

On Multi-Stack Boundary Labeling Problems*

MICHAEL A. BEKOS¹, MICHAEL KAUFMANN², KATERINA POTIKA¹, ANTONIOS SYMVONIS¹

¹National Technical University of Athens
School of Applied Mathematical & Physical Sciences
15780 Zografou, Athens
GREECE
{mikebekos, epotik, symvonis}@math.ntua.gr

²University of Tübingen
Institute for Informatics
Sand 13,72076 Tübingen
GERMANY
mk@informatik.uni-tuebingen.de

Abstract: Boundary labeling is a relatively new labeling method. It targets the areas of technical drawings and medical maps, where it is often common to explain certain parts of the drawing with large text labels arranged on its boundary, so that other parts of the drawing are not obscured.

According to this labeling method, we are given a rectangle R , which encloses a set of n sites. Each site s_i is associated with an axis-parallel rectangular label l_i . The labels must be placed in distinct positions on the boundary of R and to be connected to their corresponding sites with polygonal lines, called leaders, so that a) labels are pairwise disjoint and b) leaders do not intersect each other.

In this paper, we examine labelings with more than one stacks of uniform labels on each side of R and we aim to maximize the (uniform) label size.

Key-Words: Boundary Labeling, Sites, Labels, Leaders, Multi-Stacks, Dynamic Programming.

1 Introduction

In medical maps and technical drawings, it is often common to explain certain features of the drawing with text labels, that are arranged on its boundary. This approach is reasonable, since in most cases such labels contain long texts and if they were placed next to their features, they would obscure other features of the drawing (see Figure 1). Bekos, Kaufmann, Symvonis and Wolff [4] proposed *boundary labeling* to model this problem.

A boundary labeling in its primitive form can be described as follows: We are given a set P of n sites $s_i, i = 1, 2, \dots, n$, each associated with an open, rectangular label l_i of width w_i and height h_i . The site set P and the corresponding drawing are enclosed in an axis-parallel rectangle R of sufficient size, which is called *enclosing rectangle*. The labels should be placed on distinct positions on the boundary of R so that they do not overlap, and should be connected to their corresponding sites with non-intersecting poly-

gonal lines, called *leaders*. Such labelings are referred to as *legal* or *crossing free boundary labelings*¹.

Since a boundary labeling consists of several parameters (sites, labels, leaders, enclosing rectangle), there exist several variations of the primitive form discussed above, each giving rise to different labeling models. In the simplest form of the problem, the sites model point locations on a map (e.g. a city center, or the capital of a prefecture). In this case, each site s_i is associated with a point $p_i = (x_i, y_i)$ of the plane (see Figures 1, 3, 2). To avoid leader overlaps, which would result in unreadable drawings, we make an additional assumption regarding the site locations: We assume that the sites are placed in general position i.e. no three sites are collinear and no two sites share the same x or y coordinate. In real applications, several times we want to associate a label with a bounded area of a map (e.g. a lake or a mountain range). In such cases, we associate these regions either with a line segment or a rectangular area or a convex polygon, in general. Any point inside (or on the boundary of)

*This work has been supported by the EPEAEK program Pythagoras 89181(28).

¹For the sake of simplicity will be referred to as boundary labelings in the rest of the paper.



Figure 1: The anatomy of an eye.

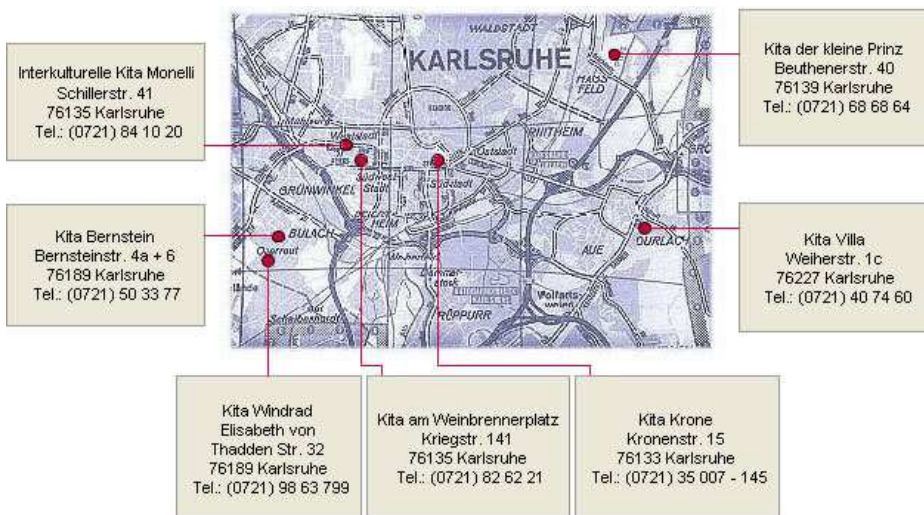


Figure 2: A map of the kindergartens of Karlsruhe.

the proposed shape can be used to represent the corresponding region. Figure 4, displays a regional map of Italy, where each region is associated with a rectangular area and a point on the boundary of each rectangle is chosen to represent the corresponding region.

As already mentioned, each site is connected with its corresponding label with non-intersecting polygonal lines, which are called leaders. We focus on three different types of leaders:

Type-*s* leaders: Leaders of type *s* consist of a single straight line segment, originating from the site

and leading to its label (see Figure 3).

Type-*po* leaders: Leaders of type *po* consist of two line segments. The first one is parallel (*p*) to the side of *R* containing the label it leads to, whereas the second one is orthogonal (*o*) to that side (see Figure 1). Degenerated case of a *po*-leader is a leader of type *o*, which consists of only one line segment orthogonal to side of *R* containing the label it leads to.

Type-*opo* leaders: Following the same notation scheme as for leaders of type-*po*, leaders of type



Figure 3: A technical drawing of a GPS device.

opo consist of three line segments (see Figures 2, 4). For each type *opo* leader, we further assume that it has its parallel *p*-segment outside the enclosing rectangle *R*, routed in the so-called *track routing area*. Again, leaders of type *o* are trivially considered to be of type *opo*, as well.

In general, the *type* of a leader is defined as an alternating string over the alphabet $\{p, o\}$. The reason why we focus only on type-*po* and type-*opo* leaders is because we target on simple and easy to visualize labelings.

The labels are attached on the boundary of an axis-parallel rectangle $R = [l_R, r_R] \times [b_R, t_R]$ of height $H = t_R - b_R$ and width $W = r_R - l_R$, which contains all sites $p_i \in P$. By allowing the labels to be placed on different sides of *R*, one can define several different labeling models. In the general case of the problem, the labels are allowed to be placed on all four sides of *R*. However, of particular interest is the case of two opposite sides (see Figures 1, 3 and 4), since it leads to quite legible drawings.

In general, the labels are of arbitrary size (*non-uniform labels*), i.e. label l_i which corresponds to site s_i has height h_i and width w_i . Assume that labels must be placed either to the west or east side of the enclosing rectangle *R*, and that the label heights sum up to twice the height *H* of *R*. It is clear that the task of assigning the labels to the two sides corresponds to the well known PARTITION problem, which is weakly NP-complete [6]. So, it is reasonable to separately consider the restricted cases where the labels are of

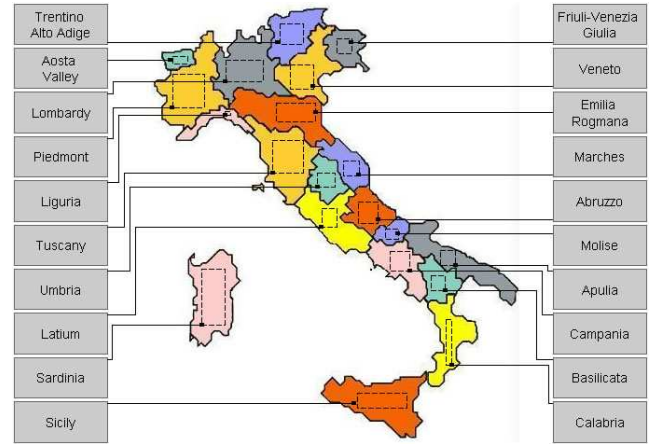


Figure 4: A regional map of Italy.

uniform size, or of *maximum uniform size*. Figures 1, 2, 3 and 4 all display labelings with labels of uniform size.

Another important parameter of a boundary labeling model is the *label port*, i.e. the point where a leader may touch its corresponding label. Ports may be fixed (e.g., the middle of a label edge) or may slide along a label edge. Figures 1 and 3 display labelings with fixed ports, whereas Figures 2 and 4 with sliding ports.

In this paper we consider a relatively new boundary labeling model, which supports multiple stacks of labels on each side of the enclosing rectangle *R*. In the case of multiple stacks of labels (say *m* stacks), a leader of type-*opo* can have its *p* segment either between the enclosing rectangle *R* and the first stack (called *first track routing area*) or between the *i*-th and the (*i* + 1)-th stack, where $i < m$ (called *(i + 1)-th track routing area*). Figures 8 and 9 are produced from the algorithms of Section 3.1 and depict two regional maps of UK and Italy, respectively. The labels occupy two consecutive stacks to the east side of the enclosing rectangle. The *opo*-leaders connected to labels on the second stack are restricted to bend only in the second track routing area. It is easily perceived that this additional requirement leads to labelings with smaller labels, however the drawings appear to be quite readable.

Under a boundary labeling model, one can define several other optimization problems, adopting one of the following optimization criteria:

Minimize the total number of bends (TNBM):

Find a legal boundary labeling, such that the total number of bends is minimum.

Minimize the total leader length (TLLM): Find a legal boundary labeling, such that the total leader length is minimum.

Minimize the maximum leader length (MLLM):

Find a legal boundary labeling, such that the length of the longest leader is minimum.

Maximize label size (LSM): Find a maximum scaling factor s and a corresponding legal boundary labeling, such that each feature is labeled with a label scaled by s .

This paper is structured as follows: Section 2, reviews previous results on boundary labeling. In Section 3, we focus on the multi stack boundary labeling model and present algorithms to produce labelings of maximum label height. We conclude in Section 4 with open problems and future work.

2 Previous Work

Bekos et al. introduced the boundary labeling problem in [4] (see also [3]). They examined a variety of models based on the type of leaders, the type of label ports, the location and the size of the labels. They presented several algorithms for legal leader label placements and leader bend and leader length minimization. An algorithm for minimizing the total leader length on four sides with type-*opo* leaders is presented for simple points in [1] and for polygons in [2]. BLer [5] is an integrated software package, which automates the boundary labeling process and supports most of the known algorithms.

Table 1 summarizes known results on boundary labeling.

3 Label Size Maximization

3.1 Two stacks of labels on the same side

In this section, we consider boundary labelings with *opo* leaders, where the labels are placed at two consecutive stacks on the same side of the enclosing rectangle R . Without loss of generality, we assume that

the labels are located to the east side of R . Leaders connected to labels that are on the second stack are restricted to bend (or equivalently have their p segments) in the second track routing area. The assumed model is quite general, since it permits sliding labels with sliding ports. We further assume that all labels have the same size (and therefore the same height h) and we seek to maximize their heights.

Lemma 1 *Given a rectangle R of height H and a set $P \subset R$ of n points (sites), each associated with a label of height h , there is an $O(n^2)$ time algorithm that determines whether there exists a legal type-*opo* boundary labeling, so that the leaders connected to labels on the second stack are restricted to bend on the second track routing area.*

Proof:

Observe that, in any legal one-side labeling with type-*opo* leaders, the vertical order of the sites is identical to the vertical order of their corresponding labels on both stacks. This, together with the assumption that no two sites share the same y -coordinate, guarantees that leaders do not intersect.

So, we assume that the sites are sorted according to increasing y -coordinate and we propose a dynamic programming algorithm, that maintains a $(n + 1) \times (n + 1)$ table T . Each entry $T[i, k]$, $i \geq k$, of table T is the highest occupied Y -coordinate of the first stack, when the labels of the first i sites have been placed and k out of them have their labels on the second stack. Entry $T[i, k]$ is ∞ , when it is impossible to place the first i labels with k labels on the second stack. Therefore, all table entries $T[i, k]$, with $i < k$ are ∞ .

As usual, the table entries are computed in a bottom-up fashion. Assuming that we have placed the labels for the first $i - 1$ sites, we try to place the i -th label, which is connected to the i -th site. We distinguish two cases based on whether the label is placed on the first or second stack. From the two alternatives, we select the one, which minimizes the occupied area by labels of the first stack. Thus, for computing $T[i, k]$ we only need to know the entries $T[i - 1, k - 1]$ and $T[i - 1, k]$.

We denote by $T_1[i, k]$, $i \geq k$ to be the highest occupied Y -coordinate of the first stack, when the i -th site has its label on the first stack, the labels of the first i sites have been placed and k out of them have

Sides	Type of Sites	Leaders	Ports	Type of Labels	Objective	Complexity	Reference
4	Simple	s	Fixed	Uniform, max size	legal	$O(n \log n)$	[4]
4	Simple	s	Fixed	Uniform, max size	TLLM	$O(n^{2+\epsilon})$	[1]
4	Simple	s	Sliding	Uniform, max size	TLLM	$O(n^3)$	[1]
2	Simple	po	Sliding	Uniform, max size	TLLM	$O(n^2)$	[4]
1	Simple	opo	Sliding	Non-uniform, sliding	legal	$O(n \log n)$	[4]
1	Simple	opo	Fixed	Uniform, max size	TLLM	$O(n \log n)$	[4]
2	Simple	opo	Sliding	Non-uniform, sliding	TLLM	$O(n^2 H)$	[4]
1	Simple	opo	Sliding	Non-uniform, sliding	TNBM	$O(n^2)$	[4]
4	Simple	opo	Sliding	Non-uniform, sliding	legal	$O(n \log n)$	[4]
4	Simple	opo	Fixed	Uniform, max size	TLLM	$O(n^2 \log^3 n)$	[1]
4	Simple	opo	Sliding	Uniform, max size	TLLM	$O(n^3)$	[1]
4	Polygonal	opo	Fixed	Uniform, max size	TLLM	$O(n^2 \log^3 n)$	[2]
4	Polygonal	opo	Sliding	Uniform, max size	TLLM	$O(n^3)$	[2]

Table 1: Known results on boundary labeling.

their labels on the second stack. $T_1[i, k]$ can be computed based on entry $T[i-1, k]$ (see Figure 5). To simplify this computation, we assume that $\infty + a = \infty, \forall a \in \mathbb{R} \cup \{\infty\}, \min\{\infty, \infty\} = \infty$ and we define the operator $\oplus : \mathbb{R} \cup \{\infty\} \times \mathbb{R} \cup \{\infty\} \rightarrow \mathbb{R} \cup \{\infty\}$ where:

$$a \oplus b = \begin{cases} a + b, & \text{if } a + b \leq H \\ \infty, & \text{otherwise} \end{cases}$$

Then, we can easily observe that:

$$T_1[i, k] = T[i-1, k] \oplus h$$

Similarly, $T_2[i, k], i \geq k$ is defined to be the highest occupied Y -coordinate of the first stack, when the i -th site has its label on the second stack, the labels of the first i sites have been placed and k out of them have their labels on the second stack. $T_2[i, k]$ can be computed based on entry $T[i-1, k-1]$. Note that in order to connect site s_i with a label placed on the second stack, it must hold that $y_i > T[i-1, k-1]$ (see Figure 6). This is because we assumed that all leaders connected to labels on the second stack are restricted to bend in the second track routing area. If this condition is not fulfilled (see Figure 7), then site s_i can not be connected with a label on the second stack. Thus, we set $T_2[i, k] = \infty$. We conclude that $T_2[i, k]$ can be computed by using the following formula:

$$T_2[i, k] = \begin{cases} y_i, & \text{if } kh \leq H \text{ and } y_i > T[i-1, k-1] \\ \infty, & \text{otherwise} \end{cases}$$

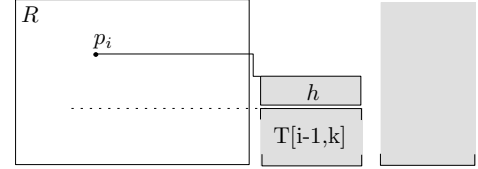


Figure 5: Placing l_i in the first stack.

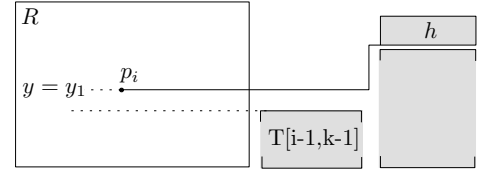


Figure 6: Placing l_i in the second stack.

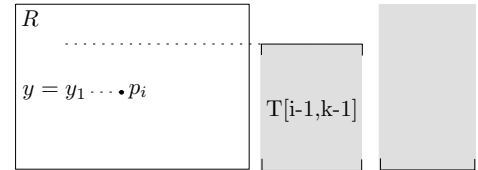


Figure 7: Placing l_i in the second stack is not possible.

Having computed $T_1[i, k]$ and $T_2[i, k]$, entry $T[i, k]$ is computed as follows:

$$T[i, k] = \min\{T_1[i, k], T_2[i, k]\}, \quad (1)$$

Algorithm 1: 1SIDEOP02STACKS

input : A set of n points $p_i = (x_i, y_i)$ in the plane, the height H of the enclosing rectangle $R = [l_R, r_R] \times [b_R, t_R]$ and a real number h .

output : A boolean value, which indicates, whether there exists a legal solution, when the height of each label is h .

require : $y_1 < y_2 < \dots < y_n$.

- 1 {Fill dynamic programming table T }
 $T[0, 0] = b_R$
for $i = 1$ **to** n **do**
 $T[i, 0] = T[i - 1, 0] \oplus h$
 $T[i - 1, i] = \infty$
 for $k = 1$ **to** i **do**
 $T_1[i, k] = T[i - 1, k] \oplus h$
 if $(y_i > T[i - 1, k - 1])$ **and** $kh \leq H$ **then**
 $T_2[i, k] = y_i$
 else
 $T_2[i, k] = \infty$
 $T[i, k] = \min\{T_1[i, k], T_2[i, k]\}$
 - 2 {Determine whether there exists legal placement}
 for $j = 1$ **to** n **do**
 if $T[n, j] < \infty$ **then**
 return true
 return false
-

For a fixed label height h , Algorithm 1 outputs a boolean value, which indicates whether there exists a legal label placement. The algorithm is directly based on the recurrence relation computed above (see block 1 of the algorithm). Block 2 of Algorithm 1 determines whether there exists a legal label placement by identifying an index j with $0 \leq j \leq n$ such that $T[n, j] < \infty$.

It is obvious that Algorithm 1 runs in $O(n^2)$ time and uses $O(n^2)$ space. By using an extra table of the same size as T , Algorithm 1 can easily be modified, such that it also computes the label and leader positions. □

Theorem 1 *Given a rectangle R of integer height H and a set $P \subset R$ of n points (sites) in general position, there is an $O(n^2 \log H)$ time algorithm that de-*

termines a legal type-opo boundary labeling, so that the uniform size of each label is maximum and all leaders connected to labels on the second stack are restricted to bend on the second track routing area.

Proof: In order to solve the *size maximization problem* (i.e. to determine the optimal solution subject to the height of each label), we simply apply a binary search on all possible discrete values for height h . To complete the proof, observe that $\frac{H}{n} \leq h \leq \frac{2H}{n}$. By calling Algorithm 1 $O(\log H)$ times, we can determine the maximum label height h . □

Figures 8 and 9 are produced from Algorithms 1. The labels occupy two stacks to the east side of the enclosing rectangle and their heights are maximized. Note that all leaders connected to labels at the second stack bend at the second track routing area.

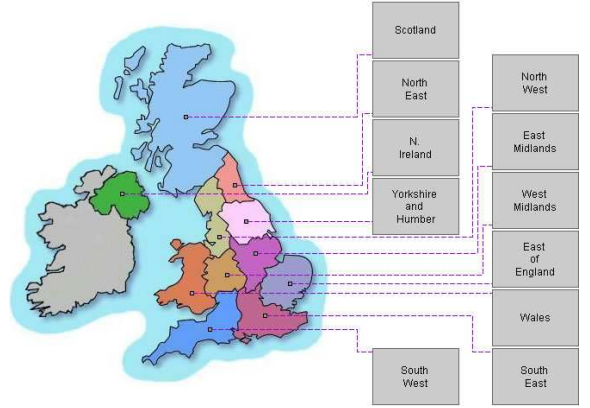


Figure 8: A regional map of UK.

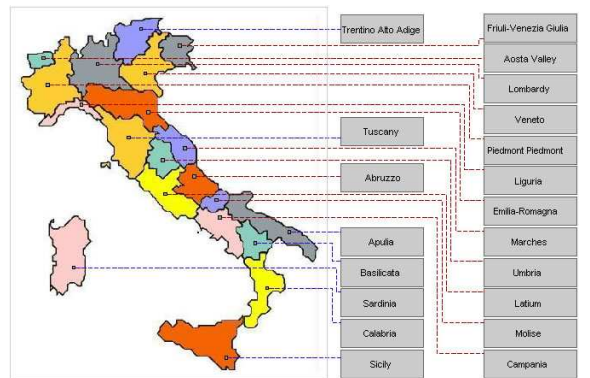


Figure 9: A regional map of Italy.

3.2 Two opposite sides labeling with 2 stacks on one side

In this section, we generalize the algorithm of Section 3.1 to support an additional stack of labels to the west side of the enclosing rectangle R (see Figure 10). It is obvious that this additional stack leads to labelings with greater labels. Again, we assume that the labels are of uniform size (and therefore of same height h) and we seek to maximize their heights. Note that the corresponding problem with non-uniform labels is NP -complete, since it can be easily reduced to the well known PARTITION problem, which is weakly NP -complete [6].

To make the description of our algorithm simple, we number the stacks as follows (see also Figure 10):

- E_1 is the first stack on the east side of R .
- E_2 is the second stack on the east side of R .
- W_1 is the stack on the west side of R .



Figure 10: Numbering of stacks.

Lemma 2 *Given a rectangle R of height H and a set $P \subset R$ of n points (sites), each associated with a label of height h , there is an $O(n^3)$ time algorithm that determines whether there exists a legal type-opo boundary labeling, when three stack of labels are proposed, one to the west and two to the east side of R .*

Proof: We propose a dynamic programming algorithm, that maintains a $(n+1) \times (n+1) \times (n+1)$ table T . Each entry $T[i, k, l]$, $i \geq k+l$, of table T is the highest occupied Y -coordinate of stack E_1 , when the labels of the first i sites have been placed and k (l) out of them have their labels on E_2 (W_1 , respectively). Entry $T[i, k, l]$ is ∞ , when it is impossible to place the first i labels with k labels on E_2 and l labels on W_1 . Therefore, all table entries $T[i, k, l]$, with $i < k+l$ are ∞ .

Assuming that we have placed the labels for the first $i-1$ sites, we try to place the i -th label, which corresponds to the i -th site. We distinguish now three cases based on whether the label is placed on E_1 , E_2 or W_1 . From the three alternatives, we select the one, which minimizes the occupied area by labels of E_1 stack. Thus, for computing $T[i, k, l]$ we only need to know the entries $T[i-1, l, k]$, $T[i-1, k-1, l]$ and $T[i-1, k, l-1]$.

We denote by $T_{W_1}[i, k, l]$, $i \geq k$ to be the highest occupied Y -coordinate of the first stack, when the i -th site has its label on W_1 , the labels of the first i sites have been placed and k (l) out of them have their labels on E_2 (W_1 , respectively). Similarly, $T_{E_1}[i, k, l]$ and $T_{E_2}[i, k, l]$, $i \geq k$ are defined. Then, following similar arguments as in proof of Lemma 1, we can show that:

$$T[i, k, l] = \min\{T_{W_1}[i, k, l], T_{E_1}[i, k, l], T_{E_2}[i, k, l]\}, \quad (2)$$

where:

$$T_{W_1}[i, k, l] = T[i-1, k, l-1] \oplus h$$

$$T_{E_1}[i, k, l] = T[i-1, k, l] \oplus h$$

$$T_{E_2}[i, k, l] = \begin{cases} y_i, & \text{if } kh \leq H \text{ and } y_i > T[i-1, k-1, l] \\ \infty, & \text{otherwise} \end{cases}$$

Algorithm 2 provides a more detailed description of our algorithm. It is directly based on Equation 2 (see block 1 of algorithm 2). To determine whether there exists a legal label placement, we have to identify a pair of indexes (i, j) with $0 \leq i+j \leq n$ such that $T[n, i, j] < \infty$ (see block 2 of Algorithm 2).

Algorithm 2 needs $O(n^3)$ time and space, since it maintains a $(n+1) \times (n+1) \times (n+1)$ table T and each entry is computed in constant time. By using an extra table of the same size as T , Algorithm 2 can easily be modified, such that it also computes the label and leader positions. \square

To solve the size maximization problem (i.e. to determine the optimal solution subject to the height of each label), we follow same arguments as in Theorem 1. Theorem 2 summarizes our results:

Theorem 2 *Given a rectangle R of integer height H*

Algorithm 2: 2SIDESOPO3STACKS

input : A set of n points $p_i = (x_i, y_i)$ in the plane, the height H of the enclosing rectangle $R = [l_R, r_R] \times [b_R, t_R]$ and a real number h .

output : A boolean value, which indicates, whether there exists a legal solution, when the height of each label is h .

require : $y_1 < y_2 < \dots < y_n$.

1 {Fill dynamic programming table T }

$T[0, 0, 0] = b_R$

for $i = 1$ **to** $n - 1$ **do**

for $k = 0$ **to** i **do**

for $l = 0$ **to** $i - k$ **do**

$T_{W_1}[i, k, l] = T[i - 1, k, l - 1] \oplus h$

$T_{E_1}[i, k, l] = T[i - 1, k, l] \oplus h$

if ($y_i > T[i - 1, k - 1, l]$ **and** $kh \leq H$) **then**

$T_{E_2}[i, k, l] = y_i$

else

$T_{E_2}[i, k, l] = \infty$

$T[i, k, l] =$

$\min\{T_{W_1}[i, k, l], T_{E_1}[i, k, l], T_{E_2}[i, k, l]\}$

2 {Determine whether there exists legal placement}

for $i = 0$ **to** n **do**

for $j = 0$ **to** $n - i$ **do**

if $T[n, i, j] < \infty$ **then**

return true

return false

and a set $P \subset R$ of n points (sites) in general position, there is an $O(n^3 \log H)$ time algorithm that determines a legal type-*opo* boundary labeling with labels placed at three stacks (one to the west and two to the east side of R), so that the uniform size of each label is maximum.

4 Open Problems and Future Work

1. For multi-stack labeling problems we presented results only for the label size maximization problem. No results are known regarding the total leader length minimization and the minimization of the total number of bends.

2. It is intuitive that the quality of boundary labeling can be improved by allowing “floating” sites inside area features of drawings. Very few algorithms exist for this model. No algorithms exist for minimizing the total number of bends.
3. All the known type-*opo* boundary labelings use a track routing area outside the enclosing rectangle to route the p segment of the leader. Routing the p segment inside the enclosing rectangle might lead to more visually appealing drawings.

References:

- [1] M. A. Bekos, M. Kaufmann, K. Potika, and A. Symvonis. Boundary labelling of optimal total leader length. In *Proc. 10th Panhellenic Conference On Informatics (PCI2005)*, LNCS 3746, pages 80–89, 2005.
- [2] M. A. Bekos, M. Kaufmann, K. Potika, and A. Symvonis. Polygon labelling of minimum leader length. In *Proc. Asia Pacific Symposium on Information Visualisation (APVIS2006)*, CR-PIT 60, pages 15–21, 2006.
- [3] M. A. Bekos, M. Kaufmann, A. Symvonis, and A. Wolff. Boundary labeling: Models and efficient algorithms for rectangular maps. *Computational Geometry: Theory and Applications*. To appear.
- [4] M. A. Bekos, M. Kaufmann, A. Symvonis, and A. Wolff. Boundary labeling: Models and efficient algorithms for rectangular maps. In *Graph Drawing*, pages 49–59, 2004.
- [5] M. A. Bekos and A. Symvonis. Bler: A boundary labeller for technical drawings. In *Proc. 13th Int. Symposium on Graph Drawing (GD’05)*, LNCS 3843, pages 503–504, 2005.
- [6] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, 1979.