Labeling Points with Weights

Sheung-Hung Poon* Chan-Su Shin† Tycho Strijk‡
Alexander Wolff§

May 25, 2001

Ernst-Moritz-Arndt-Universität Greifswald
Institut für Mathematik und Informatik
Preprint-Reihe Mathematik 7/2001

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.

Next we describe a factor-2 approximation for the case that the number of different weights is bounded and a PTAS for labeling points with sliding unit-square labels. Finally, for instances with a constant ratio β of maximum to minimum label height we give algorithms with approximation factors of $3\lceil \log_2 \beta \rceil$ and $(3+\varepsilon)\lceil \log_2 \beta \rceil$ assuming fixed-position and slider models, respectively.

Keywords

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

*Dept. of Comp. Science, HKUST, Hong Kong, hung@cs.ust.hk
†Dept. of Comp. Science, KAIST, Korea, cssin@jupiter.kaist.ac.kr
‡Dept. of Comp. Science, Utrecht University, The Netherlands, tycho@cs.ruu.nl
§Inst. für Mathematik und Inf., Universität Greifswald, Germany, awolff@uni-greifswald.de

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 computational complexity of the label-placement problem, cartographers, graph drawers, and computational geometers have suggested numerous approaches, such as expert systems [AF84], zero-one integer programming [Zor90], approximation algorithms [AvKS98, SvK99, vKSW99], simulated annealing [CMS95] and force-driven algorithms [Hir82] to name only a few. An extensive bibliography about label placement can be found at [WS96]. The ACM Computational Geometry Impact Task Force report [Cc99] denotes 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.

This paper deals with one of the most common label-placement problems, namely labeling points with axis-parallel rectangles. Many publications on this problem can be found in [WS96]. However, with two exceptions this is the first paper that gives approximation algorithms for points with weights, which is extremely important for practical applications. 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. for maximum-independent set on rectangle-intersection graphs [AvKS98] can be extended to handle weighted rectangles as well (n is the number of rectangles here). Second, Erlebach et al. recently improved these results for squares; they give a polynomial-time approximation scheme (PTAS) for the weighted case [EJS01].

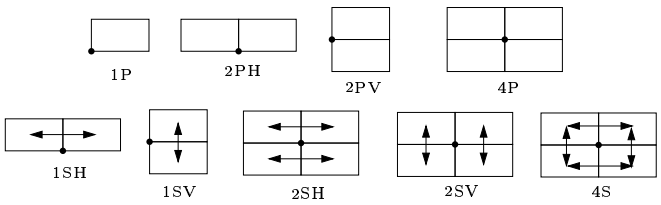


Figure 1: Each model has an abbreviation of the form xMD where $M \in \{\mathrm{P}, \mathrm{S}\}$ stands for fixed-position model (P) or slider model (S), $x \in \{1, 2\}$ refers to the number of fixed positions or sliding directions, and $D \in \{\emptyset, \mathrm{H}, \mathrm{V}\}$ indicates the horizontal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

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 will consider in this paper. In that figure, each rectangle stands for a feasible label position. An arrow between two rectangle indicates that additionally all label position 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. The main contribution of this paper besides the comparison of labeling models is the first approximation 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. The algorithms for both the fixed-position and the slider models use line stabbing, a technique that has already been used successfully for label placement [AvKS98, vKSW99], to partition the two-dimensional problem into easier one-dimensional subproblems and then solve these subproblems (near-) optimally.

Section 4 and 5 deal with two restrictions of the one-dimensional problem for sliding labels (i.e. intervals) that can be solved optimally. In Section 4, we consider the case that the number of different weights is bounded and receive a factor-2 approximation for all slider models. In Section 5, 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 6, 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.

2 Comparing labeling models

Let M_1 and M_2 be any two different labeling models from Figure 1. 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. Then the (M_1, M_2)-ratio is defined as

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

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 be a unit vector parallel to the y-axis.*

M_1 can be y-flipped *into M_2 by v 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 by v if any label position in M_1 can be translated by σv into a valid label position in M_2 for some $\sigma \in [0, 1]$.*

M_1 can be two-way y-slid *into M_2 by v if any label position in M_1 can be translated by σv for some $\sigma \in [-1, 1]$ 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 3. The numbers that are attached to the arcs between two models

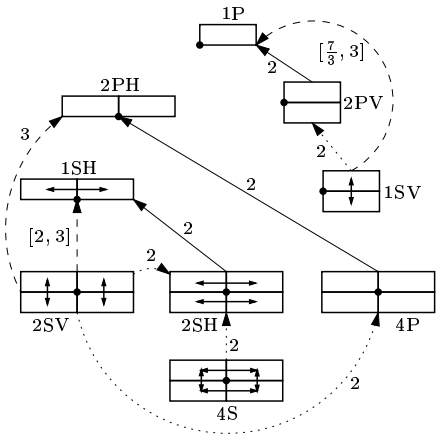


Figure 2: Constant ratios between different labeling models.

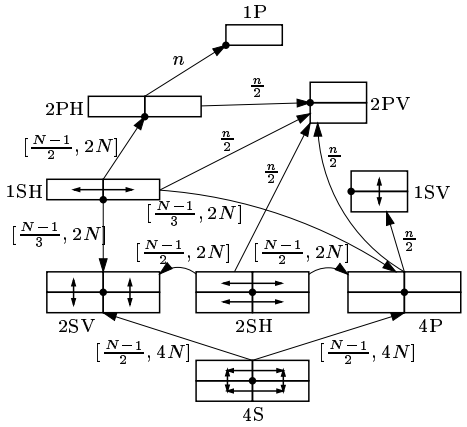


Figure 3: Ratios that cannot be bounded by constants. N is shorthand for $\log n$.

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 (bold arrows), one-way y-slid (dashed) or two-way y-slid (dotted) into the other. Note that two-way y-sliding is more restrictive than one-way y-sliding. 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 will 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 [vKSW99, Lemma 7].

Wlog., we assume that the flipping vector v is $(0, 1)$. In order to upper-bound the (M_1, M_2)-ratio, we consider an optimal M_1-labeling and flip each label by v if it is not valid in M_2. Clearly neither the flipped labels intersect each other nor do those that have not been flipped. 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.

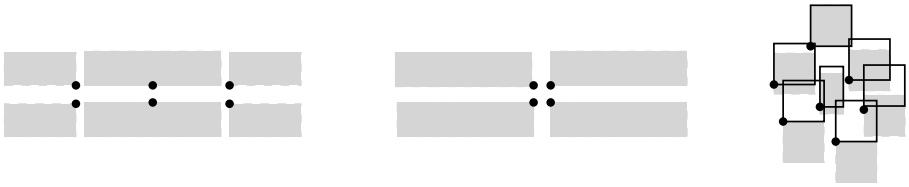
Lemma 2 *If M_1 can be one-way y-slid into M_2, then $\Psi(M_1, M_2) \leq 3$.*

Proof. Again we only consider $v = (0, 1)$, that is, M_1 can be one-way slid into M_2 by moving an M_1-label upwards into a valid M_2-position. In order to upper-bound the (M_1, M_2)-ratio, we will employ *line stabbing*, a *shifting technique* that has been widely used to design approximation algorithms not only for label placement [HM85,

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 where $\sigma \in [0,1]$ without causing intersections with any other labels in L_i as follows.

Let $l \in L_i$ be an M_1-label intersecting some h_j (wlog. $j \geq 2$) 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 $l' \in L_i \setminus \{l\}$ 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 $\Psi(M_1, M_2) \leq 3$. The lower bounds are shown in Figure 4. ☐



(a) (b) (c)

Figure 4: Lower-bound examples for (a) $\Psi(2\text{SV}, 2\text{PH}) \geq 3$ (b) $\Psi(2\text{SV}, 1\text{SH}) \geq 2$, and (c) $\Psi(1\text{SV}, 1\text{P}) \geq \frac{7}{3}$

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 $\Psi(M_1, M_2) = 2$.*

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 3. However obtaining sublinear bounds for some ratios (e.g. $\Psi(1\text{SH}, 2\text{PH}) \in \Theta(\log n)$) is not trivial as we will show in the following.

First of all, it is easy to show $\Psi(2\text{PH}, 1\text{P}) \geq n$ 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 $\Psi(2\text{SH}, 2\text{PV}) \geq n/2$. The matching upper

bounds are obvious. With the same arguments, we can achieve all the $n/2$-bounds shown in Figure 3.

To bound $\Psi(\text{1SH}, \text{2PH})$, 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 $\frac{1}{2}\log n \leq \Psi(\text{1d-1SH}, \text{1d-2PH}) \leq \log n$.

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 will exploit again in Section 3. 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 will 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' and assume wlog. 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. The label l will possibly intersect l' but no 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 $\Psi(\text{1d-1SH}, \text{1d-2PH}) \leq \log n$.

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 $(0 \leq i \leq k-1)$ 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 $i < k-1$ than p has two children p_{left} and p_{right} that lie on level $(i+1)$ at $x(p_{\text{left}}) = x(p) - \frac{3}{4}\ell(p)$

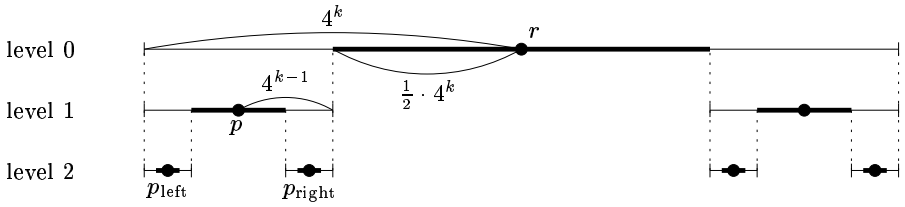


Figure 5: The lower bound construction for $\Psi(\text{1d-1SH}, \text{1d-2PH})$. The points are all meant to lie on the x-axis. Their windows are delimited by vertical strokes. The labels of an optimal 1d-1SH-labeling are indicated by the bold line segments.

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 $W_{\text{1d-1SH}}(P) = k2^k \geq n \log n$.

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 $\max\{W(L), W(R)\} + 2^k \leq 2(2^{k-1} - 1) + 2^k = 2(2^k - 1)$ and $W(L) + W(R) \leq 2(2^k - 2)$. Thus $W(T) \leq 2(2^k - 1)$, which completes the proof. Our proof also shows that exactly one point per level is labeled in the optimal 1d-2PH-labeling. ◻

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

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 will now extend the arguments of Lemma 4 and 5 to prove other $\Theta(\log n)$-bounds. For the sake of briefness we will write $\{\text{4P}, \text{2SV}\}$ when we mean that a statement holds for both 4P and 2SV.

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

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 $\Psi(\text{2SH}, \{\text{4P}, \text{2SV}\}) = t/2 \cdot 2^t/(2^t - 1) \geq t/2 \geq \frac{1}{2}(\log n) - \frac{1}{2}$.

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

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

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 $W_{1\mathrm{SH}}(P) = t2^{t+1}$.

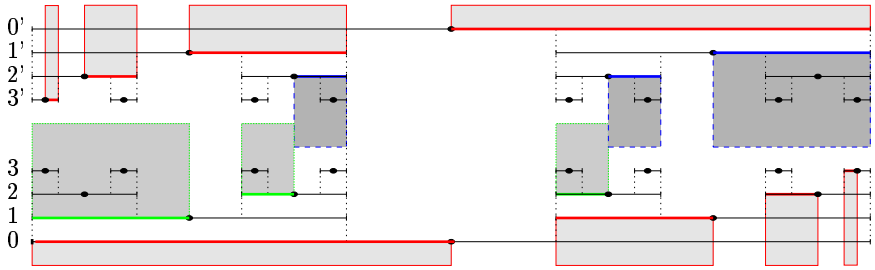


Figure 6: The lower bound construction for $\Psi(1\mathrm{SH}, 4\mathrm{P})$. The points in the upper half are meant to lie on the line $y = 1$, those in the lower half on the x-axis. The labels of an optimal 4P-labeling are indicated by rectangles (not drawn to scale).

Now let us 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'. 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 $W_{4\mathrm{P}}(P) \leq 2 \cdot 2^{t+1} + 2 \cdot 2^t = 3 \cdot 2^{t+1}$ and hence $\Psi(1\mathrm{SH}, 4\mathrm{P}) \geq t/3 \geq \frac{1}{3}(\log n) - \frac{1}{3}$. The case 2SV is analogous. ◻

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

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 Approximation algorithms for unit-height labels

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].

We first consider the problem *1d-1P* of finding a maximum weight independent set (MWIS) of n (topologically open) intervals on the x-axis. The problem is exactly the one-dimensional version of 1P, and it 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 either start or end at 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 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 *The weighted fixed-position labeling problems* 1P, 2PH, 2PV, *and* 4P *can be 2-approximated in $O(n \log n)$ time with linear space. These problems can be $(1 + \frac{1}{k})$-approximated in $O(n^{2k-1})$ time and space for any $k \geq 2$.*

Proof. We will 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 height 1. This gives rise to a set R of $4n$ rectangles.

We draw unit-distant horizontal lines 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 pigeon-hole 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.

In what follows, we consider approximation algorithms for problems with sliding labels. Again we first tackle the corresponding one-dimensional problem. Given a set of n points, each with a weight and an interval length, the problem 1d-1SH consists of maximizing the weight sum of those points that can be labeled by intervals of the prescribed length such that each interval contains its point and no two intervals intersect. Due to its close relationship to 1d-1SH we now state a scheduling problem, namely *single-machine throughput maximization*: Given a set $\mathcal{J}$ 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 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.

Lemma 10 *The single-machine throughput-maximization problem for a set $\mathcal{J}$ of n jobs can be approximated within a factor of $(1 + \epsilon)$ in $O(n^2/\epsilon)$ time using $O(n/\varepsilon)$ storage if the* stretch factor $\alpha = \max_i\{(d_i - r_i)/\ell_i\}$ *of $\mathcal{J}$ is at most 2.*

Proof. In [BD00], Berman and DasGupta present a two-phase algorithm, ε-2PA, for single-machine throughput maximization with bounded stretch factor α. Their algorithm has an approximation factor of

$$\frac{2}{1+\frac{1}{2^{\lfloor\alpha\rfloor+1}-2-\lfloor\alpha\rfloor}}+\epsilon,$$

for any $\epsilon > 0$ and runs in $O(n^2/\varepsilon)$ time. In the case $\alpha = 2$ this yields a factor-$(5/3+\epsilon)$ approximation. However, using a symbolic comparison rule as in Lemma 9, we get the same approximation factor as for $1 < \alpha < 2$, i.e. $(1+\epsilon)$.

Their algorithm uses $O(n^2/\varepsilon)$ storage. We will now show how this 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 its first phase, 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 x-value of their right endpoint and are put on a stack S. In the second phase, 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* $v_I = w_I - \sum_{I' \cap I \neq \emptyset, I' \in S} v_{I'}$ that is attributed to each interval I when it is pushed on S. The weight w_I of I equals the weight of the job to which I belongs. Phase I of ε-2PA consists of nothing but repeatedly determining the interval I^* whose right endpoint is leftmost among all intervals I with $v_I \geq \varepsilon w_I$ and pushing I^* on S. This ensures that $|S| \leq n/\varepsilon$ at the end of phase I. For the proof of correctness, see [BD00].

In order to determine I^* we maintain a monotonically decreasing staircase function f that maps x to the sum of the values of all intervals on S that intersect $[x, \infty[$. For each job J_i with $(1-\varepsilon)w_i \leq \max f$ we maintain a marker at height $h_i = (1-\varepsilon)w_i$ on f. For such a job let $b_i = f^{-1}(h_i)$, otherwise $b_i = r_i$. Now I^* is the interval $]b_j, e_j[$ of job J_j where $e_j = b_j + \ell_j$ is at most d_j and e_j is minimum among all jobs. By construction $v_{I^*} = w_j - f(b_j) \geq w_j - h_j = \varepsilon w_j$. After pushing I^* on S, a new stair of height v_{I^*} and length e_j is attached to f from below and all markers are moved downwards by v_{I^*}. Phase I terminates when $e_i > d_i$ for all jobs J_i. Since f consists of $|S| \leq n/\varepsilon$ stairs at the end of phase I, f can be maintained in $O(n/\varepsilon)$ time and space. The function f can be stored as a list of numbers $e_{I_1}, v_{I_1}, e_{I_2}, v_{I_2}, \ldots, e_{I_{|S|}}, v_{I_{|S|}}$, where $I_k =]b_{I_k}, e_{I_k}[$ is the k-th interval counted from the bottom of S and v_{I_k} its value. There are n markers and each must climb down at most $|S|$ stairs, which takes $O(n^2/\varepsilon)$ time in total. Their minimum can then simply be updated in $O(n)$ time whenever a new interval is pushed on the stack, i.e. $|S|$ times. Thus phase I takes $O(n^2/\varepsilon)$ total time and uses $O(n/\varepsilon)$ space. ◻

Throughput maximization with a stretch factor of 2 is equivalent to 1d-1SH: 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, so we can solve 1d-1SH near-optimally.

Theorem 2 *The weighted sliding problems* 1SH, 1SV, 2SH, 2SV, *and* 4S *can be* $(2+\varepsilon)$*-approximated in* $O(n^2/\varepsilon)$ *time using* $O(n/\varepsilon)$ *space.*

Proof. Again we only go into the most general problem 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 compute a labeling L_i for each R_i according to Lemma 10. This 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 a 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. ◻

4 An exact scheduling 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 that the number of different weights is bounded. We state our result in the language of scheduling.

Theorem 3 *Let $\mathcal{J}$ be a set of n jobs $J_1, \ldots, J_n$. If the stretch factor α of $\mathcal{J}$ is less than 2, the number of different weights is k, and $V_k = O(n^k)$ is the number of possible throughputs, then there is an algorithm that computes a schedule with maximum throughput in $O(nV_k \log V_k)$ time using $O(V_k)$ storage.*

Note that $V_k = nk$ if the weights are the first k integers. If additionally k is considered a constant, throughput maximization can be solved in $O(n^2 \log n)$ time. The same holds for 1d-1SH with the slight restriction that the intervals cannot use the full window for sliding, but only its interior. Thus we receive factor-2 approximation algorithms for all (in the above sense restricted) slider models as in the proof of Theorem 2. We do not know how to relax the restriction $\alpha < 2$ to $\alpha \leq 2$.

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

Proof. We use dynamic programming with a table T of size $V_k + 1$. There is an entry $T[v]$ for each possible throughput v that stores the finish time of the *leftmost schedule* with throughput v. $T[0]$ is a dummy entry. The leftmost schedule with throughput v is the schedule that has the earliest finish time among all schedules with throughput v.

First we build a binary tree over all possible throughputs. The leaves are linked to the entries of the dynamic programming table. We fill the table in order of increasing throughput. Initially all entries have value $-\infty$. We compute $T[v]$ as follows.

A job J_i is given by its weight w_i, release time r_i, deadline d_i and length l_i. For each job we check whether $w_i \leq v$ 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. If at least one job among $J_1, \ldots, J_n$ is scheduled, we set $T[v]$ to the earliest finish time among these at most n schedules, otherwise to $+\infty$.

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 entry $L[v]$ that stored the index of the last job that has been scheduled when computing $T[v]$. Let $i = L[v_{\max}]$. Then s consists of job J_i scheduled at $(T[v] - l_i, T[v])$ and the jobs that can be computed recursively by investigating $T[v_{\max} - w_i]$.

The running time is $O(nV_k \log V_k)$ since for each of the V_k entries and for each of the n jobs we have to do at most one look-up in the binary tree over T, and each look-up takes $O(\log V_k)$ time.

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 $T[v]$ our algorithm computes in fact corresponds to the schedule with the earliest finish time among all schedules with weight 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 weight 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 last job of the schedule s_{algo} for v, the finish time $T[v]$ of s_{algo} is at least as early as that of s. ∎

5 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 11 *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 $\frac{k}{k+1}$ 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 pigeon-hole principle one of these solutions must have placed at least $\frac{k}{k+1}$ 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 11 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 $O(n^{4k-2})$. ◻

6 An approximation algorithm for labeling 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 to design 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 [SvK99] presented a practical factor-$(1+\beta)$ approximation algorithm for labeling *unweighted* points with sliding labels. Their algorithm takes $O(rn \log n)$ time, r 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 [SvK99] for $\beta > 11$.

Theorem 4 *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 Section 3 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 2, 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 10. Let us 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 10. 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. ∎

7 Conclusion

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-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. We have investigated two special cases where the one-dimensional problem 1d-1SH for sliding labels (i.e. intervals) can also 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.

However we were neither able to show the NP-hardness of 1d-1SH nor did we find a general optimal algorithm for 1d-1SH—which exists for 1d-kPH. Another interesting 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.

References

[AF84] John Ahn and Herbert Freeman. AUTONAP—an expert system for automatic map name placement. In *Proceedings International Symposium on Spatial Data Handling*, 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, 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 *Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms (SODA'01)*, pages 671–679, Washington, DC, 7–9 January 2001.

[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.

[SvK99] Tycho Strijk and Marc van Kreveld. Practical extensions of point labeling in the slider model. In *Proc. 7th ACM Symposium on Advances in Geographic Information Systems*, pages 47–52, 5–6 November 1999.

[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://www.math-inf.uni-greifswald.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.