Boundary Labeling in Text Annotation

Chun-Cheng Lin[1,*] Hsiang-Yun Wu[2,†] Hsu-Chun Yen[2,3,‡]

[1] Dept. of Computer Science and Information Engineering, National Kaohsiung University of Applied Sciences, Kaohsiung, Taiwan 807, ROC
[2] Dept. of Electrical Engineering, National Taiwan University, Taipei, Taiwan 106, ROC
[3] Dept. of Computer Science, Kainan University, Taoyuan, Taiwan 338, ROC

Abstract

The text annotation system of a word processor software provides the user the function of memorandums in editing a document. In the visualization interface of the annotation system, each marked word is connected to a text comment label on the right side of the document by a polygonal line. Such a visualization interface can be viewed as a one-side *boundary labeling*, in which each point site is uniquely connected to a label placed on the right side of an enclosing rectangle by a *leader*, which may be a rectilinear or straight line segment. In the literature, there have existed some applications and some theoretical results for the boundary labeling. In this paper, we investigate the boundary labeling from the application on the annotation system. For this kind of labeling, if the number of labels on the right side is large, the leaders may be drawn too densely to be recognized easily. Therefore, in this paper, we propose a polynomial time algorithm for the so-called *1.5-side boundary labeling* for the annotation system, in which, in addition to being connected to the right side directly, leaders can be routed to the left side temporarily and then finally to the right side. In addition, we investigate a problem for two-side boundary labeling (for the annotation system) that was not discussed previously. We show the problem to be NP-complete, and then proposed a heuristic based on the genetic algorithm to solve it. The experimental results reveal that our approach performs well.

1 Introduction

In information visualization, cartography, geographic information systems (GIS), and graph drawing, *map labeling* is an important task which is concerned with efficiently placing extra information, in the form of text labels, next to features (such as point features, line features, or area features) in a drawing (map). In order to ensure readability, unambiguity and legibility, it is suggested that the labels be pairwise disjoint and close to the features to which they belong [9]. A detailed bibliography and survey on map labeling can be found in [15], [12], respectively. ACM Computational Geometry Impact Task Force [6] has identified label placement as an important area of research. The majority of map labeling problems are known to be NP-complete [7, 10] in general. (The interested reader is also referred to [13, 14] for various approximations and heuristics for map labeling.)

Most of the research on map labeling has primarily focused on labeling point features, and the basic requirement in this case is that all the labels should be pairwise disjoint. It is clear that such a requirement is difficult to be achieved in the case where large labels are placed on dense points. In practice, large labels are usually used in technical drawings or medical atlases where certain site-features are explained with blocks of texts. To address this problem, Bekos et al. proposed the so-called *boundary labeling* [1, 3, 4], in which all labels are attached to the boundary (four sides) of a rectangle R enclosing all sites, and each site is connected to a unique label by a *leader*, which may be a rectilinear or straight line segment. In such a setting, they assumed that there are no two sites with the same x- or y- coordinates, and investigated how to place the labels and leaders in a drawing such that there are no crossings among leaders and either the total leader length or the bends of leaders are minimized under a variety of constraints. In a recent article, Bekos et al. [2] investigated a similar problem for labeling polygonal sites under the framework of boundary labeling. Furthermore, Lin et al. [11] investigated the *multi-site-to-one-label boundary labeling*, in which more than one site is allowed to be connected to a common label and each site is connected only by a leader.

Conventionally, boundary labeling [1, 3, 4, 5] is identified as *k-side labeling with type-t leaders* (where $k \in \{1,2,4\}$ and $t \in \{opo, po, s, do\}$) if the labels are allowed to be attached to the k sides of the enclosing rectangle R by only type-t leaders. The parameter t specifies the way a leader is drawn to connect a site to a label. The *opo*, *po*, *s*, and *do* mean that the leader stand for *orthogonal-parallel-orthogonal*, *parallel-orthogonal*, *straight-line* and *diagonal-orthogonal*, respectively, which can easily be understood from the examples given in Figures 1 (a), (b), (c) and (d). For each type-*opo* leader, we further assume that the parallel (i.e., 'p') segment lies immediately outside R in the so-called *track routing area*, as shown in Figure 1 (a).

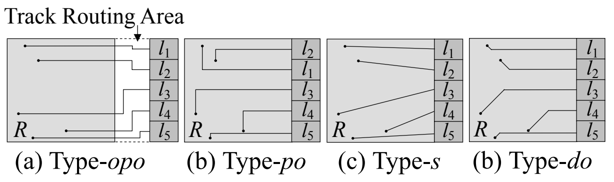


Figure 1: Illustration of leaders.

A *word processor* is a software application for editing and printing documents, e.g., MS Word, OpenOffice.org Writer, KWord, etc. In general, the word processor provides a so-called *text annotation system*, as shown in Figure 2, in which each *marked word* (which can be viewed a rectangular site) in the document is connected to a *text comment* (in the form of text labels) on the right side of the document by a type-*od* leader (i.e., each leader from its site to its label is composed of an orthogonal and then a diagonal segments), and the orthogonal segment of each leader is drawn as an underline of the text line accommodating its corresponding marked word. In such an application, a text line may contain more than one marked word (i.e., y-coordinates of sites may be the same), so that the orthogonal segments connected to those marked words are overlapped. Therefore, the text annotation system can be viewed as a one-line-to-many-label boundary labeling with type-*od* leaders where y-coordinates of sites may be the same.

[*] Corresponding author. E-mail: cclin321@gmail.com
[†] E-mail: snidget@cobra.ee.ntu.edu.tw
[‡] Research Supported in part by NSC Grant 97-2221-E-002-094-MY3, Taiwan. E-mail: yen@cc.ee.ntu.edu.tw

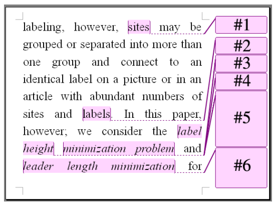

labeling, however, sites may be grouped or separated into more than one group and connect to an identical label on a picture or in an article with abundant numbers of sites and labels. In this paper, however, we consider the *label height minimization problem* and *leader length minimization* for

Figure 2: The annotation system on a word processor.

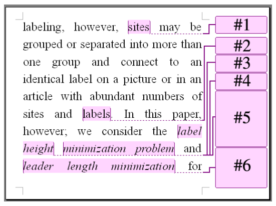

labeling, however, sites may be grouped or separated into more than one group and connect to an identical label on a picture or in an article with abundant numbers of sites and labels. In this paper, however, we consider the *label height minimization problem* and *leader length minimization* for

Figure 3: One-side boundary labeling with type-*opo* leaders.

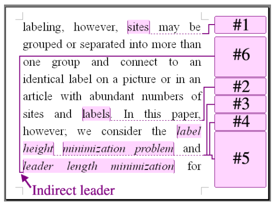

labeling, however, sites may be grouped or separated into more than one group and connect to an identical label on a picture or in an article with abundant numbers of sites and labels. In this paper, however, we consider the *label height minimization problem* and *leader length minimization* for

Figure 4: 1.5-side boundary labeling with type-*opo* leaders.

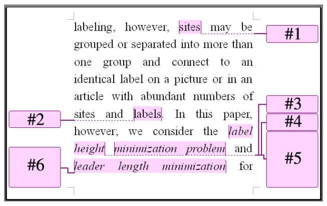

labeling, however, sites may be grouped or separated into more than one group and connect to an identical label on a picture or in an article with abundant numbers of sites and labels. In this paper, however, we consider the *label height minimization problem* and *leader length minimization* for

Figure 5: Two-side boundary labeling with type-*opo* leaders.

The text annotation system has the following merits: for individual use, the system can be viewed as memorandums, which, for example, users can apply to recording current and previous progress; for group use, the system can reduce the communication costs among the group members. Hence, the system is very helpful in practice. However, the current visualization interface for the text annotation system is not admirable and has the following drawbacks:

- When the number of text comment labels is getting larger, or there exist many large-size labels, the labels associated with the same text line may be placed too far from the text line (e.g., see labels #3, #4 and #5 which are associated with the second text line from the bottom in Figure 2). In this case, the diagonal segments of the leaders connected to those labels would be too close to be recognized easily.

- When the number of marked words grows larger, the number of text comment labels also grows larger. In this case, the annotation of placing all the labels on the same side is not admirable in visualization.

As a result, it is of importance and interest to develop better visualization interface for the text annotation system.

In this paper, we develop two approaches to the visualization interface for the text annotation system. One should notice that we still apply one-site-to-one-label boundary labeling to the interface, and the only difference between the previous work and our work is that two sites with the same y-coordinates are allowed in our work. In our setting, since too dense type-*od* leaders may not be recognized easily (see Figure 2), we prefer to apply type-*opo* leaders (see Figure 3). First, we propose the so-called *1.5-side boundary labeling with type-opo leaders*, as shown in Figure 4, in which type-*opo* leaders can be classified into two categories: direct leaders (as the original ones) and indirect leaders (i.e., we make use of the left side of document to route type-*opo* leaders so that the leaders are connected to their labels indirectly, e.g., see the leader connected to label #6 in Figure 4). Under this setting, we show that there exists a polynomial time algorithm to place labels and leaders such that the total leader length is minimized and there are no leader crossings.

Second, we propose the two-side boundary labeling with type-*opo* leaders for the annotation system, as shown in Figure 5, in which the labels placed on the two sides of the document are not crowded so that we can observe them easily. Although the two-side boundary labeling with type-*opo* leaders has been discussed in the previous work [1, 3, 4], all of them assume that the number of labels on each of the two sides of the document are given. In this paper, we consider that the number of labels on the two sides are unknown to investigate the problem of how to allocate the labels on the two sides and route only direct leaders such that both the total leader length and the difference between the sums of heights of labels on the two sides are minimized, while there are no leader crossings. When the total leader length is smaller, each label would be placed more near its corresponding site, so that users can easily trace the leader between labels and their corresponding sites. In addition, if the difference of the sums of the heights of labels on the two side is smaller, then we can observe the labels on the two sides in a more balanced way, and not too many labels are placed on the same side. Therefore, the concerned problem makes sense in practice. We show this problem to be NP-complete, and hence propose an approach based on the well-known *genetic algorithm* (GA) [8] to solve it.

The GA is a heuristic to find the solution of the problem by globally searching the feasible solution space, in which each feasible solution can be represented by a binary string or integers. In our problems for two-side boundary labeling, if we determine which side each label is placed on, then there exists only a feasible leader routing such that there are no leader crossings. Hence, we only consider how to determine which side each label is placed on. Each label is assigned a state which is zero (resp. one) if the label is placed on the right (resp., left) side of the document. By doing this, each feasible solution can be modeled as a binary string. Hence, GA is suitable to solve our problems for the two-side labeling. Finally, we develop a prototype for our approaches, and the experimental results look promising.

2 Preliminaries

In this section, we first introduce the boundary labeling model, and then the genetic algorithm.

2.1 Boundary Labeling Model

Based on Bekos et al.'s work [2], we extend their boundary labeling model to be expressed as an 8-tuple ($Side$, $LabelSize$, $LabelPort$, $LabelPos$, $Leader$, $IndirectLeader$, $Site$, $Objective$), where:

Side: Labels can be placed on the East, West, South and North sides of the enclosing rectangle R, which are denoted by N, E, W and S, respectively.

LabelSize: There are two states for LabelSize: UnifSize (all labels are of the same size), MaxSize (all labels are of uniform maximal size) or NonUnifSize (each label l_i is associated with a height h_i and a width w_i).

LabelPort: FixedPorts (points where a leader can touch a label are predefined) or SlidPorts (points can slide along the label's edge).

LabelPos: FixedPos (labels have either to be aligned with a predefined fixed set of points on the boundary of the rectangle) or SlidePos (labels can slide along the rectangle's sides).

Leader: Types of leaders (*opo*, *po*, *s* or *do*).

IndirectLeader: IndirectAllowed (Indirect leaders are allowed) or IndirectNotAllowed (Indirect leaders are not allowed).

Site: Type of the sites. Each site is a 1-point, line, rectangle, a polygon, etc.

Objective: *LEGAL* (just find a legal label placement), *TLLM* (find a legal label placement, such that the total leader length is minimum), *TLLMDSHM* (find a legal label placement, such that both the total leader length and the difference of sums of the heights of labels on each side is minimum), etc.

According to the above model, the theoretical contributions of this paper are listed in Table 1.

Table 1: Time complexity of our concerned problems.

model	time	reference
(E, UnifSize, FixedPort/SlidPorts, FixedPos, *opo*, IndirectAllowed, 1-point, TLLM)	$O(n^5)$	Thm 1
(EW, UnifSize/NonUnifSize, FixedPort/SlidPorts, FixedPos, *opo*, IndirectNonAllowed, 1-point, TLLMDSHM)	NPC	Thm 2

2.2 Genetic Algorithm

The *Genetic algorithm* (GA) [8] is a stochastic global search method that has proved to be successful for many kinds of optimization problems. GA is categorized as a global search heuristic. It works with a population of candidate solutions and tries to optimize the answer by using three basic principles, including *selection*, *crossover* (also called recombination), and *mutation*. For more details on GA, readers are referred to [8].

3 Our Theoretical Results

In this section, we first show that the 1.5-side boundary labeling with the minimum total leader length can be found in polynomial time, and then we show that the problem of finding the two-side boundary labeling so that both the total leader length and the difference between the sums of heights of labels on the two side are minimized to be NP-complete.

3.1 One-and-Half-Side Boundary Labeling

In what follows, we use a dynamic programming algorithm to solve the problem of how to find a 1.5-side boundary labeling with type-*opo* leaders, as shown in Figure 4 (where indirect type-*opo* leaders are allowed to be applied), so that the total leader length is minimized, while there are no crossings among leaders.

Before presenting our result, some notations are given as follows. In the case of using fixed ports (resp., sliding ports), we define $Right(p,i)$ to be the length of a direct type-*opo* leader from site p to the predefined port (resp., the closest point) of the i-th label on the East side. Similarly, in the case of using fixed ports (resp., sliding ports), we define $Left(p,i)$ to be the length of an indirect type-*opo* leader from site p to the predefined port (resp., the closest point) of the i-th label on the East side.

The idea of our dynamic programming algorithm is given as follows. As far as the routing of an indirect type-*opo* leader is concerned, we observe that the indirect leader separates sites into three groups and also labels into three groups, as shown in Figure 6 (see the indirect leader connected to site p_i), so that three subproblems of how to route leaders from each group of sites to its corresponding group of labels are formed. We calculate the total leader lengths of the three subproblems, respectively, and compare them with the total leader length while all sites are connected to labels by direct leaders. The minimum of the four total leader lengths is selected as the solution of the original problem.

Our result in this subsection is given in the following theorem.

Theorem 1 *The labeling of the model (E, UnifSize, FixedPort/SlidPorts, FixedPos, opo, IndirectAllowed, 1-point, TLLM) can be found in $O(n^5)$ time.*

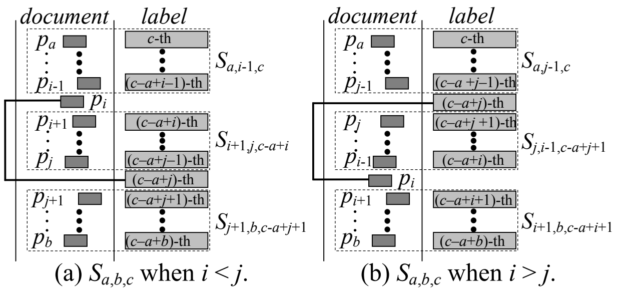


Figure 6: Illustration of a subproblem.

Proof We apply a dynamic programming approach to our concerned problem. Before addressing the approach, we give some notations as follows. There are n sites $p_1, p_2, \cdots, p_n$ in the map, with $y(p_1) \geq y(p_2) \geq \cdots \geq y(p_n)$. Since each label is of uniform size in the concerned problem, to determine the index of a label on the East side is enough to determine the position of the label. Therefore, we let $S_{a,b,c}$ denote the minimal total leader length of the subproblem for the 1.5-side boundary labeling where sites $p_a, p_{a+1}, \cdots, p_b$ are connected to the c-th to the $c+(b-a)$-th labels on the East side. That is, the solution of our concerned problem is equal to $S_{1,n,1}$.

See also Figure 6. Our dynamic programming formula is given as follows:

$$\begin{aligned} S_{a,b,c} \quad = \quad & \min\{\sum_{i=0}^{b-a} Right(p_{a+i}, c+i), \\ & \min_{i,j\in\{a,\cdots,b\},i<j} \{Left(p_i,j) + S_{a,i-1,c} \\ & \qquad + S_{i+1,j,c-a+i} + S_{j+1,b,c-a+j+1}\}, \\ & \min_{i,j\in\{a,\cdots,b\},i>j} \{Left(p_i,j) + S_{a,j-1,c} \\ & \qquad + S_{j,i-1,c-a+j+1} + S_{i+1,b,c-a+i+1}\}\} \end{aligned}$$

where inside the first min function of the above formula, the first term is the total leader length when all the leaders are connected directly to the East side; the second (resp., third) term is the minimum among all the possible 1.5-side boundary labeling when site p_i is indirectly connected to the j-th label on the East side for $i<j$ (resp., $j>i$), which can be realized by Figure 6(a) (resp., Figure 6(b)).

We construct n tables $T_1, \cdots, T_n$ where table T_i has $(n-i+1)\times(n-i+1)$ entries for $i \in \{1,\cdots,n\}$; the entry (j,k) in table T_i is assigned the minimal total leader length when sites $p_j, p_{j+1}, \cdots, p_{j+i-1}$ are connected to the k-th, the $(k+1)$-th, $\cdots$, and the $(k+i-1)$-th labels on the East side by direct and indirect type-*opo* leaders. That is, the solution of our concerned problem is recorded in the entry $(1,1)$ in table T_n. Therefore, after calculating all the entry values in each table, the solution of our concerned problem can be obtained. Note that the space complexity is $\sum_{i=1}^{n}(n-i+1)^2 = O(n^3)$.

For $i = 1,\cdots,n$, we can compute each entry in table T_i in $O(i^2)$ time. The reason is given as follows. W.l.o.g., we compute entry (j,k) in table T_i. According to our dynamic programming formula, entry (j,k) is equal to $S_{j,j+i-1,k}$. Since we consider all pairs between i sites and i labels in the computation of $S_{j,j+i-1,k}$, thus $S_{j,j+i-1,k}$ can obtained in $O(i^2)$ time. Note that when computing each entry in those tables, if the labeling leads to a infeasible solution (e.g., there are crossings among leaders), then the entry is assigned an infinity value.

Since each entry can be computed in $O(i^2)$ time, and there $(n-i+1)^2$ entries in table T_i for $i \in \{1,\cdots,n\}$, the total time complexity is $O(\sum_{i=1}^{n}(n-i+1)^2 i^2) = O(n^5)$. □

Note that it is of interest to find an improved solution for the above problem. Furthermore, if we consider a relaxed version where the label size is set as NonUnifSize, then we do not know how to solve this problem. That is, the problem of finding a legal labeling for the model (E, NonUnifSize, FixedPort/SlidPorts, FixedPos, *opo*, IndirectAllowed, 1-point, TLLM) is open.

3.2 Two-Side Boundary Labeling

In what follows, we consider the problem of how to allocate labels on the two sides and route only direct type-*opo* leaders for the two-side boundary labeling so that both the total leader length and the difference of the sums of the heights of labels on the two side are minimized, while there are no crossings among leaders.

We can show the problem of finding a labeling for the model (EW, UnifSize/NonUnifSize, FixedPort/SlidPorts, FixedPos, *opo*, IndirectNonAllowed, 1-point, TLLMDSHM) to be NP-hard. W.l.o.g., in this paper, we only show the NP-completeness of the following decision problem:

The TLLMDSHM PROBLEM: Given an integer M and an annotation system A of a document in which there are n marked words and n comment labels $l_1, \cdots, l_n$ (where each label l_i has height h_i for $i \in \{1, \cdots, n\}$), is there a two-side boundary labeling (with NonUnifSize and SlidePorts) for A in which the set of labels placed on the East (resp., West) side is denoted L_1 (resp., L_2) such that the total leader length is the minimum and $|\sum_{l_i \in L_1} h_i - \sum_{l_i \in L_2} h_i| < M$?

The TLLMDSHM problem can be shown to be NP-complete as follows, whose proof is omitted due to page limitation.

Theorem 2 *The TLLMDSHM problem is NP-complete.*

Because the TLLMDSHM problem is NP-complete, we apply the genetic algorithm to designing a heuristic for the problem in the next section.

4 GENETIC ALGORITHM FOR THE TWO-SIDE LABELING

The implementation of each steps in designing our genetic algorithm for the two-side labeling are detailed in this section.

4.1 Individual

The individual represents a candidate solution. In our problem, it represents a legal label placement without any leader crossings. An individual is stored as a binary string $s_1 s_2 \cdots s_n$, where for each $i \in \{1, \cdots, n\}$, if label l_i is placed on the West side, $s_i = 0$; otherwise, $s_i = 1$. Such a representation is enough to determine a legal label placement because there is only one boundary labeling without any leader crossings once we know which side each label is placed on [3, 4].

4.2 Initialization

At the beginning of the genetic algorithm, the individuals in the population have to be initialized. Initially many individual solutions are randomly generated to form an initial population. The population size depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions. Conventionally, the population is generated randomly, covering the entire range of possible solutions (the search space). Occasionally, the solutions may be "seeded" in the areas where the optimal solutions are likely to be found. In our case, this is done randomly. We generate a label placement with restricted given area (including rectangle R, track routing area, and space for labels) and let bad offspring eliminated by selection.

4.3 Evaluation

The choice of the evaluation function plays a crucial role in the design of a genetic algorithm. There is a big advantage of using evaluation in genetic algorithm. One can measure desired criteria on the resulting placement and weight these criteria to suit personal preferences. Because genetic algorithm is always used in multi-purpose problem, we can then analyze how important these criteria are. Our concerned criteria are stated as follows: all the leaders do not cross to each other; all the labels do not overlap each other; labels should be placed evenly on East and West sides so that the difference of the sums of the heights of the labels on the two sides is as minimal as possible; the total leader length should be as minimal as possible.

4.4 Selection

Selection is important to genetic algorithm, since only selection drives the search towards more promising regions of the search space. In our implementation, we select individuals for reproduction according to the common linear ranking selection scheme, i.e., individuals are selected according to their rank, with better individuals receiving a higher chance of being selected. So that, the selective procedure of the actual fitness values would be important, since beforehand we do not known the range of fitness values where the optimal solution is located. The algorithm is of the steady state type, i.e., the offspring is introduced into the population, and the worst-fitted individual is deleted. In this way, the best solution so far is never deleted. In addition, if there exists at least a leader crossing in the solution resulting from a certain individual, the individual should not be counted in the population. Our fitness function is defined as follow:

$$f_i = \lambda_1 \times \left(\frac{TLL}{n \times (|t_R - b_R| + |r_R - l_R| + \varepsilon)} \right) + \lambda_2 \times \left(\frac{|R_h - L_h|}{R_h + L_h} \right) \tag{1}$$

where TLL denotes the total leader length; R_h (resp., L_h) is the sum of the heights of the labels on the East (resp., West) side; ε is the width of the track routing area; t_R, b_R, b_R, and l_R are top, bottom, right, and left positions of rectangle R, respectively. In order to normalize the function, we divide these parameters to their intuitive maximum value. Thus, w.l.o.g., λ_1 and λ_2 can be chosen in the range between 0 and 1 under the constraint of $\lambda_1 + \lambda_2 = 1$.

4.5 Recombination

In order to obtain a better result, we combine two good parents into a new offspring which may become better or not. The purpose of the crossover operator is to recombine sub-placements of different individuals to produce an offspring. Since, we expect that good parts of a placement are connected, we perform crossover by choosing randomly a connected parts of the placement of two parents and swapping the sub-placements. However, there is a problem with this operator when using this method. A combination of two good parents may yield a poor offspring. This poor offspring will be deleted during the natural selection process.

4.6 Mutation

Mutation is a crucial step of genetic algorithm. While using recombination, we can only find new combination of individuals that are already present. We may lose some information forever while it is not in the population. Another method, called mutation, can introduce new material into the population, i.e., the slight changing of individuals. It is necessary and reasonable to get new materials to increase the probability of obtaining better answers. In out implementation, mutation is done by randomly changing binary vector with a given small probability. We try to change one leader from the East side to West side (or from the West side to the East side). By doing so, we have a probability to obtain a better individual through the present individual, and the result is different from the recombination (or crossover) process.

5 Experimental Results

We have developed a prototype for the GA algorithm for the two-side labeling. Experimental results under different combinations of λ_1 and λ_2 in Equation (1) are given in this section.

5.1 Total Leader Length Minimization

In some situations, we may focus on the objective of minimizing the total leader length (TLL). We can slightly change λ_1 and λ_2 to fit our destination. To this end, we set $\lambda_1 = 1.0$ and $\lambda_2 = 0.0$.

We consider an example with 20 sites and 20 labels. The experimental results of the example using our genetic algorithm and the brute-force method are given in Table 2. For comparative purpose, we provide the following information: the maximal possible TLL in this example is 12160 units, and the total label height is 14000 units. For the TLL, the TLL of our GA algorithm is 4872 units; the TLL of the optimal solution is 4140 units. We can see that the difference between our GA solution and the optimal solution is only 732 units, which is a small number in relative to the maximal possible TLL – 12160 units. Therefore, our GA algorithm can obtain a solution with TLL not too worse than the optimal TLL. As for the difference of the sums of heights of labels on the two sides (DSHL), our GA solution (36 units) is better than the solution (56 units) with the optimal TLL. Although the TLL is smaller in this setting, bad DSHL results in a unbalanced drawing of labels, as shown in Figure 7 and Figure 10.

Table 2: The experimental results of an example using our GA algorithm under various λ_1 and λ_2 versus the optimal solution.

λ_1	λ_2	solution	total leader length	difference of two-side heights
1.0	0.0	our GA (Figure 7)	4872 units	36 units
		optimal (Figure 10)	4140 units	56 units
0.0	1.0	our GA (Figure 8)	5022 units	8 units
		optimal (Figure 11)	5508 units	0 units
0.5	0.5	our GA (Figure 9)	4860 units	12 units
		optimal (Figure 12)	4154 units	0 units

5.2 Label Height Balance on the Two Sides

In some situation, we may focus on objective of minimizing the difference of the sums of heights of labels on the two sides (DSHL). To this end, we set $\lambda_1 = 0.0$ and $\lambda_2 = 1.0$.

Consider the same example as the previous subsection, in which the maximal possible TLL is 12160 units, and the total label height is 14000 units. The experimental results of the example using our genetic algorithm and the brute-force method are given in Figure 8 and Figure 11; their statistics is given in Table 2. We can see that the difference of the DSHL values between our GA solution and the optimal solution is only 8 units, which is a small number in relative to the maximal possible DSHL – 14000 units. Although the DSHL is smaller in this setting, bad TLL results in a drawing with crowded leaders, as the leaders in Figure 11 are more crowed in track routing areas than in Figure 8.

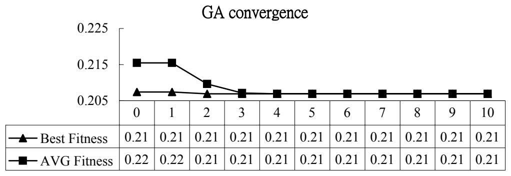
5.3 Total Leader Length Minimization and Label Height Balance at the Same Time

In some situations, we may focus on the objective of minimizing both the total leader length (TLL) and the difference of the sums of heights of labels on the two sides (DSHL). To this end, we set $\lambda_1 = 0.5$ and $\lambda_2 = 0.5$ (i.e., both measures are important evenly).

Consider the same example as the previous subsection, in which the maximal possible TLL is 12160 units, and the total label height is 14000 units. The experimental results of the example using our genetic algorithm and the brute-force method are given in Figure 9 and Figure 12; their statistics is given in Table 2. For the TLL, the TLL of our GA algorithm is 4860 units; the TLL of the optimal solution is 4154 units. We can see that the difference between our GA solution and the optimal solution is only 706 units, which is a small number in relative to the maximal possible TLL – 12160 units. For the DSHL, we can see that the difference between our GA solution and the optimal solution is only 12 units, which is a small number in relative to the maximal possible DSHL – 14000 units. Therefore, it is concluded that our GA algorithm can generate a solution which is not too worse than the optimal solution. We can observe from Figure 9 and Figure 12 that minimizing both the TLL and the DSHL results in a drawing with balanced heights of labels on the two sides and clear leaders in track routing areas.

5.4 Discussion

We show that our GA works as follows. As shown in Figure 13, the average fitness finally converges to the optimal fitness value, and it converges very quickly (see also the third generation in Figure 13). Although in other cases we may see some plots in Figure 13 which are not respected, those plots are due to the mutation process, but we still can find out the tendency of convergence in this cases.



	0	1	2	3	4	5	6	7	8	9	10
▲ Best Fitness	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.21
■ AVG Fitness	0.22	0.22	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.21

Figure 13: The GA convergence with $\lambda_1 = 0.5$ and $\lambda_2 = 0.5$.

In addition, we conduct some experiments with some other combinations of λ_1 and λ_2, as shown in Table 3. Because of different combinations, optimal solutions are also different. Thus, when we would like to compare the results, we have to compare them with their own optimal solutions. According to Table 3, we can find that there is a tendency that the TLL grows larger while focusing on the DSHL, and vice versa. Even though the results of our GA algorithm depend on the initial placement mostly, different combinations still affect them. In fact, the best combination should be defined case by case, so we do not study them more.

Table 3: Statistics of Experimental results

(λ_1, λ_2)	TLL	TLL_opt	DSHL	DSHL_opt
(1.0, 0.0)	4614.0	4140	40.0	56
(0.8, 0,2)	4667.6	4150	51.2	8
(0.6, 0,4)	4670.8	4154	41.6	0
(0.4, 0,6)	4785.0	4154	58.4	0
(0.2, 0,8)	4839.2	4154	31.2	0
(0.0, 1,0)	5020.8	5508	12.0	0

Our GA algorithm also can be applied to MS Word (see Figure 14). Two column spaces on the two sides of the document are used for the label placements. It becomes clearer to distinguish all leaders on the page. For readers, they do not need to turn the pages to find the information about the sites. Readability is also improved.

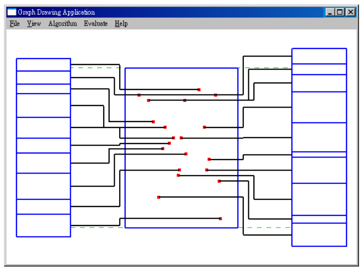


Figure 7: A labeling using our GA algorithm with $\lambda_1 = 1.0$; $\lambda_2 = 0.0$.

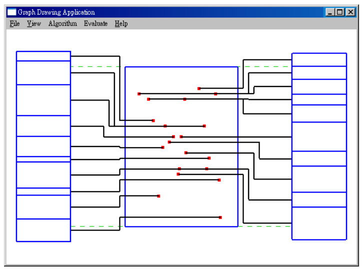


Figure 8: A labeling using our GA algorithm with $\lambda_1 = 0.0$; $\lambda_2 = 1.0$.

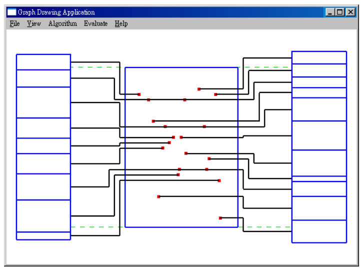


Figure 9: A labeling using our GA algorithm with $\lambda_1 = 0.5$; $\lambda_2 = 0.5$

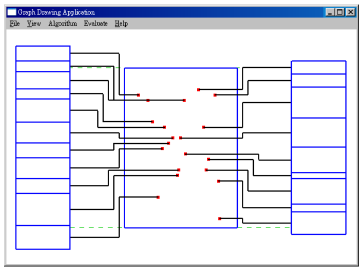


Figure 10: The optimal solution for the example of Figure 7.

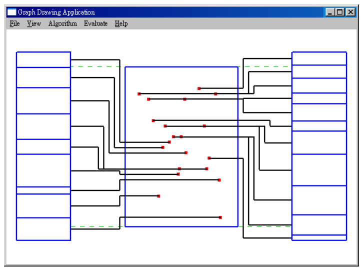


Figure 11: The optimal solution for the example of Figure 8.

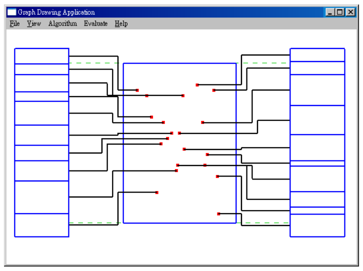


Figure 12: The optimal solution for the example of Figure 9.

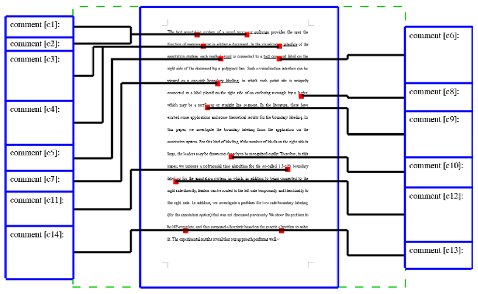


Figure 14: Application to MS Word.

5.5 Comparison between one-side and two side labelings

The relationship between these two approaches is how we would like to improve the text annotation system. Most of the time, we hate to read articles on computers because we get tired easily. Some people may used to print them out. Hence, it is good to consider how to fill one paper with most information. By doing so, we should consider not only the column space for labels, but also how large they are. These two approaches have their own pros and cons that are subjective. Thus, we may provide related parameters for users. Even though there exists a document which can be applied on one-side and two boundary labelings, it is hard to find objective criteria to judge how good they are.

References

[1] M. Bekos, M. Kaufmann, K. Potika, and A. Symvonis. Boundary labelling of optimal total leader length. In *Proc. of the 10th Panhellenic Conference on Informatics (PCI 2005)*, volume 3746 of *LNCS*, pages 80–89, 2005.

[2] M. Bekos, M. Kaufmann, K. Potina, and A. Symvonis. Area-feature boundary labeling. *The Computer Journal*, 2008. To appear.

[3] M. Bekos, M. Kaufmann, A. Symvonis, and A. Wolff. Boundary labeling: models and efficient algorithms for rectangular maps. In *Proc. of the 12th International Symposium on Graph Drawing (GD 2004)*, volume 3383 of *LNCS*, pages 49–59, 2004.

[4] M. Bekos, M. Kaufmann, A. Symvonis, and A. Wolff. Boundary labeling: models and efficient algorithms for rectangular maps. *Computational Geometry: Theory and Applications*, 36(3):215–236, 2006.

[5] M. Benkert, H. Haverkort, M. Kroll, and M. Nöllenburg. Algorithms for multi-criteria one-sided boundary labeling. In *Proc. of the 15th International Symposium on Graph Drawing (GD 2007)*, volume 4875 of *LNCS*, pages 243–254, 2008.

[6] B. Chazelle and 36 co-authors. The computational geometry impact task force report. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, volume 223, pages 407–463. AMS, 1999.

[7] M. Formann and F. Wagner. A packing problem with applications to lettering of maps. In *Proc. of the 7th Annual ACM Symposium on Computational Geometry (SoCG 1991)*, pages 281–288. ACM Press, 1991.

[8] J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

[9] E. Imhof. Positioning names on maps. *The American Cartographer*, 2(2):128–144, 1975.

[10] C. Iturriaga and A. Lubiw. NP-hardness of some map labeling problems. Technical Report CS-97-18, University of Waterloo, 1997.

[11] C.-C. Lin, K.-R. Kao, and H.-C. Yen. Many-to-one boundary labeling. *Journal of Graph Algorithms and Applications*, 12(3):319–356, 2008.

[12] G. Neyer. Map labeling with application to graph drawing. In D. Wagner and M. Kaufman, editors, *Drawing graphs: Methods and models*, volume 2025 of *LNCS*, pages 247–273. 2001.

[13] F. Wagner. Approximate map labeling is in $\omega(n \log n)$. *Information Processing Letter*, 52(3):161–165, 1994.

[14] F. Wagner and A. Wolff. Map labeling heuristics: Provably good and practically useful. In *Proc. of the 11th Annual ACM Symposium on Computational Geometry (SoCG 1995)*, pages 109–118. ACM Press, 1995.

[15] A. Wolff and T. Strijk. The map-labeling bibliography. http://i11www.ira.uka.de/map-labeling/bibliography/, 1996.