Labeling Points with Weights

Sheung-Hung Poon[1], Chan-Su Shin[2], Tycho Strijk[3], and Alexander Wolff[4]

[1] Dept. of Comp. Science, HKUST, Hong Kong, hung@cs.ust.hk
[2] School of Electronics and Information Engineering,
Hankuk University of Foreign Studies, Korea, cssin@hufs.ac.kr
[3] Dept. of Comp. Science, Utrecht University, The Netherlands, tycho@cs.ruu.nl
[4] Institut für Mathematik und Informatik, Universität Greifswald, Germany,
awolff@uni-greifswald.de

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 also investigate some special cases.

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 [6], cartographers, graph drawers, and computational geometers have suggested numerous approaches. Several heuristic methods have been analyzed experimentally [4]. An extensive bibliography about label placement can be found at [13]. The ACM Computational Geometry Impact Task Force report [3] denotes label placement as an important research area.

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 [8] showed how a factor-$O(\log n)$ approximation algorithm of Agarwal et al. [1] 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. [5] 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 [12]. 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 [12]. 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 [12] 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 $O(n \log n)$-

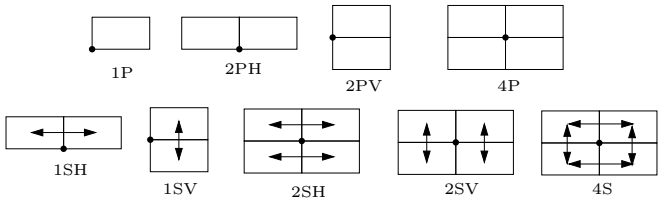

Fig. 1. Each model has an abbreviation of the form xMD where $M \in \{\mathrm{P}, \mathrm{S}\}$ stands for fixed-position model (P) or slider model (S), $x \in \{1, 2, 4\}$ refers to the number of fixed positions or sliding directions, and $D \in \{\emptyset, \mathrm{H}, \mathrm{V}\}$ indicates the horizontal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

time factor-2 approximation algorithm and a PTAS for fixed-position models [1] 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 [1, 12]. 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 (near-optimally), then the union of the subsolutions corresponding to either all odd or all even stabbing lines gives a factor-2 $(2+\varepsilon)$ approximation for the original problem.

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. [1] 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. Due to space constraints we had to sketch some of the proofs. For the details we refer the reader to the full paper [10].

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

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 M_1 and M_2 give the (M_1, M_2)-ratio; intervals specify lower and upper bounds. The proofs for the constant ratios are similar to those in [12], those for linear ratios are simple. For both we refer the reader to the full paper [10]. Instead we investigate the more interesting logarithmic ratios. These do not appear in [12] since there only square labels were taken into account.

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

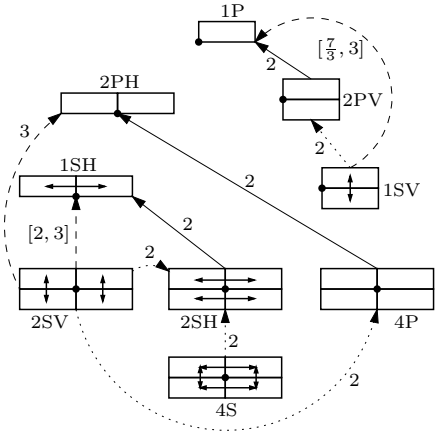

Fig. 2. Constant ratios between different labeling models.

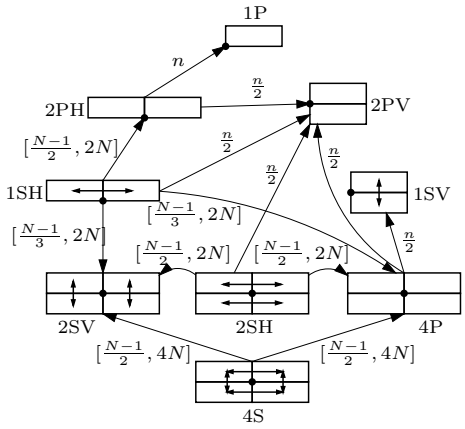

Fig. 3. Ratios that cannot be bounded by constants. N is shorthand for $\log n$.

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

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 Ψ(1d-1SH, 1d-2PH) $\leq \log n$.

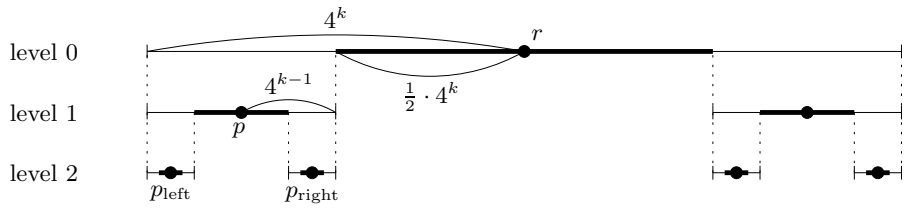

Fig. 4. The lower bound construction for Ψ(1d-1SH, 1d-2PH). The points are all meant to lie on the x-axis. Their windows are delimited by vertical strokes. The labels of an optimal 1d-1SH-labeling are indicated by the bold line segments.

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 4. 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 $i < k-1$ 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 4. 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 2 $\frac{1}{2} \log n \leq \Psi(\text{1SH}, \text{2PH}) \leq 2 \log n$

Proof. The lower bound is a direct consequence of Lemma 1. The upper bound is obtained by reducing 2PH to two sets of one-dimensional problems with the help of line stabbing as in [12], and by then applying Lemma 1. ◻

In fact, Lemma 2 even holds for fixed-position models with any finite number of label positions. We will now extend the arguments of Lemma 1 and 2 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 3 $\frac{1}{2}(\log n) - \frac{1}{2} \leq \Psi(\text{2SH}, \{\text{4P}, \text{2SV}\}) \leq 2 \log n.$

Proof. For the lower bounds we construct an instance P that consists of two point sets with the tree-like structure used in Lemma 1. 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 2. ◻

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

Proof. The upper bound can be obtained as in the proof of Lemma 2. 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 3. Again, both have the tree-like structure used in Lemma 1. Other than in Lemma 3, however, we place T' at a distance of 1 above T. Thus all points can be 1SH-labeled and $W_{\text{1SH}}(P) = t2^{t+1}$.

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 1 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 5 $\frac{1}{2}(\log n) - \frac{1}{2} \leq \Psi(4\mathrm{S}, \{4\mathrm{P}, 2\mathrm{SV}\}) \leq 4 \log n$

Proof. The lower bounds come from the same argument as that in Lemma 3 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 [1, 12].

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 [7]. 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 [7].

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 6 *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 [1] 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 $k \geq 2$.*

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 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.

Lemma 7 *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 [2], 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 $2/(1+\frac{1}{2^{\lfloor\alpha\rfloor+1}-2-\lfloor\alpha\rfloor})+\epsilon$ for any $\epsilon > 0$ and runs in $O(n^2/\varepsilon)$ time. In the case $\alpha = 2$ this yields a factor-$(8/5+\epsilon)$ approximation. However, using a symbolic comparison rule as in Lemma 6, 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 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 x-value of their right endpoint and are put on a stack 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* $v_I = w_I - \sum_{I' \cap I \neq \emptyset, I' \in S} v_{I'}$ that is attributed to each interval I when it is pushed on 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^* whose right endpoint is leftmost among all intervals I with $v_I \geq \varepsilon w_I$ and then pushing I^* on S. The threshold for v_I ensures that at most $1/\varepsilon$ intervals of one job are pushed on S since they all intersect each other in our case. Thus $|S| \leq n/\varepsilon$ at the end of phase I.

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, ∞). For each job $h_i = (1-\varepsilon)w_i$ is the difference between weight and threshold. If $h_i > \max f$ let $b_i = r_i$. Otherwise maintain a marker at height h_i "on" f and its projection b_i on the x-axis as in Figure 5. Let $e_i = h_i + \ell_i$. Note that for each job, (b_i, e_i) is the leftmost interval whose value is above the threshold. Now

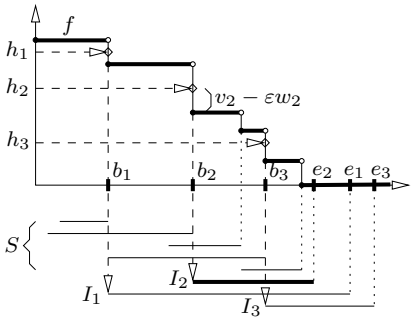

Fig. 5. Stack S and staircase function f.

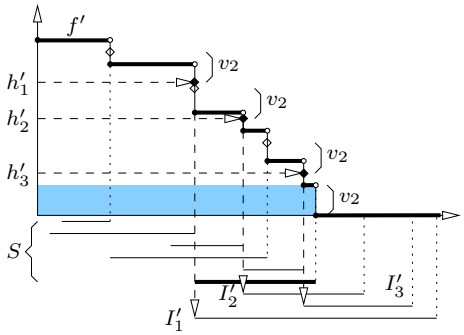

Fig. 6. After pushing I^* on the stack.

I^* is the interval (b_j, e_j) of job J_j with e_j minimum among all e_i with $e_i \leq d_i$. In Figure 5 $I^* = I_2$. By construction $v_j = w_j - f(b_j) \geq w_j - h_j = \varepsilon w_j$. After pushing I^* on S, a new stair (shaded in Figure 6) of height v_j and length e_j (measured from origin) is attached to f from below and all markers are moved downwards by v_j, see Figure 6. 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 simply as a list of numbers $e^1, v^1, e^2, v^2, \ldots, e^{|S|}, v^{|S|}$, where $I^k = (b^k, e^k)$ is the k-th interval counted from the bottom of S and v^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. The minimum over the e_i can 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. Since the right endpoints of the intervals on S are non-decreasing, phase II takes only constant time per interval, i.e. $O(n/\varepsilon)$ time in total. For the proof of the approximation factor, see [2]. ◻

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.*

The difficulty in proving Theorem 2 comes only into play with vertical sliding. Even then, however, line stabbing and ε-2PA can be applied without modification. It is only the analysis that becomes more involved, see [10].

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 $\alpha \leq 2$.

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, see [10]. ∎

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 [12].

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 6 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 6 yields

Lemma 8 *For unit-length intervals the problem* 1d-1SH *can be solved in* $O(n^2 \log n)$ *time using* $O(n^2)$ *space.*

The proof is elementary, see [10]. Combining the above discretization for 1d-1SH with line stabbing and the dynamic-programming algorithm of Agarwal et al. [1] 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.*

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) MIS in rectangle intersection graphs can be used. Agarwal et al. achieve an approximation factor of $O(\log n)$ in the unweighted case [1], Iturriaga explains how the ideas of Agarwal et al. can be extended to handle weighted rectangles as well [8]. Recently Erlebach et al. have improved this result for weighted squares by giving a PTAS [5].

Strijk and van Kreveld [11] 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 [11] for $\beta > 11$.

The idea is to partition P into $\lceil \log_2 \beta \rceil$ groups such that label heights within a group differ at most by a factor of 2. By the pigeon-hole principle there is a group whose maximum-weight labeling W is at least $1/\lceil \log_2 \beta \rceil$ times the maximum weight of a labeling for P. We combine line stabbing with the 1d-algorithms of Section 3 to compute a labeling of weight at least $W/3$ or $W/(3+\varepsilon)$ for each P_j. For the details refer to [10].

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.*

Acknowledgments

We thank Otfried Cheong and Kyung-Yong Chwa to give 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 4.

References

1. P. K. Agarwal, M. van Kreveld, and S. Suri. Label placement by maximum independent set in rectangles. *Comp. Geom.: Theory and Appl.*, 11:209–218, 1998.
2. P. Berman and B. DasGupta. Multi-phase algorithms for throughput maximization for real-time scheduling. *J. Comb. Optimization*, 4(3):307–323, Sept. 2000.
3. B. Chazelle and 36 co-authors. The computational geometry impact task force report. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, volume 223, pages 407–463. AMS, 1999.
4. J. Christensen, J. Marks, and S. Shieber. An empirical study of algorithms for point-feature label placement. *ACM Transact. on Graphics*, 14(3):203–232, 1995.
5. T. Erlebach, K. Jansen, and E. Seidel. Polynomial-time approximation schemes for geometric graphs. In *Proc. SODA'01*, pages 671–679, Washington, 2001.
6. M. Formann and F. Wagner. A packing problem with applications to lettering of maps. In *Proc. 7th Annu. ACM Sympos. Comput. Geom.*, pages 281–288, 1991.
7. J. Y. Hsiao, C. Y. Tang, and R. S. Chang. An efficient algorithm for finding a maximum weight 2-independent set on interval graphs. *IPL*, 43(5):229–235, 1992.
8. C. Iturriaga. *Map Labeling Problems.* PhD thesis, University of Waterloo, 1999.
9. J. L. Morrison. Computer technology and cartographic change. In D. Taylor, ed., *The Computer in Contemporary Cartography.* J. Hopkins Univ. Press, NY, 1980.
10. S.-H. Poon, C.-S. Shin, T. Strijk, and A. Wolff. Labeling points with weights. TR 7/2001, Universität Greifswald, May 2001. Available at `http://www.math-inf.uni-greifswald.de/map-labeling/papers/pssw-lpw-01t.ps.gz`.
11. T. Strijk and M. van Kreveld. Practical extensions of point labeling in the slider model. In *Proc. 7th ACM Symp. on Advances in GIS*, pages 47–52, 1999.
12. M. van Kreveld, T. Strijk, and A. Wolff. Point labeling with sliding labels. *Computational Geometry: Theory and Applications*, 13:21–47, 1999.
13. A. Wolff and T. Strijk. The Map-Labeling Bibliography. `http://www.math-inf.uni-greifswald.de/map-labeling/bibliography/`, 1996.