Multi-Row Boundary-Labeling Algorithms for Panorama Images

Andreas Gemsa, Karlsruhe Institute of Technology
Jan-Henrik Haunert, University of Osnabrück
Martin Nöllenburg, Karlsruhe Institute of Technology

Boundary labeling deals with placing annotations for objects in an image on the boundary of that image. This problem occurs frequently in situations where placing labels directly in the image is impossible or produces too much visual clutter. Examples are annotating maps, photos, or technical/medical illustrations. Previous algorithmic results for boundary labeling consider a single layer of labels along some or all sides of a rectangular image. If, however, the number of labels is large or the labels are too long, multiple layers of labels are needed.

In this paper we study boundary labeling for panorama images, where n points in a rectangle R are to be annotated by disjoint unit-height rectangular labels placed above R in K different rows (or layers). Each point is connected to its label by a vertical leader that does not intersect any other label. We present polynomial time algorithms based on dynamic programming that either minimize the number of rows to place all n labels, or maximize the number (or total weight) of labels that can be placed in K rows for a given integer K. For weighted labels, the problem is shown to be (weakly) NP-hard, and we give a pseudo-polynomial algorithm to maximize the weight of the selected labels. We have implemented our algorithms; the experimental results show that solutions for realistically-sized instances are computed instantaneously. Further, we have also investigated two-sided panorama labeling, where the labels may be placed above or below the panorama image. In this model all of the aforementioned problems are NP-hard. For solving them we propose mixed integer linear program formulations.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—Geometrical problems and computations, Routing and layout

General Terms: Algorithms, Design, Experimentation, Theory

Additional Key Words and Phrases: image annotation, point labeling

ACM Reference Format:
Andreas Gemsa, Jan-Henrik Haunert, and Martin Nöllenburg, 2014. Multi-Row Boundary-Labeling Algorithms for Panorama Images *ACM* 0, 0, Article 0 (2014), 30 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Annotating features of interest in images by textual labels or icons is an essential aspect of information visualization. Depending on application and image content these labels are either placed directly next to the features within the image or, if partial

This work was started at Schloss Dagstuhl during Seminar 10461 “Schematization in Cartography, Visualization, and Computational Geometry” in November 2010. A preliminary version of this paper appeared in Gemsa et al. [2011]. Andreas Gemsa and Martin Nöllenburg received financial support by the *Concept for the Future* of KIT within the framework of the German Excellence Initiative.
Author’s addresses: A. Gemsa and M. Nöllenburg, Institute of Theoretical Informatics, Karlsruhe Institute of Technology (KIT), Germany; J.-H. Haunert, Institute for Geoinformatics and Remote Sensing, University of Osnabrück, Germany.

DOI: http://dx.doi.org/10.1145/0000000.0000000

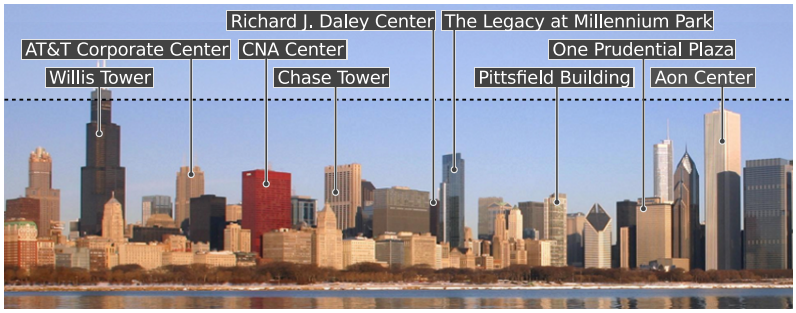


Fig. 1: Panorama labeling for the skyline of Chicago. Photography: ©J. Crocker.

occlusion of the image by labels is unacceptable or if feature density is too high, on the image boundary. In the first, so-called *internal labeling model*, which is common, for example, for placing object names in topographic maps, the association between feature and label should be clear from the spatial proximity between them. This is no longer the case in the latter *boundary labeling model* and hence features and associated labels are connected to each other using simple arcs. In this paper we consider a new and practically important boundary labeling variant, motivated by labeling features in panorama images of, for example, skylines or mountain ranges. Such labeled illustrations are frequently used for describing landmarks and buildings at lookout points or in tourist guide books. Moreover, very wide panorama photographs of streets of houses as used in popular commercial digital road maps are often annotated by information about local businesses or other points of interest. Another application in the area of augmented reality is the GeoScope [Brenner et al. 2006] system that augments camera images by textual information and thereby allows a user to explore a panorama, for example, the skyline of a city. A common aesthetic requirement in all these examples is that the labels are placed above a *horizon line*, for example, in the area taken up by the sky or simply above the actual image. In other cases, for example, when annotating events on a time line, a similar labeling scheme is used, where labels can be placed either above an upper horizon line or below a lower horizon line.

In this paper we present efficient algorithms for optimizing boundary labelings of such panorama images. Figure 1 shows a labeling in our model: we are given a set of n feature points (or *sites*) in a rectangle R and for each point a variable-width but unit-height open rectangular label (think of the bounding box of the object name written as a single line of text). In order to achieve at least a horizontal proximity between points and labels, every label must be placed vertically above its associated feature point. Each label and its associated site are connected by a vertical line segment, called *leader*. Our labeling model, in which the lower edge of a label can slide horizontally along the upper leader endpoint, can be seen as an extension of the so-called *1-slider* model [van Kreveld et al. 1999]. The algorithmic problem is to select a subset of the labels and for each selected label compute a label position (that is, a row index and a horizontal slider position) such that no two labels overlap and no leader intersects any label other than its own. We consider two basic optimization goals: (i) minimize the number of rows required to place *all* labels, and (ii) maximize the number of labels that can be placed in K rows.

We start with a review of related work on boundary labeling and point out our main contributions.

1.1. Related Work

Algorithmic label placement problems have been studied in computational geometry for more than 20 years now [Formann and Wagner 1991]; a vast body of literature is collected in the map-labeling bibliography [Wolff and Strijk 1996]. Most of the literature, however, is concerned with internal label placement as traditionally used in cartography. Boundary labeling as an alternative labeling model was first introduced as an algorithmic problem by Bekos et al. [2007] and has subsequently been studied in various flavors; see also the recent survey by Kaufmann [2009]. We note that in most boundary labeling problems all labels are placed in only a single row/column.

Different boundary labeling models can be classified by (a) the shape of the leaders, (b) the sides of R at which labels can be placed, and (c) further restrictions about the labels such as variable or uniform size, placement in multiple layers etc. Leaders are usually represented as polygonal lines; we argue that for readability, the leader shape should be as simple as possible, but also have a contrast high enough to be distinguishable from the background. Aiming for leader simplicity can be observed as a design goal in most manually created boundary labelings in practice and it also follows the unambiguity criterion for external labels proposed by Hartmann et al. [2005]. Leaders of arbitrary orientation without bends are called *straight* or *type-*s leaders. To reduce visual clutter axis-parallel leaders are often preferred over arbitrary type-s leaders. The shape of an axis-aligned polygonal leader starting from the feature point is described by a string over the alphabet $\{p, o\}$, where p and o denote, respectively, leader segments parallel and orthogonal to the side of R containing the label. If a segment is diagonal at a fixed angle (for example, $45°$), we use the letter d to refer to its orientation. The letters in the string correspond to the sequence of leader segments, starting from the feature point and ending at the label port.

Bekos et al. [2007] presented efficient labeling algorithms in the one-, two-, and four-sided model using type-s, type-po and type-opo leaders. Their main objective was to minimize the total leader length, but they also presented an algorithm for minimizing the number of bends in one-sided opo-labeling. Benkert et al. [2009] studied algorithms for one- and two-sided po- and do-labeling with arbitrary leader-dependent cost functions (including total leader length and number of bends); the algorithms were implemented and their performance was evaluated experimentally. Nöllenburg et al. [2010] presented a po-labeling algorithm for placing boundary labels that can slide along one side of the boundary so that the total leader length is minimized. Recently, Kindermann et al. [2013] presented an algorithm that finds a crossing-free po-labeling where the labels are placed on two adjacent sides of R. In the same paper the authors also extend the algorithm such that it can handle three and four-sided labelings. Bekos et al. [2010] presented algorithms for combinations of more general *octilinear* leaders of types do, od, and pd and labels on one, two, and four sides of R. For uniform labels the algorithms are polynomial, whereas the authors showed NP-hardness for a variant involving non-uniform labels. Recently, Huang et al. [2014] considered the problem of computing boundary labelings with flexible label positions. They gave several polynomial-time algorithms for computing one- and two-sided boundary labelings with minimum leader length or minimum total number of bends.

Extensions of the basic boundary labeling model include algorithms for labeling area features [Bekos et al. 2010], and a dynamic one-sided po-model, in which the user can zoom and pan the map view while the label size on the boundary remains fixed [Nöllenburg et al. 2010]. Relaxing the property that each label is connected to a unique site leads to the many-to-one model. In this model NP-hardness results, approximations and heuristics for crossing minimization with type-opo and -po leaders are known [Hao-Jen Kao 2007]; Lin [2010] presented an approach using duplicate

labels and opo-hyperleaders to avoid crossings. Bekos et al. [2014] extended the work in the many-to-one model and gave algorithms and complexity results for using po-hyperleaders, so-called *backbones*. Fink et al. [2012] consider the problem of labeling focus regions on a map by using either straight lines or Bézier curves as leaders.

The only previous work using multiple layers of labels on the boundary presented $O(n^4 \log H)$-time algorithms for label size maximization in a one-sided model with two or three "stacks" of labels on a vertical side of R and type-opo leaders [Bekos et al. 2006] (here H is the height of the rectangle R). In the algorithms all labels are assumed to be of uniform size and a maximum scaling factor is determined such that all labels can be placed in the available stacks. The authors further gave NP-hardness results for some two-stack variants of non-uniform labels and opo- or po-leaders.

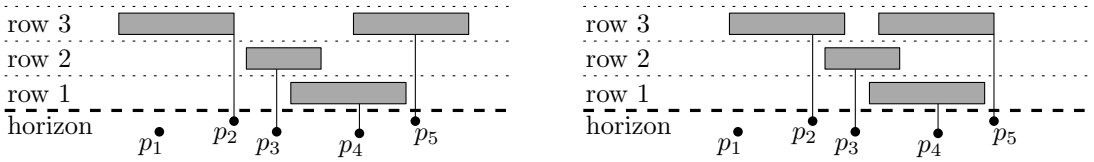
1.2. Contribution

In our paper we study a one-sided multi-row labeling problem with type-o leaders and variable-width labels. Note that for comparison with the results of Bekos et al. [2006], the same model can be transformed into an equivalent multi-stack labeling problem with variable-height labels; since for textual annotation variable-width labels are more relevant, we describe the multi-row model. However, we argue that using a multi-stack (or multi-row) model with opo-leaders exhibits the risk of cluttered drawings. For example, multiple leaders may squeeze through a narrow gap between two labels and thus correspondences between points and labels may become unclear. In contrast, by restricting the leaders to vertical straight-line segments we try to compensate for the relatively high visual complexity of a multi-row model and to maintain the visual association.

In Section 2 we introduce our labeling model, in which we place all labels above the horizon, in more detail and define three optimization problems: MINROW, which aims to find a labeling with all n labels in the minimum feasible number $K^\star$ of rows, MAXLABELS, which maximizes the number of labels placed in K given rows, and MAXWEIGHT, which considers labels with integer weights, weighted by importance (with total weight w_{total}) and computes a maximum-weight subset of labels for K rows. Section 3 describes an $O(K^\star n^3)$-time algorithm for MINROW, an $O(Kn^3 w_{\text{total}}^2)$-time algorithm for MAXWEIGHT, and an $O(Kn^3)$-time algorithm for MAXLABELS. Additionally, we discuss the problem of generating two-sided panorama labelings, where labels may be placed above and below the respective horizons. In this context, all of the above mentioned problems are NP-hard and we cannot give efficient algorithms. Instead, we present mixed-integer linear programming formulations to solve each problem. In Section 4 we present extensions of the algorithms for practically interesting variations of the basic problems. We have implemented our algorithms and report results of an experimental evaluation concerning the performance of the algorithms with respect to running time and the number rows used or number of placed labels in Section 5.

2. PANORAMA LABELING MODEL

We aim to label a set $P = \{p_1, \ldots, p_n\}$ of points in the plane, where $p_i = (x_i, y_i) \in \mathbb{R}^2$ with corresponding *labels* $L = \{l_1, \ldots, l_n\}$. Each label l_i is an open, axis-parallel rectangle of width $W_i \in \mathbb{R}_0^+$ and height 1. We assume that the points in P have distinct x-coordinates and the points are ordered from left to right, that is, $x_i < x_{i+1}$ for $i = 1, \ldots, n-1$. Moreover, we assume $y_i < 0$ for $i = 1, \ldots, n$ meaning that p_i lies below the horizontal line $y = 0$, which we call the *horizon*. For our problems, the y-coordinates of the points in P are irrelevant and we can assume that all points lie on the horizontal line $y = -1$. By $P_{i,j} = \{p_i, \ldots, p_j\} \subseteq P$ we denote the set of all input points between p_i and p_j.



(a) A non-compact $(2, 5, 3)$-labeling. (b) A compact $(2, 5, 3)$-labeling.

Fig. 2: Examples for a compact and a non-compact (i, j, k)-labeling.

The task of labeling P comprises two subproblems: selecting the labels that are to be displayed and placing them above the horizon. More formally, we define a *(panorama) labeling* $\mathcal{L}$ to be the tuple (S, π) where $S \subseteq L$ and $\pi\colon S \to \mathbb{R} \times \mathbb{N}$ is a mapping that assigns each label l_i in S a coordinate (X_i, Y_i) where X_i is the real-valued x-coordinate of the label's right border and Y_i is an integer which allows us to say that the label is placed in *row* Y_i above the horizon. We also call X_i the *x-position* of label l_i.

In order to connect each label $l_i \in S$ with its corresponding point $p_i \in P$, we draw a vertical line segment from (x_i, Y_i) to (x_i, y_i); we call this line segment the *leader* of label l_i and point p_i.

We say that a labeling $\mathcal{L} = (S, \pi)$ is *feasible* if it satisfies requirements (F1)–(F3):

(F1) For every label $l_i \in S$ the leader of l_i actually connects l_i with p_i, that is, $X_i - W_i \leq x_i \leq X_i$.

(F2) For every label $l \in S$ the leader of l does not intersect any label in S other than l.

(F3) The labels in S do not overlap.

Note that since we consider labels to be *open* rectangles, strictly speaking, a label is not connected with its leader if, for example, $X_i = x_i$. However, for the sake of simplicity we assume that a leader is connected with its label if the requirement stated in (F1) holds.

For a pair (i, j) of indices $1 \leq i \leq j \leq n$ and a row index k, we define an (i, j, k)-*labeling* as a feasible labeling of $P_{i,j}$ with $S = \{l_i, \ldots, l_j\}$ satisfying

(R1) $Y_i = Y_j = k$, that is, both l_i and l_j are in row k, and

(R2) $Y_\ell \leq k$ for $\ell = i + 1, \ldots, j - 1$, that is, the labels for all points between p_i and p_j are in row k or below.

We say that an (i, j, k)-labeling $\mathcal{L}$ is *compact* if there is no (i, j, k)-labeling where label l_j has a smaller x-coordinate than in $\mathcal{L}$; we denote the x-coordinate of l_j in a compact (i, j, k)-labeling $\mathcal{L}$ as the *x-position* $\mathcal{X}_{i,j,k}$ of $\mathcal{L}$. If no (i, j, k)-labeling exists we set $\mathcal{X}_{i,j,k} = \infty$. For an example of (non-)compact (i, j, k)-labelings see Figure 2.

For multi-row boundary labeling there are several possible optimization criteria. Generally, we want to display as many labels as possible, decrease unused space but also ensure good readability. Note that these criteria might be conflicting, for example, putting many labels as close together as possible decreases unused space, but may also be detrimental for the readability (or more specifically, it may be harder to find the label corresponding to a specific feature point). Hence, we might want to limit the number of rows used for displaying labels. However, this may come at the price of fewer displayed labels than possible. In the following we introduce three optimization problems that we investigate in this paper.

As a first, and probably most basic, problem we want to find a solution that selects *all* labels and uses as few rows as possible.

Problem 2.1 (MINROW). Given a set P of n points below the horizon and a set L of corresponding labels, find a feasible labeling with *all* labels that requires the minimum number of rows.

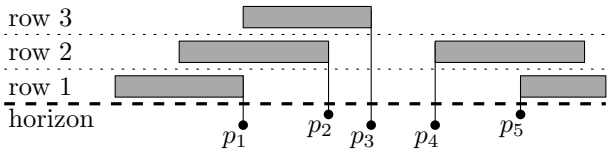


Fig. 3: A feasible labeling for $n = 5$ labels using $\lceil n/2 \rceil = 3$ rows.

A solution to this problem contains all provided information, but might be hard to read if too many features with corresponding labels are supplied. Nevertheless, there exists for every instance such a labeling.

LEMMA 2.2. *For each* MINROW *instance with n points there exists a feasible labeling with $\lceil n/2 \rceil$ rows and for each $n \in \mathbb{N}$ there exists a* MINROW *instance with n points that requires $\lceil n/2 \rceil$ rows.*

PROOF. We begin by showing the first part of Lemma 2.2, that is, we show that for each MINROW instance with n points, there exists a feasible labeling with $\lceil n/2 \rceil$ rows. We label all points in P as in Figure 3, that is, for $i = 1, \ldots, \lceil n/2 \rceil$ we set $Y_i = i$ and $X_i = x_i$; for $i = \lceil n/2 \rceil + 1, \ldots, n$ we set $Y_i = n - i + 1$ and $X_i = x_i + W_i$. Clearly, requirement (F1) holds. Requirement (F2) holds since for $i = 1, \ldots, n-1$ and $j = i + 1, \ldots, n$ label l_j lies above Y_i if $j \leq \lceil n/2 \rceil$ and to the right of x_i if $j > \lceil n/2 \rceil$. Requirement (F3) holds since each row either contains a single label or two labels l_i, l_j with $i \leq \lceil n/2 \rceil < j$. In the latter case l_i lies to the left of l_j since the right boundary of l_i is $X_i = x_i$, the left boundary of l_j is $X_j - W_j = x_j$, and $i < j$ implies $x_i < x_j$. To show that $\lceil n/2 \rceil$ rows may be required let $p_i = (i, -1)$ and $W_i = n$ for $i = 1, \ldots, n$. Since the distance between p_1 and p_n is $n - 1$ at most two labels fit in one row. □

Obviously, solutions as the one in Figure 3 are rather useless in practice—when placing no more than two labels in a single row the required space to display all labels becomes huge. If the horizontal distances between the points are small and the labels are wide, however, we will fail to find a better solution. Therefore, a natural alternative to MINROW is to discard some labels if the available space is limited. If we are restricted to a certain number of rows, then a sensible optimization goal is to maximize the number of displayed labels.

Problem 2.3 (MAXLABELS). Given a set P of n points below the horizon, a corresponding set L of labels, and a positive integer K, determine a feasible labeling that displays the maximum number of labels in at most K rows.

However, by simply maximizing the number of labels we might fail to measure the quality of a labeling appropriately. We should account for the fact that often some objects are more important than others and thus should have a higher priority to become labeled. This can be expressed with *weighted* points and yields our third optimization problem.

Problem 2.4 (MAXWEIGHT). Given a set P of n points below the horizon, a corresponding set L of labels, a positive integer K, and a positive integer weight w_i for each point $p_i \in P$, find a feasible labeling $\mathcal{L} = (S, \pi)$ that maximizes the total weight $\sum_{l_i \in S} w_i$ among all feasible labelings that use at most K rows.

3. ALGORITHMS

In this section we describe our algorithms to solve the problems MINROW, MAXWEIGHT, and MAXLABELS. For MINROW and MAXLABELS we give polynomial-time algorithms,

while we show that MAXWEIGHT is weakly NP-hard by a reduction from the PARTITION problem; we give a pseudo-polynomial time algorithm in this case.

We begin this section with a simple algorithm that decides for a given panorama labeling instance whether there exists a valid labeling in a single row, that is, whether MAXLABELS has a solution with n labels for $K = 1$.

3.1. Single-Row

Before we describe our algorithm for MINROW in Section 3.2, we briefly discuss the following single-row label placement problem, which serves as a base case for MINROW.

Problem 3.1 (SINGLEROW). Given a set P of n points below the horizon and a label for each point, decide whether there is a feasible labeling for P with *all* labels in a single row above the horizon.

Bekos et al. [2008] considered a similar problem, more specifically, they considered the problem of computing a boundary labeling of points on a horizontal line. However, they allow that the leaders have bends and optimize the total leader length or the number of total bends, where we require that the leaders are connected with the labels with a single vertical line.

The SINGLEROW problem is closely related to a single-machine job scheduling problem, where n ordered jobs $J_1 < \cdots < J_n$ with processing times z_i and release and due times r_i and d_i are to be non-preemptively scheduled in the given order such that all jobs finish before their due times. The weighted version of this problem is known as single-machine throughput maximization [Arkin and Silverberg 1987] and has been related to one-dimensional weighted point labeling problems before [Poon et al. 2003; Bekos et al. 2008]. Note that for our problem the property that the scheduling is non-preemptive (that is, no job is interrupted) is crucial since each job corresponds to a label, and each label is displayed either completely or not at all.

SINGLEROW can be solved with a simple greedy algorithm, which we denote as SingleRowAlg. The algorithm processes the points in increasing x-order and places the next label l_i in the leftmost possible position such that it does not intersect the previous label, that is, $X_i = \max\{X_{i-1} + W_i, x_i\}$. If for any i we have $X_i > x_i + W_i$, then obviously no feasible single-row labeling exists (requirement (F1) violated), the algorithm reports failure and returns ∞. Otherwise it reports success and returns the position X_n of the last label. The correctness of SingleRowAlg is immediate and for sorted points it takes linear time.

COROLLARY 3.2. *If the input points are sorted by their x-coordinates,* SINGLEROW *can be solved in $O(n)$ time.*

Using SingleRowAlg we can compute the x-positions $\mathcal{X}_{i,j,1}$ of all compact $(i, j, 1)$-labelings in $O(n^2)$ time by running it n times, once for every set $P_{i,n}$, where $i = 1, \ldots, n$. If there is a feasible position for placing label l_j ($j \geq i$) in the instance $P_{i,n}$, we store its x-position X_j as the x-position $\mathcal{X}_{i,j,1}$. If there is no feasible position we set $\mathcal{X}_{i,j,1} = \infty$.

3.2. Row Number Minimization

We now show how to solve MINROW efficiently using dynamic programming. Our idea is to construct compact (i, j, k)-labelings for all $1 \leq i \leq j \leq n$ and successively increasing values of k until a feasible labeling for all points is found. Recall that by Lemma 2.2 the value of k is upper bounded by $\lceil n/2 \rceil$.

To ease notation we introduce two dummy points p_0 and p_{n+1} to the left of p_1 and to the right of p_n such that any labeling positions of l_0 and l_{n+1} do not influence the feasibility

of labeling P. We set $W_0 = W_{n+1} = 0$, $x_0 = x_1 - 2W_{\max}$, and $x_{n+1} = x_n + 2W_{\max}$, where $W_{\max} = \max_{1 \leq i \leq n} W_i$ is the maximum label width.

Our algorithm MinRowAlg computes a three-dimensional table $\mathcal{T}$, where each entry $\mathcal{T}[i, j, k]$ for $0 \leq i \leq j \leq n + 1$ and $1 \leq k \leq \lceil n/2 \rceil$ stores the x-position $\mathcal{X}_{i,j,k}$ of a compact (i, j, k)-labeling as well as some backtracking information in order to reconstruct the compact (i, j, k)-labeling. With these semantics it is clear that there is a solution to MINROW with k rows if and only if there is a feasible $(0, n + 1, k)$-labeling, that is, $\mathcal{T}[0, n + 1, k] < \infty$.

We compute $\mathcal{T}$ in a bottom-up fashion with respect to the row index k. First, we compute $\mathcal{T}[\cdot, \cdot, k]$ for $k = 1$ and if $\mathcal{T}[0, n + 1, 1] = \infty$ proceed to computing $\mathcal{T}[\cdot, \cdot, k]$ for $k = 2$ and so on until eventually $\mathcal{T}[0, n + 1, k] < \infty$. The entries of $\mathcal{T}[\cdot, \cdot, k]$ for $k = 1$ are computed by the algorithm SingleRowAlg described in Section 3.1. For $k > 1$ the entries $\mathcal{T}[i, i, k]$ for $i = 0, \ldots, n+1$ are set to $\mathcal{T}[i, i, k] = \mathcal{X}_{i,i,k} = x_i$. We use the recurrence $\mathcal{T}[i, j, k] = \min \Theta^k_{i,j}$ for all $i < j$ and $k > 1$, where $\Theta^k_{i,j}$ for $i < j$ and $k > 1$ is defined as the set

$$\Theta^k_{i,j} = \{\max\{x_j, \mathcal{T}[i, \ell, k] + W_j\} \mid i \leq \ell < j,\ \mathcal{T}[i, \ell, k] \leq x_j,\ \mathcal{T}[\ell, j, k - 1] < \infty\}\,. \quad (1)$$

Note that $\Theta^k_{i,j}$ can be empty; in that case we define $\min \emptyset := \infty$ and obtain $\mathcal{T}[i, j, k] = \infty$. For the pseudo code of MinRowAlg see Algorithm 1.

ALGORITHM 1: MinRowAlg

```
initialize T, k
compute T[·,·,1] using SingleRowAlg
for k = 2 to ⌈n/2⌉ do
    for j = 1 to n + 1 do
        for i = 0 to j − 1 do
            T[i, j, k] = min Θ^k_{i,j}
        T[j, j, k] = x_j
    if T[0, n + 1, k] < ∞ then break;
reconstruct solution from T[0, n + 1, k]
```

THEOREM 3.3. *MinRowAlg solves* MINROW *in* $O(K^\star \cdot n^3)$ *time, where* $K^\star$ *is the number of rows in the optimal* MINROW *solution.*

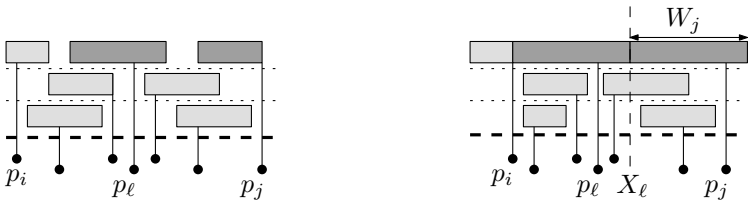
PROOF. The running time of MinRowAlg follows immediately. The outer loop is iterated $K^\star$ times, and in each iteration of the outer loop we perform $O(n^2)$ iterations of the nested inner loops. In each iteration it takes linear time to find the minimum of the set $\Theta^k_{i,j}$, which contains $O(n)$ elements.

It remains to prove the correctness of MinRowAlg by showing $\mathcal{T}[i, j, k] = \mathcal{X}_{i,j,k}$ for all $0 \leq i \leq j \leq n + 1$ and $1 \leq k \leq K^\star$. The proof is by induction over k and for each k by induction over $j - i$.

For $k = 1$ we use SingleRowAlg described in Section 3.1 and it follows immediately that the entries $\mathcal{T}[\cdot, \cdot, 1]$ are correct.

Next we prove correctness for $k > 1$. In the base case $i = j$ the algorithm sets $\mathcal{T}[i, i, k] = x_i$, which is the x-position $\mathcal{X}_{i,i,k}$ of a compact (i, i, k)-labeling that simply consists of the single label l_i placed in its leftmost possible position $X_i = x_i$ in row $Y_i = k$.

Now let's assume $i < j$ and let $\mathcal{L}$ be an (i, j, k)-labeling. By definition l_i and l_j are in row k. This implies that there is a well-defined *predecessor* l_ℓ of l_j in row k for $i \leq \ell < j$;



(a) $X_j = x_j$ (b) $X_j = X_\ell + W_j$

Fig. 4: Two ℓ-compact $(i, j, 3)$-labelings.

we say that l_ℓ *precedes* l_j in row k. We call a label l_ℓ a *feasible* predecessor of l_j if there exists an (i, j, k)-labeling, where l_ℓ precedes l_j. Let $F(i, j, k)$ be the set of all feasible predecessors of l_j in an (i, j, k)-labeling. Now we define an (i, j, k)-labeling $\mathcal{L}$ to be *ℓ-compact* if l_ℓ precedes l_j and there is no other (i, j, k)-labeling with predecessor l_ℓ where the position of l_j is further to the left. Figure 4 shows two ℓ-compact $(i, j, 3)$-labelings.

Every compact (i, j, k)-labeling $\mathcal{L}$ is also ℓ-compact for the predecessor l_ℓ of l_j in $\mathcal{L}$ since by definition of a compact (i, j, k)-labeling, there is no other (i, j, k)-labeling where the position of l_j is further to the left. On the other hand the ℓ-compact (i, j, k)-labeling with the leftmost position of l_j over all feasible predecessors $l_\ell \in F(i, j, k)$ is a compact (i, j, k)-labeling, since there is no other (i, j, k)-labeling where the position of l_j is further to the left. For every ℓ-compact (i, j, k)-labeling the leftmost x-position of l_j is $X_j^\ell = \max\{x_j, X_\ell + W_j\}$, since l_j must be W_j to the right of its predecessor but cannot be left of x_j, see Figure 4. Hence the x-position of a compact (i, j, k)-labeling $\mathcal{X}_{i,j,k} = \min\{X_j^\ell \mid l_\ell \in F(i, j, k)\}$. Note that if $F(i, j, k) = \emptyset$ we obtain $\mathcal{X}_{i,j,k} = \infty$. We claim that $\Theta_{i,j}^k = \{X_j^\ell \mid l_\ell \in F(i, j, k)\}$ and thus MinRowAlg correctly computes $\mathcal{T}[i, j, k] = \min \Theta_{i,j}^k = \mathcal{X}_{i,j,k}$.

Let $\mathcal{L}$ be an ℓ-compact (i, j, k)-labeling and let $\mathcal{L}_{\text{left}}$ be the (i, ℓ, k)-labeling formed by labels $l_i, \ldots, l_\ell$ of $\mathcal{L}$. We can assume that $\mathcal{L}_{\text{left}}$ is compact since otherwise we can replace $\mathcal{L}_{\text{left}}$ in $\mathcal{L}$ by a compact (i, ℓ, k)-labeling that actually constrains the position of l_j *less* than $\mathcal{L}_{\text{left}}$. Hence $X_j^\ell = \max\{x_j, \mathcal{X}_{i,\ell,k} + W_j\}$. Since $\ell < j$ we obtain from the induction hypothesis that $\mathcal{X}_{i,\ell,k} = \mathcal{T}[i, \ell, k]$, that is, the values in $\Theta_{i,j}^k$ are actually $X_j^\ell = \max\{x_j, \mathcal{T}[i, \ell, k] + W_j\}$.

It remains to show that a value X_j^ℓ is in $\Theta_{i,j}^k$ if and only if l_ℓ is a feasible predecessors in $F(i, j, k)$. For the only-if-part, let l_ℓ be a feasible predecessor in $F(i, j, k)$. Then obviously $i \leq \ell < j$. Moreover, $\mathcal{X}_{i,\ell,k} = \mathcal{T}[i, \ell, k] \leq x_j$ since otherwise l_j would be pushed too far to the right for being part of an (i, j, k)-labeling. Finally, we observe that any ℓ-compact (i, j, k)-labeling induces a labeling $\mathcal{L}_{\text{right}}$ of the labels $l_{\ell+1}, \ldots, l_{j-1}$, in which they are restricted to lie to the right of x_ℓ, to the left of x_j and below row k. We can extend $\mathcal{L}_{\text{right}}$ to an $(\ell, j, k-1)$-labeling by placing l_ℓ at $X_\ell = x_\ell$, $Y_\ell = k - 1$ and l_j at $X_j = x_j + W_j$, $Y_j = k - 1$; see Figure 5. This implies $\mathcal{T}[\ell, j, k-1] < \infty$.

For the if-part, let $i \leq \ell < j$ such that $\mathcal{T}[i, \ell, k] \leq x_j$ and $\mathcal{T}[\ell, j, k-1] < \infty$. We combine a compact (i, ℓ, k)-labeling $\mathcal{L}_{\text{left}}$ (which exists because $\mathcal{T}[i, \ell, k] \leq x_j < \infty$) and labels $l_{\ell+1}, \ldots, l_{j-1}$ from a compact $(\ell, j, k-1)$-labeling with the label l_j at position $X_j = x_j + W_j$ in row k as illustrated in Figure 5. This yields a feasible (i, j, k)-labeling $\mathcal{L}$ since $\mathcal{L}_{\text{left}}$ is feasible and labels $l_{\ell+1}, \ldots, l_{j-1}$ lie below row k, to the right of x_ℓ, and to the left of x_j. We know that $\mathcal{T}[i, \ell, k] < x_j$ and hence l_j can be placed at $x_j + W_j$ without overlapping l_ℓ. Furthermore, both l_i and l_j are in row k and l_ℓ precedes l_j, that is, l_ℓ is a feasible predecessor in $F(i, j, k)$. This concludes the proof of correctness. □

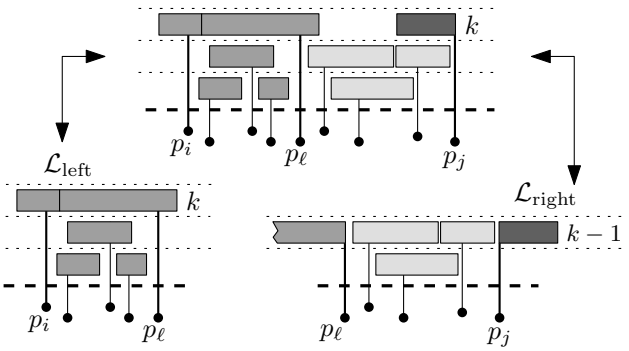


Fig. 5: (De-)composition of an ℓ-compact (i, j, k)-labeling into an (i, ℓ, k)-labeling and an $(\ell, j, k-1)$-labeling.

Note that $K^\star$ is in $O(n)$, hence the worst-case time complexity of MinRowAlg is $O(n^4)$.

3.3. Weight Maximization

In this section we first show NP-hardness of MAXWEIGHT. Then, we present a pseudo-polynomial time algorithm showing that MAXWEIGHT is actually only weakly NP-hard. Recall that for a MAXWEIGHT instance we are given, in addition to the set P of feature points and the set L of corresponding labels, for each point p_i in P a positive integer weight w_i, and we are also given a positive integer K. The problem then asks for a feasible labeling $\mathcal{L} = (S, \pi)$ that maximizes the total weight $\sum_{l_i \in S} w_i$ among all feasible labelings that use at most K rows.

THEOREM 3.4. MAXWEIGHT *is* NP*-hard, even for* $K = 1$.

PROOF. Our proof is inspired by an NP-hardness proof for an internal 4-slider point labeling problem by Garrido et al. [2001, Theorem 3]. It is by reduction from the following NP-hard variant of PARTITION [Garey and Johnson 1990]: Given a set $A = \{a_1, a_2, \ldots, a_{2m}\}$ and a size $s(a) \in \mathbb{Z}^+$ for each $a \in A$, is there a subset $A' \subseteq A$ such that $\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A'} s(a)$ and A' contains exactly one of a_{2i-1}, a_{2i} for every $1 \leq i \leq m$? For each PARTITION-instance I we construct a MAXWEIGHT-instance J such that the weight of an optimal solution to J allows us to decide whether or not I is a yes-instance of PARTITION. We specify the set P of points, their weights, and the widths of their labels in J as illustrated in Figure 6. More precisely, we define a large constant $C = 1000 \sum_{a \in A} s(a)$. Next we define (from left to right) points $p_{\text{L}}, p_1, \ldots, p_{2m}, p_{\text{R}}$ on a horizontal line and their distances as $|\overline{p_{\text{L}} p_1}| = |\overline{p_{2m} p_{\text{R}}}| = C/2$, $|\overline{p_{2i-1} p_{2i}}| = (s(a_{2i-1}) + s(a_{2i}))/2$, and $|\overline{p_{2i} p_{2i+1}}| = C$. The corresponding labels have widths $W_{\text{L}} = W_{\text{R}} = |\overline{p_{\text{L}} p_{\text{R}}}|$ and $W_i = C + s(a_i)$. Furthermore, we define $w_{\text{L}} = w_{\text{R}} = W_{\text{L}}$ and $w_i = W_i$, that is, the weight of a point is equal to the width of its label.

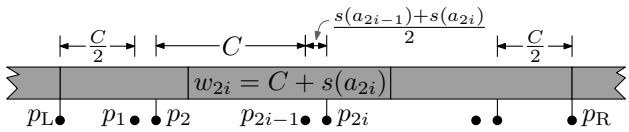


Fig. 6: Reducing Partition to MAXWEIGHT.

A solution to the PARTITION instance I exists if and only if a feasible labeling in a single row with total weight $3\,|\overline{p_{\mathrm{L}}p_{\mathrm{R}}}|$ exists. To see why, we first assume that we are given such a labeling. Clearly, this labeling contains the label for p_{L} and the label for p_{R}. Moreover, the space between p_{L} and p_{R} is completely filled with labels for other points. Since the labels are extremely wide (compared to the distances $|\overline{p_{2i-1}p_{2i}}|$) exactly one of the labels l_1, l_2 has to be selected and its leader has to lie roughly in the center of the label. This again implies that exactly one of the labels l_3, l_4 has to be selected, and so on. By induction, it follows that for $i = 1, \ldots, m$ exactly one of the labels l_{2i-1}, l_{2i} has to be selected. Since $|\overline{p_{\mathrm{L}}p_{\mathrm{R}}}|$ is exactly half the total width of labels $l_1, \ldots, l_{2m}$ the labeled points correspond to a feasible partition. If, however, we are given a solution $A' \subseteq A$ to the PARTITION instance, we can construct the labeling by selecting labels p_{L}, p_{R} plus the labels for the points corresponding to the elements in A'. It is easy to see that by applying algorithm SingleRowAlg from Section 3.1 to the selected labels, we obtain a feasible labeling. □

Since MAXWEIGHT is NP-hard, we cannot hope for a polynomial-time algorithm that solves it, unless P = NP. But our reduction uses labels with extremely large weights. Hence, we propose a pseudo-polynomial time algorithm, that is, an algorithm whose running time is polynomial in n, k, and the numeric value of $\sum_{i=1}^{n} w_i$, but exponential in the length of the encoding of $\sum_{i=1}^{n} w_i$. If the label weights are small integer numbers such an algorithm can still be fast in practice.

Pseudo-Polynomial Time Algorithm for MAXWEIGHT.. We first extend our notation from Section 3.2. Recall that for a labeling $\mathcal{L}$ the set S contains all labels selected for display. For a pair of indices $0 \le i \le j \le n+1$, a row index k, and a weight c we define an (i, j, k, c)-labeling as a feasible labeling of $P_{i,j}$ such that $l_i, l_j \in S$ and $Y_i = Y_j = k$, all other labels $l_\ell \in S$ are placed in row k or below (i.e., $Y_\ell \le k$), and the total weight of the labels in S is c. Analogously, we say an (i, j, k, c)-labeling is *compact* if there is no other (i, j, k, c)-labeling where l_j has a smaller x-position. Again, we call the x-position of l_j in a compact (i, j, k, c)-labeling $\mathcal{L}$ the x-position of $\mathcal{L}$. Note that this definition generalizes our definition of a compact (i, j, k)-labeling, since an (i, j, k, c)-labeling is an (i, j, k)-labeling if $c = \sum_{\ell=i}^{j} w_\ell$.

In our algorithm MaxWeightAlg for MAXWEIGHT we add a fourth dimension to the table $\mathcal{T}$ that allows us to distinguish labelings of different weights. Let $w_{\text{total}} = \sum_{i=1}^{n} w_i$ be the total weight of all points in P. Then, each entry $\mathcal{T}[i, j, k, c]$ stores the x-position of a compact (i, j, k, c)-labeling, where $0 \le i \le j \le n+1$, $1 \le k \le \lceil n/2 \rceil$, and $0 \le c \le w_{\text{total}}$. The maximum-weight labeling using K rows can be constructed by backtracking from $\mathcal{T}[0, n+1, K, c_{\max}]$, where $c_{\max} = \max\{c \mid \mathcal{T}[0, n+1, K, c] < \infty\}$.

As before, we compute $\mathcal{T}$ in a bottom-up fashion with respect to the topmost row k, the weight c, and the distance $j - i$. We set $\mathcal{T}[i, i, k, w_i] = x_i$ for all k and $\mathcal{T}[i, i, k, c] = \infty$ for all k and $c \ne w_i$. In all other cases we use the recurrence $\mathcal{T}[i, j, k, c] = \min \Theta_{i,j}^{k,c}$, where $\Theta_{i,j}^{k,c}$ for $k = 1$ is defined as the set

$$\Theta_{i,j}^{1,c} = \{\max\{x_j, \mathcal{T}[i, \ell, 1, a] + W_j\} \mid i \le \ell < j,\ \mathcal{T}[i, \ell, 1, a] \le x_j,\ a = c - w_j\} \tag{2}$$

and for $k > 1$ as the set

$$\Theta_{i,j}^{k,c} = \left\{ \max\{x_j, \mathcal{T}[i, \ell, k, a] + W_j\} \mid \begin{array}{c} i \le \ell < j,\ \mathcal{T}[i, \ell, k, a] \le x_j, \\ \mathcal{T}[\ell, j, k-1, b] < \infty,\ a = c - b + w_\ell \end{array} \right\}. \tag{3}$$

We give the pseudo code for MaxWeightAlg in Algorithm 2.

THEOREM 3.5. *MaxWeightAlg solves* MAXWEIGHT *in* $O(Kn^3 w_{\text{total}}^2)$ *time.*

ALGORITHM 2: MaxWeightAlg

```
initialize 𝒯
for i = 0 to n + 1 do
    for k = 1 to K do
        𝒯[i, i, k, w_i] = x_i //initialze values for 𝒯
for j = 1 to n + 1 do
    for i = 0 to j − 1 do
        for c = 0 to w_total do
            𝒯[i, j, 1, c] = min Θ^{1,c}_{i,j} //compute optimal solutions for the first row
for k = 2 to K do
    for j = 1 to n + 1 do
        for i = 0 to j − 1 do
            for c = 0 to w_total do
                𝒯[i, j, k, c] = min Θ^{k,c}_{i,j} //compute optimal solution for all remaining rows
reconstruct solution from 𝒯[0, n + 1, K, c_max]
```

PROOF. The algorithm MaxWeightAlg is similar to MinRowAlg, but uses four instead of three nested loops to compute the $O(Kn^2 w_{\text{total}})$ entries of $\mathcal{T}$. Each entry is computed as the minimum of a set $\Theta_{i,j}^{k,c}$ containing $O(nw_{\text{total}})$ elements. This yields an overall running time of $O(Kn^3 w_{\text{total}}^2)$.

We now show the correctness of the algorithm analogously to the proof of Theorem 3.3 but taking the weight constraints into account. For the case $i = j$ and arbitrary k it is easy to see that the x-position of a compact (i, i, k, c)-labeling is the leftmost possible position x_i of l_i if $c = w_i$ and ∞ otherwise.

Before we consider the general case, we extend the notion of ℓ-compact as introduced in Section 3.2. For given weights $a, c \in \mathbb{N}$ with $a \geq c$, we call a pair (l_ℓ, a) a feasible predecessor pair of (l_j, c) if there exists an (i, j, k, c)-labeling ($i < j$), where l_ℓ precedes l_j and the total weight of the selected labels $S \cap \{l_i, \ldots, l_\ell\}$ is a. We define $F(i, j, k, c)$ as the set of all feasible predecessors pairs of (l_j, c) in an (i, j, k, c)-labeling. We then define an (i, j, k, c)-labeling $\mathcal{L}$ to be (ℓ, a)-compact if (l_ℓ, a) precedes (l_j, c) and there is no other (i, j, k, c)-labeling $\mathcal{L}'$ where (l_ℓ, a) precedes (l_j, c) and which has smaller x-position than $\mathcal{L}$. As before it is clear that every compact (i, j, k, c)-labeling $\mathcal{L}$ is also (ℓ, a)-compact for the predecessor pair (l_ℓ, a) of (l_j, c). Conversely, the (ℓ, a)-compact (i, j, k, c)-labeling with smallest x-position over all feasible predecessor pairs $(l_\ell, a) \in F(i, j, k, c)$ is compact. For every (ℓ, a)-compact (i, j, k, c)-labeling $\mathcal{L}$ the x-position of $\mathcal{L}$ is $X_j^{\ell,a} = \max\{x_j, X_\ell + W_j\}$. Note that the value X_ℓ depends implicitly on a since the value of X_ℓ depends on $\mathcal{L}$ which directly depends on a. To ease notation we omit this parameter in the description of X_ℓ. The x-position of a compact (i, j, k, c)-labeling is $\min\{X_j^{\ell,a} \mid (l_\ell, a) \in F(i, j, k, c)\}$. Our claim is that $\Theta_{i,j}^{k,c} = \{X_j^{\ell,a} \mid (l_\ell, a) \in F(i, j, k, c)\}$ and thus the algorithm is correct.

Let $\mathcal{L}$ be an (ℓ, a)-compact (i, j, k, c)-labeling and let $\mathcal{L}_{\text{left}}$ be the induced (i, ℓ, k, a)-labeling. As in the proof of Theorem 3.3 we can assume that $\mathcal{L}_{\text{left}}$ is compact and hence by the induction hypothesis $X_j^{\ell,a} = \max\{x_j, \mathcal{T}[i, \ell, k, a] + W_j\}$, that is, the values in $\Theta_{i,j}^{k,c}$ are actually $X_j^{\ell,a}$ for some pairs (ℓ, a).

It remains to show that $X_j^{\ell,a} \in \Theta_{i,j}^{k,c}$ if and only if the pair $(l_\ell, a) \in F(i, j, k, c)$. We start with the case $k = 1$. If $(l_\ell, a) \in F(i, j, 1, c)$, then obviously $i \leq \ell < j$ and also $\mathcal{T}[i, \ell, 1, a] \leq x_j$ since otherwise l_j cannot be in a feasible position. Moreover, for the total

weight of the labeling to be c, the weight of the labels in $S \cap \{l_i, \ldots, l_\ell\}$ must be $c - w_j$, since in a single row l_j is the only label to the right of l_ℓ. If, on the other hand, the three constraints for the set $\Theta_{i,j}^{1,c}$ hold, we can combine a compact $(i, \ell, 1, a)$-labeling (which exists because $\mathcal{T}[i, \ell, 1, a] < \infty$) with weight a and the label l_j with weight w_j placed in row 1 at position $X_j = x_j + W_j$ into an $(i, j, 1, c)$-labeling for $c = a + w_j$. Therefore, (l_ℓ, a) is indeed a feasible predecessor pair in $F(i, j, 1, c)$.

In the general case for $k > 1$ the argument is similar to $k = 1$. Let first $(l_\ell, a) \in F(i, j, k, c)$ be a feasible predecessor pair. Then $i \leq \ell < j$ and $\mathcal{T}[i, \ell, k, a] \leq x_j$ as before, but additionally any (ℓ, a)-compact labeling induces a labeling $\mathcal{L}_{\text{right}}$ of labels $S \cap \{l_{\ell+1}, \ldots, l_{j-1}\}$ that is strictly below row k, to the right of x_ℓ and to the left of x_j. Furthermore, $\mathcal{L}_{\text{right}}$ has weight $c - a - w_j$. Again, we extend $\mathcal{L}_{\text{right}}$ to an $(\ell, j, k-1, b)$-labeling by placing l_ℓ and l_j in row $k-1$ with x-positions $X_\ell = x_\ell$ and $X_j = x_j + W_j$, similar to the situation depicted in Figure 5. Note that the weight b of this labeling is $b = c - a - w_j + w_\ell + w_j$ so that all constraints put on set $\Theta_{i,j}^{k,c}$ are satisfied.

Conversely, if all constraints for $\Theta_{i,j}^{k,c}$ are satisfied we can compose a feasible (i, j, k, c)-labeling $\mathcal{L}$ from a compact (i, ℓ, k, a)-labeling $\mathcal{L}_{\text{left}}$ and a compact $(\ell, j, k-1, b)$-labeling $\mathcal{L}_{\text{right}}$ as sketched in Figure 5. The labels in $S \cap \{l_i, \ldots, l_\ell\}$ are placed as in $\mathcal{L}_{\text{left}}$, the labels in $S \cap \{l_{\ell+1}, \ldots, l_{j-1}\}$ as in $\mathcal{L}_{\text{right}}$ and l_j at $x_j + W_j$. The weights of the sub-labelings are chosen such that the weight of $\mathcal{L}$ correctly adds up to c. □

3.4. Label Number Maximization

In this section we present an algorithm to solve MAXLABELS, that is, to place as many labels as possible in K given rows. Note that Theorem 3.5 of the previous section directly yields a polynomial time algorithm if we set $w_i = 1$ for all $1 \leq i \leq n$. However, this would result in a running time of $O(Kn^5)$. Here we show that there is a faster algorithm which uses an exchange argument based on the fact that all labels have the same weight.

We first introduce an adapted notation for cardinality-maximal labelings. We define a *cm-(i, j, k)-labeling* to be a feasible labeling of a subset $\hat{P} \subseteq P_{i,j}$ in rows 1 to k with both l_i and l_j placed in row k such that there is no feasible labeling of another subset $\bar{P} \subseteq P_{i,j}$ with the same properties but $|\bar{P}| > |\hat{P}|$. We extend the meaning of *compact* to cm-(i, j, k)-labelings, that is, we say a say a cm-(i, j, k)-labeling $\mathcal{L}$ is compact if there is no other cm-(i, j, k)-labeling with smaller x-position.

We will compute two three-dimensional tables $\mathcal{T}$ and $\mathcal{K}$. An entry $\mathcal{T}[i, j, k]$ for $0 \leq i \leq j \leq n+1$ and $1 \leq k \leq K$ stores the x-position of a compact cm-(i, j, k)-labeling. We note that $\mathcal{T}$ can no longer contain the value ∞ since for any triple (i, j, k) there is always a feasible labeling with $\hat{P} = \{p_i, p_j\}$ and their labels placed disjointly in row k. An entry $\mathcal{K}[i, j, k]$ stores the actual cardinality of a cm-(i, j, k)-labeling.

The recursive definitions of $\mathcal{T}$ and $\mathcal{K}$ are as follows:

$$\mathcal{T}[i, j, k] = \begin{cases} x_i & \text{if } i = j \\ \min \Theta_{i,j}^k & \text{otherwise} \end{cases}, \tag{4}$$

with

$$\Theta_{i,j}^k = \left\{ \max\{x_j, \mathcal{T}[i, \ell, k] + W_j\} \;\middle|\; \begin{array}{c} i \leq \ell < j,\ \mathcal{T}[i, \ell, k] \leq x_j, \\ \mathcal{K}[i, j, k] = \mathcal{K}[i, \ell, k] + \mathcal{K}[\ell, j, k-1] - 1 \end{array} \right\}, \tag{5}$$

$$\mathcal{K}[i, j, k] = \begin{cases} 1 & \text{if } \quad i = j \\ \max \kappa_{i,j}^k & \text{otherwise} \end{cases}, \tag{6}$$

with

$$\kappa_{i,j}^{k} = \{\mathcal{K}[i,\ell,k] + \mathcal{K}[\ell,j,k-1] - 1 \mid i \leq \ell < j,\ \mathcal{T}[i,\ell,k] \leq x_j\}\,. \tag{7}$$

The dynamic programming algorithm MaxLabelsAlg for MAXLABELS computes $\mathcal{K}$ and $\mathcal{T}$ in a bottom-up fashion analogously to our previous algorithms. Note that for each triple i,j,k we first compute $\mathcal{K}[i,j,k]$ and then based on that $\mathcal{T}[i,j,k]$. The final solution contains $\mathcal{K}[0,n+1,K]-2$ labels and can be obtained by backtracking from $\mathcal{T}[0,n+1,K]$. We give the pseudo code for MaxLabelsAlg in Algorithm 3.

ALGORITHM 3: MaxLabelsAlg

```
initialize T
for i = 0 to n + 1 do
    for k = 1 to K do
        T[i, i, k] = x_i
        K[i, i, k] = 1
for k = 1 to K do
    for j = 1 to n + 1 do
        for i = 0 to j − 1 do
            K[i, j, k] = max κ^k_{i,j}
            T[i, j, k] = min Θ^k_{i,j}
reconstruct solution from T[0, n + 1, K]
```

THEOREM 3.6. MAXLABELS *can be solved by dynamic programming in* $O(Kn^3)$ *time.*

PROOF. The running time of the dynamic programming algorithm follows from the fact that the tables are both of size $O(Kn^2)$ and computing each entry consists of finding the minimum or maximum of a set of $O(n)$ elements.

The correctness proof follows exactly the same arguments about the decomposition of a cm-(i,j,k)-labeling $\mathcal{L}$ into two labelings $\mathcal{L}_{\text{left}}$ and $\mathcal{L}_{\text{right}}$ by splitting $\mathcal{L}$ at the predecessor l_ℓ of l_j in row k (Figure 5). The definition of $\Theta_{i,j}^{k}$ implies that a value $X_j^\ell = \max\{x_j, \mathcal{T}[i,\ell,k] + W_j\}$ is contained in the set if and only if it leads to a cm-(i,j,k)-labeling; this is achieved by requiring $\mathcal{K}[i,j,k] = \mathcal{K}[i,\ell,k] + \mathcal{K}[\ell,j,k-1] - 1$. Note that here we need to subtract 1 because label l_ℓ is counted twice otherwise. The set $\kappa_{i,j}^{k}$ contains the cardinalities of all feasible ($\mathcal{T}[i,\ell,k] \leq x_j$) compositions of a compact cm-(i,ℓ,k)-labeling and a compact cm-$(\ell,j,k-1)$-labeling. Entry $\mathcal{K}[i,j,k]$ is then simply the maximum value in $\kappa_{i,j}^{k}$.

The interesting aspect for showing the correctness is the following exchange argument. Assume that there is a cm-(i,j,k)-labeling $\mathcal{L}$ with predecessor l_ℓ of l_j but $\mathcal{T}[i,\ell,k] > x_j$, that is, the x-position of this labeling is not contained in $\Theta_{i,j}^{k}$. Since $\mathcal{T}[i,\ell,k] > x_j$ it follows that the sub-labeling $\mathcal{L}'$ for $P_{i,\ell}$ induced by $\mathcal{L}$ is not cardinality-maximal (recall that $\mathcal{T}[i,\ell,k]$ stores the smallest x-position among all cm-(i,j,k)-labelings). So in order to have l_ℓ as the predecessor of l_j some other label left of l_ℓ in row k must be removed from the selected labels, that is, we lose at least one label from S. But since all labels are worth the same, we can just as well remove l_ℓ itself from S and use label $l_{\ell'}$, the predecessor of l_ℓ in a compact cm-(i,ℓ,k)-labeling, as l_j's predecessor. Since $l_{\ell'}$ is the predecessor of l_ℓ in that labeling we know $\mathcal{T}[i,\ell',k] \leq x_\ell < x_j$, so that the x-position $X_j^{\ell'}$ is in fact contained in $\Theta_{i,j}^{k}$.

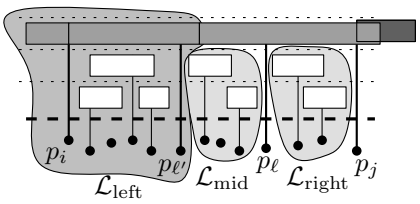


Fig. 7: Construction of a labeling for Theorem 3.6.

It remains to argue that this labeling with predecessor $l_{\ell'}$ is at least as good as $\mathcal{L}$, that is, $\mathcal{K}[i,j,k] \geq |\mathcal{L}|$. By the arguments presented above, we know that there is an (i,j,k)-labeling where $l_{\ell'}$ is the predecessor of l_j. Combining this with the definition of $\kappa^k_{i,j}$ we obtain:

$$\mathcal{K}[i,j,k] \geq \mathcal{K}[i,\ell',k] + \mathcal{K}[\ell',j,k-1] - 1 \tag{8}$$

Next, we consider the (i,ℓ,k)-labeling with $l_{\ell'}$ as predecessor of l_ℓ and ignore the label l_ℓ from it. We split this labeling into two parts. We denote the labeling for the points from p_i to $p_{\ell'}$ by $\mathcal{L}_{\text{left}}$, and the labeling for the points from $p_{\ell'+1}$ to $p_{\ell-1}$ by $\mathcal{L}_{\text{mid}}$; see Figure 7. Finally, there is a labeling $\mathcal{L}_{\text{right}}$ for the points from $p_{\ell+1}$ to p_{j-1}. It is easy to see that $|\mathcal{L}_{\text{right}}|$ is at most $\mathcal{K}[\ell,j,k-1]-2$. Now, consider both $\mathcal{L}_{\text{mid}}$ and $\mathcal{L}_{\text{right}}$. By removing the leader of p_ℓ separating $\mathcal{L}_{\text{mid}}$ from $\mathcal{L}_{\text{right}}$ we obtain

$$\mathcal{K}[\ell',j,k-1] \geq \mathcal{K}[\ell',\ell,k-1] + \mathcal{K}[\ell,j,k-1] - 2 \tag{9}$$

Combining equations (8) and (9) yields

$$\mathcal{K}[i,j,k] \geq \mathcal{K}[i,\ell',k] + \mathcal{K}[\ell',\ell,k-1] + \mathcal{K}[\ell,j,k-1] - 3 \tag{10}$$

Recall that ℓ' is the label in the (i,ℓ,k)-labeling that is the predecessor of l_ℓ in row k. Hence, by definition of $\mathcal{K}[i,\ell,k]$ it follows that $\mathcal{K}[i,\ell',k] + \mathcal{K}[\ell',\ell,k-1] - 1 \leq \mathcal{K}[i,\ell,k]$. Plugging this into equation (10) yields:

$$\mathcal{K}[i,j,k] \geq \mathcal{K}[i,\ell,k] - 1 + \mathcal{K}[\ell,j,k-1] - 1 \tag{11}$$

As we have already argued, the labeling that is induced by $\mathcal{L}$ for the points p_i to p_ℓ cannot be cardinality maximal. Hence, the labeling for those points can contribute at most $\mathcal{K}[i,\ell,k]-1$ labels to $\mathcal{L}$. The labeling induced by $\mathcal{L}$ for the remaining points can contribute at most $\mathcal{K}[\ell,j,k-1]-1$ labels. Finally, we can conclude

$$\mathcal{K}[i,j,k] \geq |\mathcal{L}| \tag{12}$$

This means that our algorithm indeed finds a labeling that has at least as many labels as $\mathcal{L}$. □

3.5. Two-sided Panorama Labeling

In this section we consider the two-sided panorama labeling problem. In this variation of panorama labeling we allow rows above and below the panorama we want to label; for an example see Figure 8. Unfortunately, by a result of Garrido et al. [2001, Theorem 3] it follows directly, that MINROW, MAXLABELS, and MAXWEIGHT are NP-hard in the two-sided panorama labeling model, even if the problems are restricted to one row per side.

THEOREM 3.7. *For two-sided panorama labeling the problems* MINROW*,* MAXLABELS*, and* MAXWEIGHT *are* NP*-hard.*

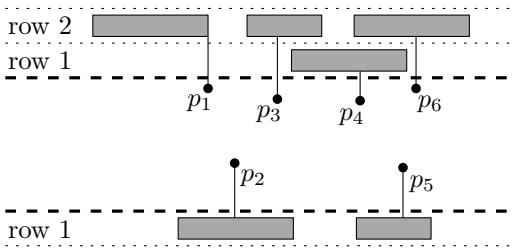


Fig. 8: Example for a two-sided panorama labeling.

For still finding a two-sided panorama labeling we propose in the following a mixed-integer linear (MILP) program formulation that can handle MINROW, MAXLABELS, and MAXWEIGHT.

Mixed-Integer Linear Programming.. In the following we give a MILP formulation that solves a given instance of two-sided panorama labeling. We begin by describing a MILP formulation for MAXLABELS, and proceed by sketching how to extend this formulation such that it can handle MAXWEIGHT and MINROW. For simplicity we assume, without loss of generality, that the x-coordinates of the input points are all non-negative. We can transform an instance with negative x-coordinates to an instance with strictly positive x-coordinates by simply shifting all points sufficiently far to the right.

Recall that for MAXLABELS we need to find the maximum number of labels which can be placed into K rows. For the two-sided case we denote the number of rows that are above the horizon by K_a and the number of rows below the panorama by K_b.

For each label l_i and each row $1 \leq r \leq K_\mathrm{a}$ above the horizon we introduce a binary variable a_i^r, and for each row $1 \leq r' \leq K_\mathrm{b}$ we introduce a binary variable $\mathrm{b}_i^{r'}$. The desired meaning of the variables is that, if a variable a_i^r ($\mathrm{b}_i^{r'}$) has value 1, then the label l_i is placed in row r (r'), and a_i^r ($\mathrm{b}_i^{r'}$) has value 0 if the label l_i is not placed in this row. Further, we introduce for every label l_i a continuous variable X_i that stores the x-position of l_i, if l_i is selected.

We are now ready to introduce the constraints of the MILP. First we introduce a constraint that ensures that a label is placed in at most one row. Then, we give two constraints which in combination guarantee that each label is connected to its leader; see requirement **(F1)** in Section 2. For each label l_i we require:

$$\sum_{1 \leq r \leq K_\mathrm{a}} \mathrm{a}_i^r + \sum_{1 \leq r' \leq K_\mathrm{b}} \mathrm{b}_i^{r'} \leq 1 \tag{13}$$

$$X_i \geq x_i \tag{14}$$

$$X_i \leq x_i + W_i \tag{15}$$

To generate a valid panorama labeling, we need to ensure that no two labels intersect (see requirement **(F3)**), that is, we require $X_i \leq X_j - W_j$ for every pair of labels l_i and l_j placed in the same row with $i < j$. However, this constraint must not influence the position of pairs of labels in different rows. To ensure this we introduce a large constant M which we use to "activate" or "deactivate" the constraint. This is a common trick used in formulating MILPs [Chen et al. 2011]. In the following, we set M to be the largest x-coordinate of the input points plus the largest width of all labels among the input. Since there is in principle no difference between placing labels above the horizon or below it we explicitly introduce the constraints only for placing labels *above* the

horizon in the following (constraints 16–18). The constraints for placing labels *below* the horizon are obtained by replacing each occurrence of a_i^r, a_j^ℓ, and K_a with b_i^r,b_j^ℓ, and K_b, respectively.

We introduce for each pair of labels l_i, l_j with $i < j$ and for every r, $1 \leq r \leq K_\mathrm{a}$, the following constraint:

$$X_i \leq X_j - W_j + (1 - \mathrm{a}_i^r)M + (1 - \mathrm{a}_j^r)M \tag{16}$$

Note that this constraint has only an effect on X_i and X_j if both a_i^r and a_j^r are set to 1, that is, both l_i and l_j are placed in the same row (then the constraint is the desired $X_i \leq X_j - W_j$). Otherwise, the value on the right-hand side of the equation is always larger than the value of the left-hand side.

We also need to guarantee that no label can intersect a leader; see requirement **(F2)**. First, we introduce a constraint that ensures that no label can intersect the leader of a label to its right. For a label l_i placed in row r we need the constraint $X_i \leq x_j$ for all labels l_j, $i < j$ whose labels are placed in rows above r. Making use of the constant M as above we introduce for every pair of labels l_i, l_j with $i < j$ and for every r, $1 \leq r \leq K_\mathrm{a}$, the following constraint:

$$X_i \leq x_j + (1 - \mathrm{a}_i^r)M + \left(1 - \sum_{r < \ell \leq K_\mathrm{a}} \mathrm{a}_j^\ell\right) M \tag{17}$$

Similarly, to ensure that the label l_i does not intersect the leader of a label l_j to its left ($j < i$) that is placed in a row above l_i we need the constraint $X_i - W_i \geq x_j$. We introduce for every pair of labels l_i, l_j with $j < i$ and for every r, $1 \leq r \leq K_\mathrm{a}$, the following constraint:

$$X_i - W_i \geq x_j - (1 - \mathrm{a}_i^r)M - \left(1 - \sum_{K_\mathrm{a} \geq \ell > r} \mathrm{a}_j^\ell\right) M \tag{18}$$

Constraints 17 and 18 influence X_i only if both $\mathrm{a}_i^r = 1$ (that is, the label l_i is placed in row r) and the sum of all a_j^ℓ, $\ell > r$ is 1 (that is, the label l_j is placed in a row above l_i).

Subject to all introduced constraints we maximize the objective function:

$$\sum_{1 \leq i \leq n} \left(\sum_{1 \leq r \leq K_\mathrm{a}} \mathrm{a}_i^r + \sum_{1 \leq r' \leq K_\mathrm{b}} \mathrm{b}_i^{r'} \right)$$

Extending the above formulation such that it can handle MAXWEIGHT can be done straightforwardly by modifying the objective function slightly to $\sum_{1 \leq i \leq n}((\sum_{1 \leq r \leq K_\mathrm{a}} \mathrm{a}_i^r + \sum_{1 \leq r' \leq K_\mathrm{b}} \mathrm{b}_i^{r'}) \cdot w_i)$.

We can extend the above formulation such that it can solve MINROW by adding additional constraints and modifying the objective function. Recall that for MINROW the number of rows necessary to find a solution is bounded by $\lceil n/2 \rceil$. Hence, we can set $K_\mathrm{a} = K_\mathrm{b} = \lceil n/2 \rceil$. Since for this problem all labels need to be placed we need to modify constraint 13 to the following.

$$\sum_{1 \leq r \leq K_\mathrm{a}} \mathrm{a}_i^r + \sum_{1 \leq r' \leq K_\mathrm{b}} \mathrm{b}_i^{r'} = 1 \tag{19}$$

Further, we add a binary variable z_r for each row above the horizon and a binary variable z'_r for each row below it. The desired meaning of the newly added variables is

that if a row r contains at least one label, then, the corresponding variable z_r is 1. For each label l_i we require

$$z_r \geq \mathrm{a}_i^r \qquad 1 \leq r \leq K_\mathrm{a}, \tag{20}$$

$$z'_r \geq \mathrm{b}_i^{r'} \qquad 1 \leq r' \leq K_\mathrm{b}. \tag{21}$$

Finally, the objective function we want to minimize is $\sum_{r \leq K_\mathrm{a}} z_r + \sum_{r \leq K_\mathrm{b}} z'_r$. Since we minimize the objective function, we ensure that every z_r is set to 0 if it is possible, that is, if the row r does not contain a label. Note that although this formulation minimizes the number of rows in which the labels are placed, it might contain empty rows which can be removed in a post-processing step.

4. EXTENSIONS

In this section we sketch several extensions to our algorithms. With some extensions we aim at handling additional aesthetic constraints, for example, to ensure that two labels or a label and a leader of another label do not come too close. With this particular constraint the problem MINROW can become infeasible if the distance between two input points is too small. Similar problems can arise with other extensions. We do not address these problems in detail; often they can be revealed and possibly resolved in a pre-processing step.

4.1. Label Spacing

Our dynamic programming algorithms ensure that no two labels intersect and that no label intersects a leader other than its own. It is possible, however, that two labels (or a label and a leader) get arbitrarily close to each other. At first glance there seems to be an obvious solution to the problem: simply enlarge the labels by an arbitrary buffer amount. However, this can result in labels that are disconnected from their leaders. To see this consider a label that is in its leftmost/rightmost position. Since we increased the width of the label its x-position may be to the left of its own leader, which results in a label that is disconnected from its leader. Hence, we propose a different approach. To avoid this problem, we add the requirement that the horizontal distance between two labels in the same row (or between a label and the leader of a label in a higher row) must not be smaller than a user-defined value $\varepsilon \geq 0$; see Figure 9. MinRowAlg ensures this requirement if we add $+\varepsilon$ to the definition of equation (1) at the appropriate positions. We need it to ensure that no label l_ℓ is considered for which the distance between the right border of l_i and the feature of l_ℓ has distance less than ε. This is ensured by requiring $\mathcal{T}[i,\ell,k] + \varepsilon \leq x_j$. Further, we need that if a label l_ℓ is placed right of l_i in the k-th row, it has distance of at least ε. We obtain this by requiring $\max\{x_j, \mathcal{T}[i,\ell,k] + W_j + \varepsilon\}$. This modification of equation (1) yields

$$\Theta_{i,j}^k = \{\max\{x_j, \mathcal{T}[i,\ell,k] + W_j + \varepsilon\} \mid i \leq \ell < j,\, \mathcal{T}[i,\ell,k] + \varepsilon \leq x_j,\, \mathcal{T}[\ell,j,k-1] < \infty\}\,.$$

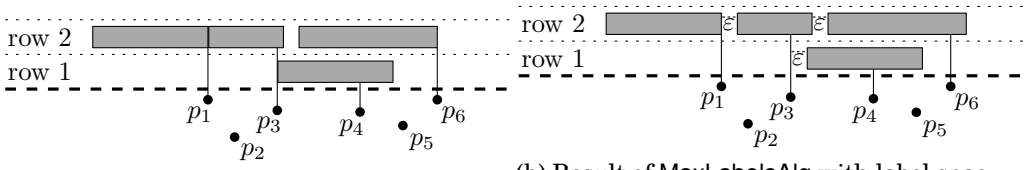
For MaxWeightAlg similar modifications are necessary. We modify equations (2) and (3). The first equation is for the recurrence relation for the first row (that is, $k = 1$)

$$\Theta_{i,j}^{1,c} = \{\max\{x_j, \mathcal{T}[i,\ell,1,a] + W_j + \varepsilon\} \mid i \leq \ell < j,\, \mathcal{T}[i,\ell,1,a] + \varepsilon \leq x_j,\, a = c - w_j\}$$

and for $k > 1$

$$\Theta_{i,j}^{k,c} = \left\{\max\{x_j, \mathcal{T}[i,\ell,k,a] + W_j + \varepsilon\} \mid \begin{array}{c} i \leq \ell < j,\, \mathcal{T}[i,\ell,k,a] + \varepsilon \leq x_j, \\ \mathcal{T}[\ell,j,k-1,b] < \infty,\, a = c - b + w_\ell \end{array}\right\}.$$

Finally, the modifications required for MaxLabelsAlg are in principle identical to those desribed before. We need to modify the equations (5) and (7) yielding:



(a) Result of MaxLabelsAlg.

(b) Result of MaxLabelsAlg with label spacing.

Fig. 9: Result of MaxLabelsAlg of a MAXLABELS instance for two rows with and without label spacing.

$$\Theta^k_{i,j} = \left\{ \max\{x_j, \mathcal{T}[i,\ell,k] + W_j + \varepsilon\} \;\middle|\; \begin{array}{c} i \le \ell < j,\ \mathcal{T}[i,\ell,k] + \varepsilon \le x_j, \\ \mathcal{K}[i,j,k] = \mathcal{K}[i,\ell,k] + \mathcal{K}[\ell,j,k-1] - 1 \end{array} \right\},$$

$$\kappa^k_{i,j} = \{\mathcal{K}[i,\ell,k] + \mathcal{K}[\ell,j,k-1] - 1 \mid i \le \ell < j,\ \mathcal{T}[i,\ell,k] + \varepsilon \le x_j\}.$$

4.2. Horizontal Space Constraints

In Section 3 we gave algorithms to compute labelings that use a small number of rows but we did not care about space consumption in the horizontal direction. If we want to ensure, for example, that no label intersects the left or right border of the panorama, we have to restrict the sliding of labels. This can be done by changing the definition of the two dummy points p_0 and p_{n+1} in Section 3.2, that is, we set $x_0 = x_{\min}$ and $x_{n+1} = x_{\max}$, where $x_{\min}$ ($x_{\max}$) is the smallest (largest) allowed x-coordinate for the left (right) boundary of a label. Then, a $(0, n+1, k)$-labeling with the minimum feasible value for k is an optimal solution to the horizontally constrained version of MINROW.

Note that in this case Lemma 2.2 is not true anymore. There are MINROW instances for which more than $\lceil n/2 \rceil$ rows are necessary, and there are even instances for which no solution is possible. However, it is easy to see that no MINROW instance that has a solution requires more than n rows. Hence, the worst-case time complexity of MinRowAlg is still in $O(K^\star n^3)$, where $K^\star = O(n)$. If an instance of MINROW has no valid solution, then $\mathcal{T}[0, n+1, n] = \infty$. The algorithms MaxLabelsAlg and MaxWeightAlg can be adapted by the same technique.

4.3. 360°-panoramas

We consider a $360°$-panorama as an image curved around the inside of a cylinder, thus it does not have a left or right boundary. We can extend our algorithm for MINROW to handle such images. Our extension is based on the following observation: if there is a solution to MINROW with k rows, there is also a solution with k rows that contains one of the labels l_i at position $X_i = x_i$, $Y_i = k$. A trivial approach is thus to solve the problem for $i = 1, 2, \ldots, n$, each time splitting the cylindric image at x_i to obtain a conventional image. Since we have to solve the problem n times the asymptotic running time increases by one order of magnitude. With a simple change, however, *one* execution of our MINROW-algorithm suffices.

Our idea is to extend table $\mathcal{T}$, which so far contained values $\mathcal{T}[i,j,k]$ only for $i \le j$. The new version of the algorithm computes values $\mathcal{T}[i,j,k]$ *for all* $1 \le i, j \le n$ and $1 \le k \le K^\star$, where $K^\star$ is the smallest feasible number of rows. If $j < i$, a labeling corresponding to $\mathcal{T}[i,j,k]$ simply contains labels $l_{i+1}, \ldots, l_n$ and $l_1, \ldots, l_j$. To ensure that the solutions for $l_{i+1}, \ldots, l_n$ and $l_1, \ldots, l_j$ are compatible we need to transform the x-coordinates of points p_ℓ with $\ell > i$ to $x_\ell - x_{\max}$, where $x_{\max}$ is the width of the

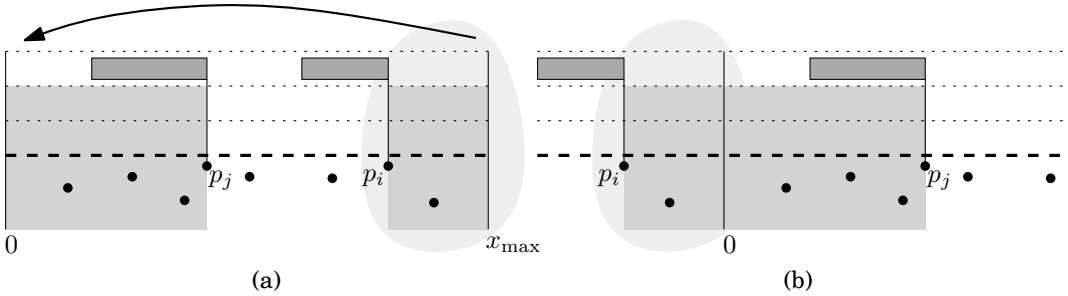


Fig. 10: Example of $360°$-panoramas before (a) and after (b) transformation.

panorama. This transformation essentially moves the points p_ℓ to the left of $x = 0$; see Figure 10. We also need to transform the appropriate values in $\mathcal{T}$ by the same method.

We compute $\mathcal{T}[i, j, k]$ as before, ignoring the fact that in a $360°$-panorama l_i and l_j may intersect in the interval bounded to the left by x_j and to the right by x_i. If our algorithm determines $\mathcal{T}[i, j, k] < \infty$ we need to ensure that there exists a feasible solution where l_i and l_j do not intersect. This, however, can be done by a simple sweep over all labels that are placed in row k in the solution. With this change, MinRowAlg still needs $O(K^\star n^3)$ time. The algorithms MaxLabelsAlg and MaxWeightAlg can be adapted the same way.

4.4. Elite Labels

For the most important sites, for example, the Willis Tower in the panorama of Chicago (see Figures 1 and 12), we may want to guarantee that the solution contains the corresponding labels. We term such a label an *elite label*. Now we can ask for a feasible labeling that contains all elite labels plus as many other labels as possible in a given number k of rows. Assuming that such a solution exists, we can apply our algorithm for MAXWEIGHT. We simply have to set the weight of each elite label to a large enough number (for example, $n + 1$) and the weight of each non-elite label to one. With this approach we have $w_{\text{total}} \in O(n^2)$, thus the algorithm requires $O(kn^7)$ time. With the following simple modifications of recurrences (6) and (4), however, we can reuse our $O(kn^3)$-time algorithm for MAXLABELS.

Each $(i, j, 1)$-labeling with $j > i$ contains a label l_ℓ preceding l_j. We have to avoid that we omit an elite label between l_i and l_j. To do so, we exchange the condition $i \leq \ell < j$ in the definitions of $\kappa^1_{i,j}$ and $\Theta^1_{i,j}$ with $e \leq \ell < j$, where l_e is the last elite label among $l_{i+1}, \ldots, l_{j-1}$. Obviously, we do not lose any feasible solution with this. Moreover, the modification suffices since our algorithm always constructs a solution based on feasible one-row solutions for smaller (sub-)instances. Similarly we can extend our algorithm for MAXWEIGHT to find a labeling in k rows containing all elite labels plus other labels of maximum total weight.

4.5. Optimizing Label Positions

A secondary criterion for aesthetically pleasing panorama labelings is that the labels are as close as possible to being centered above their respective sites. So far we did not consider this criterion in our problem definitions. In fact, the dynamic programming algorithms described in Section 3 place all labels at their leftmost feasible positions. We can remedy this unwanted side effect by subsequently fine-tuning the horizontal label positions. We traverse each row k of a given feasible labeling from right to left. Let S_k be the set of labels in row k. The general idea is that during the traversal, we shift a label to the right if it decreases the overall cost $H_k = \sum_{l_i \in S_k} |\Delta_i|$, where $\Delta_i = x_i - (X_i - W_i/2)$,

which means that $|\Delta_i|$ is the horizontal distance between the center $X_i - W_i/2$ of label l_i and the feature point p_i. Note that a right shift of a label l might require a right shift of the label directly to the right of l.

Since this approach does not change the row assignment of the labels we only need to ensure that while shifting the labels no label intersects another label or leader. This suffices to guarantee that the labeling remains valid. In the following we describe a simple $O(n^2)$-time algorithm that minimizes H_k by centering the labels as much as possible above their respective feature points. The general idea is to identify maximal sets of touching labels in the same row called *chains* and to repeatedly move subsets of the chains to the right.

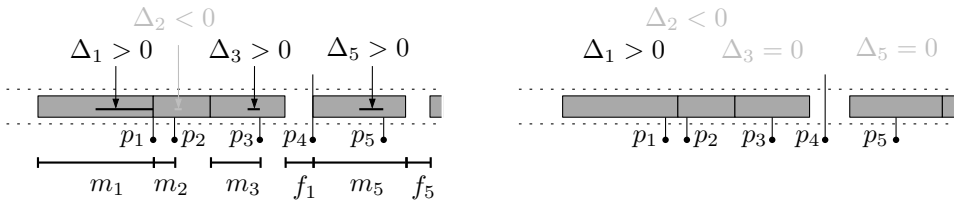
In the first step of the algorithm we assign each label to a chain. Two labels that are in the same row are assigned to the same chain if their borders touch each other. If a label touches no other label it defines a singleton chain. We say that the *suffix* of a label l_i consists of l_i and all labels in its chain that are to the right of l_i.

The general idea behind our algorithm is to iteratively move one suffix at a time to the right in order to improve the value of the objective function H_k. We distinguish between four types of *events* which may occur while moving a suffix: (i) a label in the suffix becomes centered above its feature, (ii) a label in the suffix is moved to its rightmost position, (iii) the right border of the rightmost label in the suffix touches the leader of another label in a row above, or (iv) the rightmost label in the suffix touches a label that belongs to another chain. The first type of event occurs when a positive Δ_i becomes 0, and the second type occurs when the horizontal distance m_i between a label's left border and its feature becomes 0, that is, $m_i = x_i - (X_i - W_i) = 0$. Since the row assignment of the solution is fixed we can easily determine in a pre-processing step for each label whether it has to its right a label, a leader or nothing. This allows us to determine in a later step the distance f_i that the suffix of l_i can be moved until an event of the third or fourth type occurs. For an illustration of these concepts see Figure 11.

Now, in a single row, the algorithm traverses the chains from right to left beginning with the rightmost chain. In each chain it performs a second-level traversal from right to left. In this second-level traversal our algorithm determines for each label l_i the values for Δ_i, m_i, and the difference d_i between the number of positive and non-positive offsets Δ_j in its suffix. Note that if d_i of a suffix is positive, it means moving the suffix to the right decreases H. The algorithm also computes for each label l_i how far its suffix can be moved to the right until the first event occurs by determining $M_i = \min\{\Delta_j, m_j, f_i \mid \Delta_j > 0, l_j \text{ in suffix of } l_i\}$.

Finally, we need to select the suffix of the currently considered chain that we actually move to the right. After determining the value of d_i for each label we choose the label with maximum positive d_i. If there is more than one label with maximum d_i we choose the left-most label among them. If no label has a positive d_i we move on to the next chain since a shift to the right would increase H_k. Now, consider we have selected a label l_i with maximum positive d_i in its chain. We move the whole suffix of l_i by M_i to the right. Note that if l_i is not the leftmost label in its chain, this splits the chain into two separate chains. Further, if after moving the suffix of l_i the rightmost label of the suffix touches the left-most label of another chain (that is, an event of the fourth type has occurred) we have merged two independent chains into a new one. If, after moving l_i's suffix, the suffix cannot be moved further to the right (that is, an event of type (ii) or (iii) has occurred) we move on to the next chain directly left of the current chain. Otherwise, we repeat the second-level traversal for the current chain.

Note that the output of our dynamic programming algorithms described in Section 3 ensures that no label can initially be moved to the left. Now, to see that the algorithm minimizes $H = \sum_{l_i \in S} |\Delta_i|$ consider a single row. It is clear that after a chain has been



(a) Before applying our post-processing algorithm.

(b) After applying the post-processing algorithm.

Fig. 11: Illustrations for our post-processing algorithm. In both illustrations two chains are shown.

considered by the algorithm it does no longer contain a suffix that would decrease H if moved further to the right. Since this is true for all chains of a each row the correctness of the algorithm follows. It remains to argue that the running time is $O(n^2)$. Consider again a single row, say the k-th row (which contains $|S_k|$ labels). Determining the values of Δ_i, d_i, m_i and M_i for each label $l_i \in S_k$ can be done in constant time per label. Every time we move a label at most $O(|S_k|)$ labels in its suffix are moved with it. Since for each label at most three events can occur there are at most $3|S_k|$ move operations, each of which requires $O(|S_k|)$ time. Hence, for the k-th row our algorithm requires $O(|S_k|^2)$ time. The total running time of our algorithm is thus $O(\sum_k |S_k|^2) = O(n^2)$.

4.6. Minimizing the Total Leader Length

Since long leaders produce visual clutter we may want to place all labels while minimizing the total length of their leaders. Our algorithm for MAXWEIGHT can be modified to solve this problem. We simply have to define the weight w_i for a label l_i depending on the row in which we place l_i. We define $w_i = n + 1 - Y_i$, which implies that a larger weight is assigned to positions closer to the horizon. Our setting ensures $w_i > 0$ since n is an upper bound on the number of rows required for an optimal solution. Using our algorithm for MAXWEIGHT we can solve the problem in $O(n^5 w_{\text{total}}^2) = O(n^9)$ time.

4.7. Fixed Order k-position Feature Labeling

Sometimes we do not want to label one distinct point, but we rather want to label a feature that is represented by an area (for example, a building) where each point inside this area is equally representative of the feature we want to label. The flexibility of choosing a representative feature point might help increase the quality of the labeling (for example, fewer rows are required to display all labels). Since in our setting the problem is defined only by the x-coordinates of the feature points, each area feature reduces to a horizontal line segment or interval spanning the horizontal width of the area. Note that these intervals may overlap. However, in most labeling instances the objects to be labeled have a natural fixed order (for example, the order of buildings is implicitly given by their centroids), and a labeling should reflect this order. Hence, we may assume that there is a given total order on the feature points.

Unfortunately, it is difficult to adapt our algorithms to handle feature points that can freely move along a horizontal line segment. We can, however, solve a slightly less general variant of this problem. If instead of line segments, we restrict the possible positions of each feature to z distinct points we can solve MINROW, and MAXLABELS in polynomial time with respect to n and z.

More specifically, for each label l_i we are given a set of z points $p_i^1 = (x_i^1, -1), \ldots, p_i^z = (x_i^z, -1)$ with x-coordinates $x_i^1, \ldots, x_i^z$ (for simplicity we assume that all y-coordinates

are set to -1). Further, to reflect the order of the features we aim to label, we are also given a total order on the feature points of the labels. More specifically, the feature of a label l_i is always placed to the left of l_j's feature for all $j > i$. Now, we are ready to describe how to adapt MinRowAlg. First, we extend the definition of a compact (i, j, k)-labeling to a compact (i, g, j, h, k)-labeling, where $1 \leq g, h \leq z$, p_i^g and p_j^h are the two feature points for l_i, and l_j, respectively. The definition of $\mathcal{X}_{i,g,j,h,k}$ can be extended the same way. We increase the size of the table $\mathcal{T}$ by two dimensions. An entry $\mathcal{T}[i, g, j, h, k]$ stores the x-position $\mathcal{X}_{i,g,j,h,k}$ of a compact (i, g, j, h, k)-labeling, and some additional backtracking information. Each entry $\mathcal{T}[i, g, j, h, k]$ is obtained by computing $\min \Theta^k_{i,g,j,h}$, where $\Theta^k_{i,g,j,h}$ for $1 \leq i < j \leq n$, $1 \leq g, h \leq z$, and $k \geq 1$ is defined as the set

$$\Theta^k_{i,g,j,h} = \left\{ \max\{x_j^h, \mathcal{T}[i, g, \ell, f, k] + W_j\} \;\middle|\; \begin{array}{c} i \leq \ell < j, \\ \mathcal{T}[i, g, \ell, f, k] \leq x_j^h, \\ \mathcal{T}[\ell, f, j, h, k-1] < \infty, \\ 1 \leq f, g, h \leq z, \\ x_i^g \leq x_l^f < x_j^h \end{array} \right\}.$$

The correctness follows by the same argumentation as in the proof of Theorem 3.3. The running time of this algorithm is $O(K^\star n^3 z^3)$. For small values of z the factor z^3 can still be considered a constant.

The extension of MaxLabelsAlg, and MaxWeightAlg to handle multiple possible positions can be done by following the same basic idea.

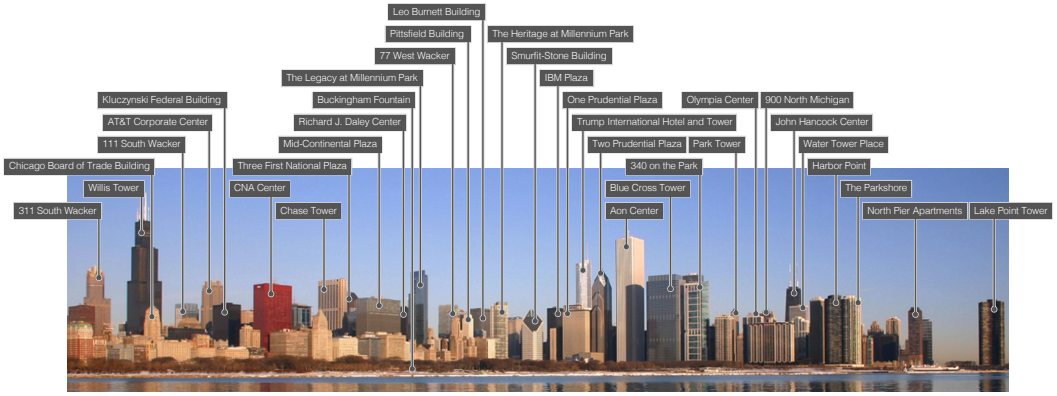
5. EXPERIMENTAL RESULTS

In this section we evaluate our algorithms with respect to running time and the number rows used or number of placed labels. We implemented the algorithms in C++ using GTK+ and Cairo for the visual output. We tested the algorithms both for real-world instances as well as randomly generated instances.

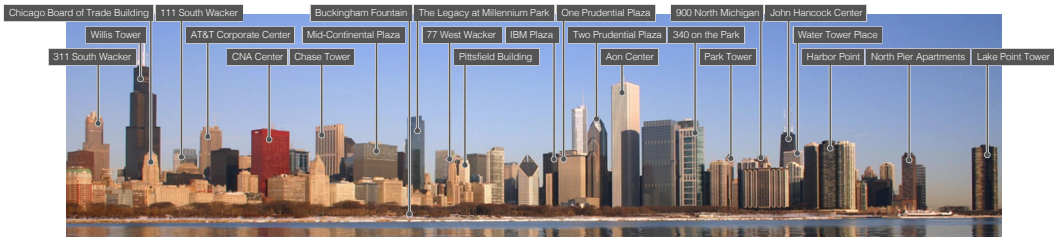
5.1. Case Study

Figure 12 shows the results of a case study with the Chicago skyline using three of our algorithms. The input data for all three figures consists of the same 33 labels. On a laptop clocked at 2.4 Ghz it took roughly 1ms to compute the panorama labelings in Figures 12a and 12b and about 160ms to compute the labeling shown in Figure 12c. We used $\varepsilon = 10$px as label spacing, and applied the *optimize label position* extension.

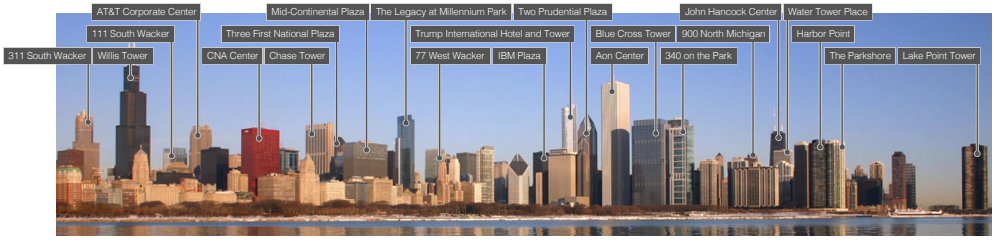
Figure 12a shows the solution of the MINROW algorithm that requires ten rows. We observe that the solution is aesthetically not appealing. In the center of the panorama the arrangement of labels is reminiscent of the worst-case scenario illustrated in Figure 3. Figure 12b shows the solution of the MAXLABELS algorithm for three rows. Due to restricting the number of label rows, the result is much more pleasing than the MINROW solution (12a). Of all 33 labels 24 are displayed. This indicates that only a few labels of densely placed points are responsible for the visually unpleasing MINROW result. Figure 12c shows the solution of the MAXWEIGHT algorithm for three rows. We divided the buildings into four equal-size classes based on their height, that is, the $\lceil n/4 \rceil$ tallest buildings have weight 8, and the other classes have weights $4, 2,$ and 1 accordingly. This yields the maximum total weight of all labels $w_{\text{total}} = 128$. Of the 33 labels 23 labels are displayed and they have a total weight of 114. Although the two labelings in Figures 12b and 12c look similar at first sight, note that for instance the MAXWEIGHT solution contains a label for "Trump International Tower", the third tallest building in the panorama, while the MAXLABELS solution misses that label. The MAXLABELS solution achieves a total weight of 98.



(a) MINROW solution.



(b) MAXLABELS solution.



(c) MAXWEIGHT solution.

Fig. 12: Case study with the Chicago skyline using three of our algorithms. Photography: ©J. Crocker (http://en.wikipedia.org/w/index.php?title=File:2010-02-19_16500x2000_chicago_skyline_panorama.jpg).

5.2. Performance Evaluation

We executed the experiments on a single core of an Intel Xeon E5-2670 processor that is clocked at 2.66 GHz. The machine is running Linux 3.4.28-2.20 and has 64 GiB of RAM. We compiled our C++ implementation with GCC 4.7.1, using optimization level 3.

For each set of test parameters we generated 1,000 test instances with a width of 1280 pixels. All input points have integer x-coordinates between 0 and 1279. These coordinates were chosen uniformly at random. The label widths were randomly chosen from a Gaussian distribution with mean 108.52 and standard deviation $\sigma = 44.72$. We obtained these values by taking 298 names of the world's tallest buildings from

input	running time [ms]			rows		
#labels	min.	avg.	max.	min.	avg.	max.
10	< 1	< 1	< 1	1	2	4
25	< 1	< 1	< 1	4	5	12
50	1	2	3	10	17	25
75	4	8	11	16	35	38
100	20	31	36	28	49	50
125	55	90	98	36	62	63
150	181	198	200	74	75	75

Table I: Performance of our implementation of MINROW for instances with 10–150 labels.

Wikipedia[1], and determined the width of the names in pixels with the font `Helvetica` at size 12.

MINROW.. The first set of our experiments focuses on our implementation of MinRowAlg. Table I reports the minimum, average, and maximum running time for labeling instances with varying numbers of labels. We observe that even for instances where the points lie unreasonably dense (for example, instances with 150 labels where the average distance between points is roughly 8 pixels) our algorithm solves all instances in less than 0.2s. Interestingly, the majority of instances with 75 or more labels is formed by worst-case instances for MINROW, that is, they require $\lceil n/2 \rceil$ rows. Thus for reasonable real-world instances we can expect that our algorithm generates an optimal solution near-instantaneously.

MAXLABELS.. In the second set of experiments we investigate the running times of our implementation of MaxLabelsAlg. We believe that in most practical application no more than four rows of labels are used and hence ran the first part of experiments for $K = 4$. We observe from the data in Tables II that our algorithm performs again very well, even for large instances. In this case it even outperforms the previous MINROW algorithm. This is mainly due to the fact that many instances are worst-case instances for the MINROW algorithm. Note that the number of placed labels for $K = 4$ seems very high. This can only happen when there are many relatively narrow labels.

Our algorithm is also able to quickly generate solutions for less realistic instances where we allow the labels to be placed in at most 50 rows, see Table III. We observe that even in the worst-case scenario, that is, trying to place 150 labels in at most 50 rows, the execution time is less than 250ms. Unsurprisingly, the algorithm nearly always succeeds in placing all or at least the vast majority of labels in this case.

MAXWEIGHT.. Next, we evaluate our MAXWEIGHT algorithm. For this we use two different weight distributions. In some use cases it might be useful to define a ranking of the label importance. Then, for a set of n labels, we assign each label a distinct integer weight between 1 and n. In all generated instances the weights were distributed uniformly at random. A different way of determining label weights is to group labels into classes of equal importance. Generally, we can expect that there is only a limited number of such classes. In the experiment we defined four classes and the labels were assigned to the classes uniformly at random. Labels in class i have weight 2^i.

We report the running times of our implementation of MaxWeightAlg for both types of weight distributions in Tables IV and V. These results were again generated using

[1] http://en.wikipedia.org/wiki/List_of_tallest_buildings_in_the_world

input	running time [ms]			#placed labels		
#labels	min.	avg.	max.	min.	avg.	max.
10	< 1	< 1	< 1	10	10	10
25	< 1	< 1	< 1	21	23	25
50	< 1	< 1	2	34	36	39
75	2	2	6	38	43	49
100	5	5	10	43	48	54
125	9	10	16	46	51	55
150	16	17	17	50	54	59

Table II: Performance of our implementation of MAXLABELS with $K = 4$ rows for instances with 10–150 labels.

input	running time [ms]			#placed labels		
#labels	min.	avg.	max.	min.	avg.	max.
10	< 1	< 1	< 1	10	10	10
25	< 1	1	2	25	25	25
50	7	8	14	50	50	50
75	25	26	32	75	75	75
100	65	66	67	99	99	100
125	132	135	141	119	121	125
150	233	237	243	140	143	149

Table III: Performance of our implementation of MAXLABELS with $K = 50$ rows for instances with 10–150 labels.

$K = 4$ rows. Note that the measured execution times in these tables are reported in seconds and not milliseconds as before. We also give the number of labels placed.

input	running time [s]			#placed labels		
#labels	min.	avg.	max.	min.	avg.	max.
10	< 0.01	< 0.01	< 0.01	10	10	10
25	0.25	0.33	0.42	21	23	25
30	0.57	0.95	1.37	24	26	29
40	4.65	5.94	7.69	28	30	34
50	23.63	27.86	33.64	32	34	37
75	308.96	394.51	465.56	38	40	44

Table IV: Performance of our implementation of MAXWEIGHT for instances with 10–75 ranked labels.

Since the algorithm has a pseudo-polynomial running time the higher execution times compared to our other algorithms were expected. Although the results reported in both tables confirm this expectation, we observe that for small and medium numbers of rows and labels, the algorithm still runs within an acceptable time frame. However, if we raise the number of available rows substantially or increase the total number of labels, the execution time grows quickly and may become unacceptable in practice.

input	running time [s]			#placed labels		
#labels	min.	avg.	max.	min.	avg.	max.
10	< 0.01	< 0.01	0.01	10	10	10
25	0.07	0.11	0.15	22	23	25
30	0.12	0.20	0.34	24	26	29
40	0.64	0.80	1.02	25	30	34
50	1.98	2.56	3.30	30	33	36
75	11.18	15.75	19.68	36	39	43
100	39.73	55.20	67.82	37	40	43

Table V: Performance of our implementation of MAXWEIGHT for instances with 10–100 labels in four importance classes with weights $\{1, 2, 4, 8\}$.

input	running time [s]			#placed labels		
#labels	min.	avg.	max.	min.	avg.	max.
10	< 0.01	< 0.01	0.01	10	10	10
25	0.01	0.03	0.1	24	24	25
30	0.03	11.96	90.78	27	29	30
40	1.86	396.48	> 600	-	-	-

Table VI: Performance of our MILP-implementation for the two-sided MAXLABELS problems for instances with 10–40 labels and $K_{\mathrm{a}} = 3$ and $K_{\mathrm{b}} = 1$. We have omitted the number of placed labels in the last row since not all instances were finished within the time limit of 10 minutes.

Two-sided panorama labeling.. Finally, we have also implemented the MILP formulation for MAXLABELS for the two sided case in C++ using the MILP solver Gurobi 5.60. We set the number of rows above the panorama $K_{\mathrm{a}} = 3$ and the number of rows below the panorama to $K_{\mathrm{b}} = 1$, since in most visual depictions of such panorama labelings, the majority of labels is above the horizon and only a minority is placed below the picture.

For small instances (of at most 25) labels, the optimum solution can be obtained quickly, such that it is sufficient for real-time applications, most of the time. However, increasing the number of labels to 30 already requires several seconds on average. Increasing the number of labels to 40 yields running times well above the 10 minute mark, which we set as timeout; see Table VI for detailed results.

Extensions.. Here, we give a brief evaluation of some of the extension to our algorithms which we described in Section 4.

Unsurprisingly, the label spacing extension has no significant impact on the running time of the algorithms. For sensible values (for example, $\varepsilon = 10$) the impact on the number of displayed labels, or rows necessary to display all labels is negligibly.

Since we aim to produce visually appealing panorama labelings, we also implemented the algorithm that tries to optimize the position of each label in a post-processing step described in Section 4.5. The implementation is straightforward and for the tested instances the running time is even on the largest instances consisting of 150 labels less than 1ms. Since this step requires only little time we recommend to automatically apply this algorithm after computing a solution by one of our algorithms.

The last extension we implemented is the fixed order k-position feature labeling described in Section 4.7. We have implemented this extensions for both MinRowAlg

input	kPosMinRowAlg		MinRowAlg		kPosMaxLabelsAlg		MaxLabelsAlg	
#labels	∼time	∼rows	∼time	∼rows	∼time	∼labels	∼time	∼labels
10	< 1	2	< 1	2	1	10	< 1	10
25	1	5	1	5	12	23	< 1	23
50	24	17	2	17	53	36	< 1	36
100	660	49	31	49	516	51	5	48
150	3363	75	198	75	1832	57	17	54

Table VII: Evaluation of our implementation of kPosMinRowAlg and kPosMaxLabelsAlg algorithms with instances consisting of 10–150 labels. The average time in ms and the average number of rows/labels is reported.

as well as MaxLabelsAlg and we call the respective algorithms kPosMinRowAlg and kPosMaxLabelsAlg. For MinRowAlg and kPosMinRowAlg we compare the number of rows necessary to display all labels, and the time required for the algorithms to obtain the solution. For kPosMaxLabelsAlg we consider the number of labels the algorithm was able to place into $K = 4$ rows compared to the standard MaxLabelsAlg. We used the same generated instances as before, but added for each label four additional possible feature points evenly spaced at 5 pixels. Two of the new possible feature point positions were to the left and two were to the right of the initial feature point position. Please note that we do not claim that the instances we tested resemble real-world data, but are only used to give a rough intuition on the algorithm's performance. We report the result of both implementations in Table VII. For convenience we also repeat the results of the original MinRowAlg and the MaxLabelsAlg.

Unfortunately, the results of the of the k-position MinRowAlg are not promising. The algorithm requires significantly more time than the original MinRowAlg algorithm and the number of rows necessary to display all labels decreases on average only slightly.

The results of kPosMaxLabelsAlg are slightly more promising. We observe that the number of labels the algorithm is able to place in the 4 rows is slightly higher than in the standard MaxLabelsAlg algorithm (except for the small instances consisting of only 10 labels, where all labels can be always placed). Unfortunately, this comes with a significant increase in running time, but the algorithm might still be suitable for real-time applications for realistic instances. Usually, we expect less than 50 labels in a realistic instance which yields running times between 10 and 60ms.

6. CONCLUSION

We have presented polynomial time and practically fast algorithms for label placement using a new boundary-labeling model that allows multiple rows of sliding unit-height rectangular labels. In this model, each label is connected with its associated point by a vertical line segment. We have presented an $O(K^\star n^3)$-time algorithm for the basic problem MINROW, which minimizes the number of rows needed to place all labels. If the labeling is restricted to use at most K rows, however, we cannot generally place all labels. Therefore, we have investigated the problem MAXLABELS, which aims at maximizing the total number of labels in K rows, and the problem MAXWEIGHT, which aims at maximizing the total weight of labels in K rows. While MAXLABELS can be solved in $O(Kn^3)$ time, MAXWEIGHT turned out to be weakly NP-hard, yet can be solved by a pseudo-polynomial $O(Kn^3 w_{\text{total}}^2)$-time algorithm, where w_{total} is the total weight of all input labels. For the case of two-sided panorama labeling, we have proposed simple mixed-integer linear programming formulations.

According to our experiments, the algorithms for MINROW and MAXLABELS are very fast for instances typically arising in practice, that is, they solve instances with

up to 150 labels in less than one second. The algorithm for MAXWEIGHT is fast if the weights are not too large. For example, with integer weights between 1 and 8, we can solve instances with 50 labels in less than two seconds. We think that our setting is realistic, since labeled images already with more than 50 labels quickly appear visually cluttered and more than 150 labels seems unrealistic in most cases. Similarly, if sites are assigned importance levels, there are usually few of them (for example, main landmarks, distinctive buildings, public buildings, other). We conclude that our dynamic programming algorithms can quickly produce visually pleasing labelings of real-world panorama images. For the two-sided panorama labeling our MILP formulation is fast only for relatively small instances.

Although we considered several optimization problems in the context of panorama labeling, there are still many unanswered questions. For some applications it is of interest to investigate panorama labeling with labels that take up more than one row, which may be relevant for features with long names. As argued in Section 4.7, it is interesting to consider labelings of area features instead of point features, which do not have a unique horizontal order. In this paper we focused on point feature labeling and more research on other feature types is necessary. We have also discussed the two-sided panorama labeling problem and showed NP-hardness of the problem, but our proof (which is a simple reduction of a proof by Garrido et al. [2001]) requires that the labels have non-uniform width. The computational complexity of two-sided panorama labeling problem with uniform width-labels is left open for future research. Finally, the vertical position of labels may have an influence on which features are observed first. In many cases higher features (mountain peaks, tall buildings) are more important and thus should have their labels in the top rows, while smaller features should use the lower rows. Minimizing leader length (Section 4.6) implicitly incorporates this idea, but it is interesting to find faster algorithms that optimize vertical positions explicitly, for example, taking weights of features into account as priorities to place their labels in higher rows.

REFERENCES

Esther M. Arkin and Ellen B. Silverberg. 1987. Scheduling jobs with fixed start and end times. *Discrete Applied Mathematics* 18, 1 (1987), 1–8. DOI: http://dx.doi.org/10.1016/0166-218X(87)90037-0

Michael Bekos, Sabine Cornelsen, Martin Fink, Seok-Hee Hong, Michael Kaufmann, Martin Nöllenburg, Ignaz Rutter, and Antonios Symvonis. 2014. Many-to-one Boundary Labeling with Backbones. In *Proc. 21st Internat. Sympo. Graph Drawing (GD'13) (Lecture Notes in Computer Science)*, Vol. 8242. Springer, 244–255. DOI: http://dx.doi.org/10.1007/978-3-319-03841-4_22

Michael A. Bekos, Michael Kaufmann, Martin Nöllenburg, and Antonios Symvonis. 2010. Boundary Labeling with Octilinear Leaders. *Algorithmica* 57, 3 (2010), 436–461. DOI: http://dx.doi.org/10.1007/s00453-009-9283-6

Michael A. Bekos, Michael Kaufmann, Katerina Potika, and Antonios Symvonis. 2006. Multi-Stack Boundary Labeling Problems. In *Proc. 26th Conf. Foundations of Software Technology and Theoretical Computer Science (FSTTCS'06) (Lecture Notes in Computer Science)*, S. Arun-Kumar and N. Garg (Eds.), Vol. 4337. Springer, 81–92. DOI: http://dx.doi.org/10.1007/11944836_10

Michael A. Bekos, Michael Kaufmann, Katerina Potika, and Antonios Symvonis. 2010. Area-Feature Boundary Labeling. *Comp. J.* 53, 6 (2010), 827–841. DOI: http://dx.doi.org/10.1093/comjnl/bxp087

Michael A Bekos, Michael Kaufmann, and Antonios Symvonis. 2008. Efficient Labeling of Collinear Sites. *J. Graph Algorithms Appl.* 12, 3 (2008), 357–380.

Michael A. Bekos, Michael Kaufmann, Antonios Symvonis, and Alexander Wolff. 2007. Boundary Labeling: Models and Efficient Algorithms for Rectangular Maps. *Comput. Geom. Theory Appl.* 36, 3 (2007), 215–236. DOI: http://dx.doi.org/10.1016/j.comgeo.2006.05.003

Marc Benkert, Herman Haverkort, Moritz Kroll, and Martin Nöllenburg. 2009. Algorithms for Multi-Criteria Boundary Labeling. *J. Graph Algorithms Appl.* 13, 3 (2009), 289–317. DOI: http://dx.doi.org/10.1007/978-3-540-77537-9_25

Claus Brenner, Volker Paelke, Jan-Henrik Haunert, and Nora Ripperda. 2006. The GeoScope – a Mixed-reality System for Planning and Public Participation. In *Proc. 25th Urban Data Management Symposium (UDMS'06)*.

Der-San Chen, Robert G Batson, and Yu Dang. 2011. *Applied integer programming: modeling and solution*. Wiley.

Martin Fink, Jan-Henrik Haunert, André Schulz, Joachim Spoerhase, and Alexander Wolff. 2012. Algorithms for labeling focus regions. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (2012), 2583–2592. DOI:http://dx.doi.org/10.1109/TVCG.2012.193

Michael Formann and Frank Wagner. 1991. A Packing Problem with Applications to Lettering of Maps. In *Proc. 7th Annual ACM Sympos. on Comput. Geom. (SoCG'91)*. 281–288. DOI:http://dx.doi.org/10.1145/109648.109680

M. R. Garey and D. S. Johnson. 1990. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co.

M. Garrido, C. Iturriaga, A. Márquez, J. R. Portillo, P. Reyes, and A. Wolff. 2001. Labeling Subway Lines. In *Proc. 12th Internat. Symp. Algorithms and Computation (ISAAC'01) (Lecture Notes in Computer Science)*, Vol. 2223. Springer, 649–659. DOI:http://dx.doi.org/10.1007/3-540-45678-3_55

Andreas Gemsa, Jan-Henrik Haunert, and Martin Nöllenburg. 2011. Boundary-labeling algorithms for panorama images. In *Proc. 19th ACM SIGSPATIAL Internat. Conf. Advances Geograph. Inform. Syst. (ACM GIS'11)*. ACM, 289–298. DOI:http://dx.doi.org/10.1145/2093973.2094012

Hsu-Chen Yen Hao-Jen Kao, Chun-Cheng Lin. 2007. Many-to-one Boundary Labeling. In *Proc. Asia-Pacific Sympos. on Visualisation (APVIS'07)*. IEEE, 65–72. DOI:http://dx.doi.org/10.7155/jgaa.00169

Knut Hartmann, Timo Götzelmann, Kamran Ali, and Thomas Strothotte. 2005. Metrics for Functional and Aesthetic Label Layouts. In *Proc. 5th Internat. Conf. on Smart Graphics (SG'05)*. Lecture Notes in Computer Science, Vol. 3638. Springer, 115–126. DOI:http://dx.doi.org/10.1007/11536482_10

Zhi-Dong Huang, Sheung-Hung Poon, and Chun-Cheng Lin. 2014. Boundary Labeling with Flexible Label Positions. In *Proc. 8th Internat. Workshop on Algorithms and Computation (WALCOM'14)*. Lecture Notes in Computer Science, Vol. 8344. Springer, 44–55. DOI:http://dx.doi.org/10.1007/978-3-319-04657-0_7

Michael Kaufmann. 2009. On Map Labeling with Leaders. In *Festschrift Mehlhorn*. Lecture Notes in Computer Science, Vol. 5760. Springer, 290–304. DOI:http://dx.doi.org/10.1007/978-3-642-03456-5_20

Philipp Kindermann, Benjamin Niedermann, Ignaz Rutter, Marcus Schaefer, André Schulz, and Alexander Wolff. 2013. Two-Sided Boundary Labeling with Adjacent Sides. In *Proc. 13th Internat. Workshop on Algorithms and Data Structures (WADS'13)*. Lecture Notes in Computer Science, Vol. 8037. Springer, 463–474. DOI:http://dx.doi.org/10.1007/978-3-642-40104-6_40

Chun-Cheng Lin. 2010. Crossing-Free Many-to-One Boundary Labeling with Hyperleaders. In *Proc. IEEE Pacific Visualisation Symposium (PacificVis'10)*. 185–192. DOI:http://dx.doi.org/10.1109/PACIFICVIS.2010.5429592

M. Nöllenburg, V. Polishchuk, and M. Sysikaski. 2010. Dynamic one-sided boundary labeling. In *Proc. 18th ACM SIGSPATIAL Internat. Conf. Advances Geograph. Inform. Syst. (ACM GIS 2010)*. 310–319. DOI:http://dx.doi.org/10.1145/1869790.1869834

Sheung-Hung Poon, Chan-Su Shin, Tycho Strijk, Takeaki Uno, and Alexander Wolff. 2003. Labeling Points with Weights. *Algorithmica* 38, 2 (2003), 341–362. DOI:http://dx.doi.org/10.1007/s00453-003-1063-0

Marc van Kreveld, Tycho Strijk, and Alexander Wolff. 1999. Point Labeling with Sliding Labels. *Comput. Geom. Theory Appl.* 13 (1999), 21–47. DOI:http://dx.doi.org/10.1016/S0925-7721(99)00005-X

Alexander Wolff and Tycho Strijk. 1996. The Map-Labeling Bibliography. (1996). http://i11www.iti.kit.edu/map-labeling/bibliography