i and length l_i. Let k be the number of different job weights and let V_k be the number of possible throughputs. There is an algorithm that computes a schedule for $\mathcal{J}$ with maximum throughput in $O(nV_k)$ time using $O(V_k)$ storage if the stretch factor α of $\mathcal{J}$ is less than 2.*

Note that V_k is always at least n. Though in general $V_k \in O(n^k)$ we have that $V_k = nk$ in the interesting special case where the weights are the first k integers. In this case throughput maximization with $\alpha < 2$ can be solved in $O(n^2)$ time (considering k a constant). We do not know how to relax the restriction $\alpha < 2$ to $\alpha \leq 2$. If $a < 2$ then the order of the jobs is the same in all possible schedules.

If nothing is known about the distribution of the weights, the price for exactness is high: the runtime then becomes $O(n^{k+1})$ compared with $O(n^2/\varepsilon)$ for a factor-$(1+\varepsilon)$ approximation.

Proof. To simplify the presentation we allow a throughput of 0 and assume it is counted by V_k. First we construct a sorted list L with one entry for each possible throughput in n steps as follows. Start with $L = (0)$. Suppose that after i steps L is sorted and that it contains all throughputs that can be generated with the weights $w_1, \ldots, w_i$ for some $i < n$. In step $i+1$ simply make a copy L' of L, add w_{i+1} to each entry in L' and merge L' into L. Since the length of L is bounded by V_k, the n steps take $O(nV_k)$ time in total.

Now we use dynamic programming with a table T of size V_k. There is an entry $T[v]$ for each possible throughput $v \in L$ that stores the finish time of the

leftmost schedule with throughput v. The leftmost schedule with throughput v is the schedule that has the earliest finish time among all schedules with throughput v. We fill the table in order of increasing throughput. Initially all entries have value $-\infty$. We compute $T[v]$ for $v > 0$ as follows.

For each job we check whether $v - w_i \geq 0$ and if so, whether J_i can be scheduled to the right of $T[v - w_i]$. If yes, we schedule J_i as early as possible, i.e. we set its finish time t_i to $\max\{r_i, T[v - w_i]\} + l_i$. If no, we set t_i to $+\infty$. Finally we set $T[v] := \min\{t_1, \ldots, t_n\}$ to the earliest possible finish time.

The maximum throughput $v_{\max}$ of $\mathcal{J}$ is the largest v for which $T[v] < \infty$. The corresponding schedule s can be computed by using an additional table X of size V_k. An entry $X[v]$ of X stores the index of the last job that has been scheduled when computing $T[v]$. Let $i = X[v_{\max}]$. Then s consists of job J_i scheduled at $(T[v_{\max}]-l_i, T[v_{\max}])$ and the jobs that can be computed recursively by investigating $X[v_{\max} - w_i]$.

In order to analyze the running time of our algorithm, observe that for each job J_i we have to do at most V_k look-ups in T. The indices of the entries that we look up for J_i increase monotonically while we fill T. Thus it is enough to maintain for each of the k different weights w_i a pointer that always points to $T[v - w_i]$. The maintenance of the pointers takes $O(kV_k)$ time in total. With this data structure each entry $T[v]$ can be computed in $O(n)$ time, thus we have a total time complexity of $O(nV_k)$.

The proof of correctness is by induction over the throughput. Let $v > 0$ be a possible throughput value. Assume that all entries of T for values smaller than v are correct. We have to show that the finish time $T[v]$ which our algorithm computes in fact corresponds to the schedule with the earliest finish time among all schedules with throughput v. First observe that our algorithm computes only legal schedules. This is due to the fact that no two intervals overlap and the stretch factor α is less than 2; thus no two intervals of the same job can be scheduled. Let s be a schedule of throughput v whose last job is J_i. Then we know that the finish time of $s \setminus \{J_i\}$ is at least $T[v - w_i]$ by our induction hypothesis. Since our algorithm tried all jobs including J_i as the last job of the schedule for v, the finish time $T[v]$ is at least as early as that of s. ⊡

Interestingly enough we can ignore the restriction $\alpha < 2$ when using the above scheduling algorithm for point labeling with sliding labels. This is due to the fact that 1d-1SH and single-machine throughput maximization are not completely equivalent. 1d-1SH is slightly more restrictive in that the input points are required to come from a set, i.e. they are pairwise different, while the corresponding midpoints of the job intervals can be arbitrary.

Lemma 13 *Given a set $P = \{p_1, \ldots, p_n\}$ of points on the x-axis, each with a weight w_i and a label length l_i,* 1d-1SH *can be solved in $O(nV_k)$ time using $O(V_k)$ space, where k is the number of different point weights and V_k is the number of possible weight sums.*

Proof. We first convert our 1d-1SH-instance into a scheduling problem by setting $r_i = p_i - l_i$ and $d_i = p_i + l_i$, and then apply a modified version of the dynamic programming algorithm given in the proof of Lemma 12. When the original algorithm computes $T[v]$, the algorithm checks for each job J_i whether (a) $v - w_i \geq 0$ and (b) $T[v - w_i] \leq d_i - l_i$, and then chooses the job that can be placed leftmost, i.e. for which $t_i = \max\{r_i, T[v - w_i]\} + l_i$ is minimum. For 1d-1SH (with stretch factor 2) we additionally have to check whether (c) $X[v - w_i] \neq i$, i.e. whether point p_i has not already received a label. We also have to modify the way in which we choose the job that is scheduled. Let a job with $t_i = r_i + l_i$ be an *early* job, otherwise a *late* job. If there are several jobs with t_i minimum, we choose a late job since an early

job can still be scheduled later; right after the current job. In the case of 1d-1SH the points $p_i = r_i + l_i$ are unique, thus there is at most one early job and we can always choose a late job if there is any choice at all.

The correctness of the modified algorithm can be shown similarly to that of the original algorithm in the proof of Lemma 12: while so far $\alpha < 2$ ensured that no jobs are scheduled twice, this is now guaranteed by checking condition (c). ◻

Combining Lemma 13 with line stabbing as in the proof of Theorem 3 yields factor-2 approximation algorithms for all slider models:

Theorem 4 *Weight maximization given slider models can be 2-approximated in $O(nV_k)$ time using $O(V_k)$ space, where k is the number of different point weights and V_k is the number of possible weight sums.*

7 An approximation scheme for unit-square labels

This section deals with a special case of the problem 1d-1SH where all intervals have unit length. This corresponds to labeling points with unit squares. We address this special case since the more general problem of designing a PTAS for unit-height rectangles seems to be difficult in the weighted case, and is solved in the unweighted case [vKSW99].

The idea of our algorithm for 1d-1SH for unit-length intervals is to discretize the continuous space of label positions of each point to a small number of label candidates such that each optimal solution of the continuous problem corresponds to a solution of the discrete problem that has the same weight. Then Lemma 9 solves the problem.

The algorithm is as follows. Sort the n different input points from left to right and denote them by $p_1, p_2, \ldots, p_n$ in this order. Clearly, p_1 can do with only one label candidate, namely its leftmost, $[x_1 - \ell_1, x_1]$. For p_i $(i > 1)$ we also take its leftmost candidate but additionally all the endpoints of the candidates of p_{i-1} that fall into the label window $[x_i - \ell_i, x_i + \ell_i]$ of p_i. Note that other than in the general case at most *one* of the two endpoints can do that for each candidate of p_{i-1}. Intuitively speaking, we do not have to worry about the candidates of points p_j with $j < i - 1$ since their endpoints either do not fall into the window of p_i or, if they do, they also fall into that of p_{i-1} and thus will be taken into account. Hence p_i has at most i candidates. Lemma 9 yields

Lemma 14 *For unit-length intervals the problem* 1d-1SH *can be solved in $O(n^2 \log n)$ time using $O(n^2)$ space.*

Proof. Take any optimal solution of the continuous version of 1d-1SH. Consider its leftmost label interval l_1. Either l_1 is in its leftmost position or we can push it there without intersecting any other label. The next label l_2 is either a label candidate of our algorithm (if l_2 touches l_1 or is in its leftmost position) or we can slide it to the left until it reaches such a position, again without intersecting other labels. By repeating this procedure for all intervals of the optimal solution we obtain an equivalent solution that uses merely label candidates which our algorithm computes. Thus our algorithm finds a solution that is also optimal in the continuous case. ◻

Combining the above discretization for 1d-1SH with line stabbing and the dynamic-programming algorithm of Agarwal et al. [AvKS98] gives us a PTAS for labeling points with sliding unit-square labels.

Corollary 1 *Given a set P of n points and an integer $k \geq 1$ there is an algorithm that finds a* 1S*-labeling for P whose weight is at least $k/(k+1)$ times the maximum weight. The algorithm takes $O(n^{4k-2})$ time and uses $O(n^{4k-2})$ space.*

Proof. In [AvKS98] Agarwal et al. give a PTAS for our labeling model 1P given unit-height rectangles without weights. Their algorithm runs in $O(n^{2k-1} + n \log n)$ time and uses $O(n^{2k-1})$ space. They use line stabbing with horizontal lines of unit distance to partition the input rectangles into sets $R_1, \ldots, R_m$. Then they use dynamic programming to find the maximum independent set of rectangles in each block B_i of k consecutive lines $R_i, \ldots, R_{i+k-1 \bmod m}$ for $i = 1, \ldots, m$. They consider the $k+1$ different solutions for the original problem that arise by dropping the rectangles on every $(k+1)$th line and joining the solutions obtained for the blocks in between. They argue that by the pigeonhole principle one of these solutions must have placed at least $k/(k+1)$ times the maximum number of non-intersecting rectangles.

The same algorithm can be used for the weighted case by storing the sum of the weights of the selected rectangles in the dynamic programming table instead of simply their number. After projecting all squares in a block B_i on the x-axis, we can use our algorithm for 1d-1SH to compute $O(b_i^2)$ label candidates for the b_i points that correspond to the squares in B_i. Analogously to Lemma 14 we can argue that there is a solution of the resulting discrete problem with the same weight as the maximum-weight solution of the 1S-problem for B_i. Thus applying the dynamic programming algorithm of Agarwal et al. to the discrete set of label candidates for B_i yields an optimal 1S-solution for B_i. The size of the dynamic programming table becomes $O(b_i^{2(2k-1)})$ since we have $O(b_i^2)$ label candidates. The time needed to fill the table is also $O(b_i^{2(2k-1)})$. Thus computing maximum-weight solutions for all m blocks takes $O(n^{4k-2})$ time in total, and since b_i can be linear in n, the dynamic-programming table can have size $O(n^{4k-2})$. ◻

8 An approximation algorithm for instances with bounded height ratio

In this section, we label points with weighted sliding labels whose heights may vary, but only within a constant factor. For each input point p in P we are given its label length $\ell(p)$ and height $h(p)$. Let β be the ratio of maximum and minimum label height, i.e. $\beta = \max_{p \in P} h(p) / \min_{p \in P} h(p)$. Usually a map or diagram uses only a small number of different fonts whose sizes do not vary too much, thus β is relatively small in practice and it is worthwhile designing an algorithm whose approximation factor depends on β.

For the case of fixed-position models and arbitrary label heights, algorithms for (weighted) maximum independent set in rectangle intersection graphs can be used. Agarwal et al. achieve an approximation factor of $O(\log n)$ in the unweighted case [AvKS98], Iturriaga explains how the ideas of Agarwal et al. can be extended to handle weighted rectangles as well [Itu99]. Recently Erlebach et al. have improved this result for weighted squares by giving a PTAS [EJS01].

Strijk and van Kreveld [SvK02] presented a practical factor-$(1+\beta)$ approximation algorithm for labeling *unweighted* points with sliding labels. Their algorithm takes $O(rn \log n)$ time, r being the number of different label heights. We present a new approximation algorithm for the weighted case. Its runtime is independent of r and its approximation factor is better than that of [SvK02] for $\beta > 11$.

Theorem 5 *Let P be a set of n points, each with a label, and let β be the ratio of maximum to minimum height among these labels. Then the maximum-weight labeling for P can be $3\lceil \log_2 \beta \rceil$-approximated in $O(kn \log n)$ time given a fixed-position model with at most k positions per point and $(3+\varepsilon)\lceil \log_2 \beta \rceil$-approximated in $O(n^2/\varepsilon)$ time for slider models.*

Proof. We normalize all label heights to the interval $[1, \beta]$. First we partition P into $m = \lceil \log_2 \beta \rceil \geq 1$ groups P_j such that $P_j = \{p \in P \mid 2^{j-1} \leq h(p) < 2^j\}$ for $1 \leq j < m$ and P_m contains the remaining points. Let L_j (L) denote a maximum-weight labeling for P_j (P) and W_j (W) its weight. By the pigeon-hole principle there is a labeling L_j whose weight W_j is at least W/m. Now we combine line stabbing with the 1d-algorithms of Sections 3 and 5 to compute a labeling of weight at least $W_j/3$ or $W_j/(3+\varepsilon)$ for each P_j.

We show how to approximate the most general problem 4S; the same method works for the other labeling models. As in Theorem 3, define $2n$ candidate rectangles of P_j, denoted by R, and draw horizontal lines so that (i) two consecutive lines are apart by 2^j units, and (ii) each line contains neither points of P_j nor top or bottom edges of rectangles in R. (For ease of presentation we ignore the fact that rectangles might be very far apart vertically.) Let R_i be the set of candidate rectangles intersecting line i. Then a (near-) optimal solution for R_i can be obtained by our algorithms for the corresponding one-dimensional problems, see Lemma 9 and ε-2PA, respectively. We now consider candidate rectangles that lie completely in the horizontal strip Δ between the lines i and $(i+1)$. Since the height of these rectangles is at least 2^{j-1} and the distance of the lines is 2^j, there must be a horizontal line in Δ that stabs all of the rectangles. This implies that again we can compute a (near-) optimal solution for the rectangles inside Δ by Lemma 9 or ε-2PA, respectively. Clearly, the partial solutions we compute for rectangles stabbed by even-numbered lines are independent, similarly for odd-numbered lines and for the inter-line strips. Thus one of these three solutions is at least of weight $W_j/3$ under fixed-position models and $W_j/(3+\varepsilon)$ under slider models. ◻

Conclusions

We have presented a number of fixed-position and slider models for labeling points with axis-parallel rectangles. In the case of unit-height rectangles we have given factor-2 and factor-$(2+\varepsilon)$ approximation algorithms for maximizing the weight sum of those points that receive a label under fixed-position and slider models, respectively. These algorithms use line stabbing and (near-) optimal algorithms for the corresponding one-dimensional label- (or rather interval-) placement problems. In the case of fixed-position models the 1d-problem is a special case of maximum weight independent set, in the case of slider models the 1d-problem is a special scheduling problem, namely single-machine throughput maximization. While there is a fully polynomial-time approximation scheme for the latter problem [BD00], we showed that the decision version is in fact NP-hard. Our reduction from subset sum is the first such proof in the label-placement literature. We simplified the approximation scheme and improved its space complexity.

We have investigated two special cases of label-placement problems with sliding labels where the one-dimensional version can be solved optimally: if all intervals are of equal length or if the number of different weights is bounded. Finally we have presented an algorithm for arbitrary-height labels whose approximation factor depends logarithmically on the ratio of maximum to minimum label height.

An interesting open question is whether there is a PTAS for sliding unit-height labels as in the unweighted case [vKSW99]. We have given such a scheme for unit-square labels.

Acknowledgments

We thank Otfried Cheong and Kyung-Yong Chwa for giving us a chance to discuss this problem at HKUST and KAIST, respectively. We thank Jae-Hoon Kim for contributing an idea to the proof of Theorem 5.

References

[AF84] John Ahn and Herbert Freeman. AUTONAP—an expert system for automatic map name placement. In *Proc. International Symposium on Spatial Data Handling (SDH'84)*, pages 544–569, 1984.

[AvKS98] Pankaj K. Agarwal, Marc van Kreveld, and Subhash Suri. Label placement by maximum independent set in rectangles. *Computational Geometry: Theory and Applications*, 11:209–218, 1998.

[BD00] Piotr Berman and Bhaskar DasGupta. Multi-phase algorithms for throughput maximization for real-time scheduling. *Journal of Combinatorial Optimization*, 4(3):307–323, September 2000.

[Cc99] Bernard 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. American Mathematical Society, Providence, RI, 1999.

[CMS95] Jon Christensen, Joe Marks, and Stuart Shieber. An empirical study of algorithms for point-feature label placement. *ACM Transactions on Graphics*, 14(3):203–232, 1995.

[EJS01] Thomas Erlebach, Klaus Jansen, and Eike Seidel. Polynomial-time approximation schemes for geometric graphs. In *Proc. 12th ACM-SIAM Symposium on Discrete Algorithms (SODA'01)*, pages 671–679, Washington, DC, 7–9 January 2001.

[FW91] Michael Formann and Frank Wagner. A packing problem with applications to lettering of maps. In *Proc. 7th Annuual ACM Symposium on Computational Geometry (SoCG'91)*, pages 281–288, 1991.

[GIM+01] Mari Ángeles Garrido, Claudia Iturriaga, Alberto Márquez, José Ramon Portillo, Pedro Reyes, and Alexander Wolff. Labeling subway lines. In Peter Eades and Tadao Takaoka, editors, *Proc. 12th Annual International Symposium on Algorithms and Computation (ISAAC'01)*, volume 2223 of *Lecture Notes in Computer Science*, pages 649–659, Christchurch, 19–21 December 2001. Springer-Verlag.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979.

[Hir82] Stephen A. Hirsch. An algorithm for automatic name placement around point data. *The American Cartographer*, 9(1):5–17, 1982.

[HM85] Dorit S. Hochbaum and Wolfgang Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM*, 32:130–136, 1985.

[HTC92] Ju Yuan Hsiao, Chuan Yi Tang, and Ruay Shiung Chang. An efficient algorithm for finding a maximum weight 2-independent set on interval graphs. *Information Processing Letters*, 43(5):229–235, October 1992.

[Itu99] Claudia Iturriaga. *Map Labeling Problems*. PhD thesis, University of Waterloo, 1999.

[Mor80] Joel L. Morrison. Computer technology and cartographic change. In D.R.F. Taylor, editor, *The Computer in Contemporary Cartography*. Johns Hopkins University Press, Baltimore, MD, 1980.

[PSSW01] Sheung-Hung Poon, Chan-Su Shin, Tycho Strijk, and Alexander Wolff. Labeling points with weights. In Peter Eades and Tadao Takaoka, editors, *Proc. 12th Annual International Symposium on Algorithms and Computation (ISAAC'01)*, volume 2223 of *Lecture Notes in Computer Science*, pages 610–622, Christchurch, 19–21 December 2001. Springer-Verlag.

[SvK02] Tycho Strijk and Marc van Kreveld. Practical extensions of point labeling in the slider model. *GeoInformatica*, 6(2):181–197, 2002.

[vKSW99] Marc van Kreveld, Tycho Strijk, and Alexander Wolff. Point labeling with sliding labels. *Computational Geometry: Theory and Applications*, 13:21–47, 1999.

[WS96] Alexander Wolff and Tycho Strijk. The Map-Labeling Bibliography. `http://i11www.ira.uka.de/map-labeling/bibliography/`, 1996.

[Zor90] Steven Zoraster. The solution of large 0-1 integer programming problems encountered in automated cartography. *Operations Research*, 38(5):752–759, 1990.