Minimum-Change Drawings for Labeled Graphs*

MICHAEL A. BEKOS, KATERINA POTIKA, ANTONIOS SYMVONIS

National Technical University of Athens
School of Applied Mathematical & Physical Sciences
15780 Zografou, Athens
GREECE

Abstract: The interest in algorithms that automate the process of information visualization by the usage of textual labels for features of interest within a visualization has increased with the advance in type-setting technology and the amount of information to be visualized. In this paper, we study *minimum-change drawings for labeled graphs*, adopting the boundary labeling model [6]. Changes within the graph may occur because of the actions of a user, which is able to insert new features within the visualization, delete previously placed features and change the sizes of the labels. Each time the user acts, a small change should be performed within the visualization in order to rearrange its components with respect to some optimization criterion. We model this problem and we also provide several heuristics to obtain "near-optimal" solutions.

Key–Words: Labeled graphs, Minimum-change drawings, Boundary Labeling.

1 Introduction

A common task in the process of information visualization is the placement of textual labels for features of interest within the visualization. The interest in algorithms that automate this task has recently received considerable attention, due to the numerous applications that stem from diverse areas such as Cartography and Graphical Information Systems.

Recent research on automated label placement has concentrated on placing the textual labels as close to the features of interest as possible, so that the labels do not overlap each other. Due to the computational complexity of this problem, which is NP-hard in general [7, 9, 11, 12], cartographers, graph drawers, and computational geometers have proposed several approaches, among them approximation algorithms [7, 13], expert systems [1], gradient descent [8], zero-one integer programming [14] and simulated annealing [15]. However, there exist cases in which these approaches are difficult or even undesirable to be applied. Such cases arrive when the labels are large enough (i.e. they contain blocks of texts rather than a single word), or when the features lie close to each other and there is not enough space to place the labels. In addition, there exist cases, where the underlying map contains useful information and must not be obscured or occluded by the use of labels.

As a response to this problem Bekos, Kaufmann, Symvonis and Wolff [6] proposed a reasonable alternative, according to which the labels are placed on the boundary of a rectangle enclosing the drawing and they are connected to their associated features with non-intersecting polygonal lines. This approach is denoted as *boundary labeling*, and the lines are called *leaders*.

In practice, this approach is quite usual in medical maps and technical drawings, where it is often common to explain certain features of the drawing with blocks of text, that are arranged on its boundary. In such a setting, labels containing long texts do not overlap each over, do not obscure the site set and more importantly they do not occlude the underlying drawing.

*The work is co - funded by the European Social Fund (75%) and National Resources (25%) - Operational Program for Educational and Vocational Training II (EPEAEK II) and particularly the Program PYTHAGORAS.

Adopting the boundary labeling model, we investigate *minimum-change drawings for labeled graphs*. More precisely, we assume that a labeling is being created interactively based on the actions of a user. Each time the user acts (e.g. inserts a new site or deletes a site), a small change should be performed within the visualization in order to rearrange its components with respect to some optimization criterion. Clearly, this could be done by applying an optimal algorithm each time the user acts. However, this is a time consuming task, especially when the number of features to be labeled is large. In addition, such an approach could lead into confusing situations for the user's point of view, since the new labeling could be completely different from the one obtained before the user's action has taken place. Therefore, we propose an alternative to cope with this problem, relaxing our demands on optimality and focusing on the user's point of view.

This paper is structured as follows: In Section 2, we define formally the boundary labeling model. Section 3, reviews previous results. In Section 4, we focus on minimum-change drawings for labeled graphs. We conclude in Section 5 with open problems and future work.

2 The boundary labeling model

A boundary labeling in its primitive form specifies as part of its input a set P of n sites $s_i, i = 1, 2, \ldots n$, 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 each over and to be connected to their corresponding sites with non-intersecting polygonal lines, called *leaders*. Such labelings are referred to as *legal* or *crossing free labelings*.

In boundary labeling, the type of the leaders used to produce labelings, is one of the most important aspects that determine the readability and the ambiguity of the drawing. Surprisingly, it is not always the best choice to rely exclusively on straight-line leaders (also refereed to as type-*s* leaders) . The reason is that the variety of different slopes among the leaders may unnecessarily clutter the visualization, especially if the number of labels is large. Leaders look less disturbing if their shapes are more uniform, as the rectilinear leaders do. When a leader is rectilinear, it consists of a sequence of axis-parallel segments, which are either parallel (p) or orthogonal (o) to the side of R containing the label it leads to. This implies that a leader c of type $c_1 c_2 \ldots c_k$, where $c_i \in \{o, p\}$ consists of a x- and y-monotone connected sequence $(s_1, s_2, \ldots, s_k)$ of segments from the site to the label, where segment s_i is parallel to the side of R containing the label if $c_i = p$; otherwise it is orthogonal to that side. Since we target on simple and easy to visualize labelings, we focus only on leaders of types *po* and *opo* . For each *opo* leader, we further insist that its p segment lies outside R, in the so called *track routing area*, so that the visualization is not cluttered unnecessarily. Leaders of type o are trivially considered to be of type *opo* and *po*, as well.

The labels are attached on the boundary of an axis-parallel rectangle $R = [l_R, r_R]\text{x}[b_R, t_R]$ of height $H = t_R - b_R$ and width $W = r_R - l_R$, which contains the site set P. In the general case of the problem, the labels are of arbitrary sizes (*non-uniform labels*), i.e. label l_i, which corresponds to site s_i has height h_i and width w_i. However, of particular interest are the labels of *uniform size*, or of *maximum uniform size*.

Keeping in mind that we want to obtain simple and easy to visualize labelings, the following criteria can be adopted from the areas of VLSI and graph drawing: (1) *minimize the total number of bends* used for the leaders, i.e. determine a legal labeling, such that the total number of leader bends is minimum and (2) *minimize the total leader length*, i.e. determine a legal labeling, such that the total leader length is minimum.

3 Previous Work

Bekos, Kaufmann, Symvonis and Wolff introduced the boundary labeling problem in [6]. The focus of their work is on efficient algorithms for minimizing the total leader length and for minimizing the total number of leader bends. An algorithm for minimizing the total leader length with type-*opo* leaders, when the labels are allowed to be placed on all four sides of R is presented in [2]. In subsequent works Bekos et al. study variations, where the labels are arranged in multiple stacks on one side of the rectangle [3] or where a set of polygons is labeled instead of a set of fixed

points [4]. In [5], they presented another variation, where the sites to be labeled are collinear.

Kao, Lin and Yen [10] introduced the term *Many-to-One boundary labeling* to describe a variation of boundary labeling, where several sites are associated with a common label. In the case of Many-to-One boundary labeling, the presence of crossings among leaders often becomes inevitable. Therefore, they presented several algorithms, approximations and heuristics for minimizing the total number of crossings.

4 Minimum-Change Drawings for Labeled Graphs

In this section, we adopt the boundary labeling model and we study minimum-change drawings for labeled graphs.

We assume that a labeling is being created interactively based on the actions of a user (e.g. insertion of a new site). This suggests that, small changes should be performed within the visualization, each time the user acts, otherwise, the user may be confused. Therefore, we can not apply an optimal algorithm each time the user acts, since the resulting visualization may be completely different from the one of some previous stage (i.e. before the user's action takes place). Figure 1 illustrates this case. Both Figures 1a and 1b are optimal in terms of the number of leader bends, however the insertion of the new site (Yorkshire; pointed by an arrow) has introduced a "big change" in the labeling of Figure 1b compared with the one of Figure 1a. This is because several labels have been rearranged and some of them have changed side, in order to maintain the optimality of the latter visualization.

A reasonable alternative to cope with this problem is to relax our demands regarding the optimality of the visualization and focus on the user's point of view.

4.1 User Actions and their Consequences

Since this is our first attempt to model this problem, we assume that the user is only allowed to insert and delete sites and change the size of the labels. In more complicated cases, the user may be able to zoom and navigate over the map.

(a) Before the insertion of the new site.

(b) After the insertion of the new site, which is pointed out by the arrow.

Figure 1: Both labelings are optimal in terms of total leader bends.

The actions of the user influence the structure of the visualization. When the user inserts a new site, then sufficient space must be allocated in order to accommodate a new label, associated with this site. This implies that other neighboring labels might have to be rearranged somehow in order to realize this placement. Additionally, the placement has to be done so that to obtain a "near-optimal" solution. However, since the labeling is not optimal anymore, this may imply crossings among the leaders, which is a drawback, as crossings are often regarded as the main source of confusion in information visualization. This is exactly the point where the optimal algorithm comes into play. If the number of crossings exceeds a previously defined quantity (e.g. the number of sites

being used), then the optimal algorithm should be executed and the user must be informed, that he will face a greater change within the visualization being created.

When a site is removed, then its associated label has also to be removed. This implies that some space is released, so that some bottlenecks to be improved. Keeping in mind that small changes should be performed, a good choice would be to rearrange some of its neighboring labels so that to reduce the number of crossings or archive a better solution regarding the optimization criterion we have adopted.

Finally, the user is also able to change the size of the labels. Such an action can be done either manually because of a user's demand or can be generated because of some change on the font format or font size. In this case, the label has to enlarge or shrink, which suggests that a new change should be performed within the structure of the visualization. Both cases can be treated by following similar approaches as the ones for site creation and deletion, respectively.

4.2 Greedy Heuristics

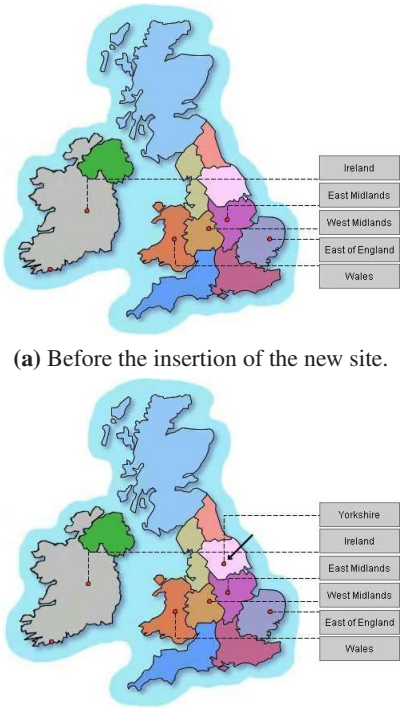
To evaluate our heuristics, we adopt the following boundary labeling model: The labels are allowed to be placed on one side of the enclosing rectangle R, say the east side of R. The assumed model is quite general, since it permits non uniform labels with sliding label ports (i.e. each leader is allowed to touch the label anywhere on its boundary). We seek to determine a near-optimal type-po boundary labeling with respect to the total number of leader bends.

4.2.1 Insertion of Sites

The heuristic regarding the site creations is stated as follows: Each time a new site is being created, determine whether it can be routed using a type-o leader. In a straight-forward approach, this can be determined in $O(n)$ total time, where n is the number of labels already placed. However, we can improve ourselves in $O(\log n)$ by applying a binary search over the label set. If this is not possible, then the label is placed so that to minimize the number of crossings implied by this placement. Again, in a straight-forward approach this can be done in $O(n^2)$ total time, however we can improve ourselves in $O(\log^3 n + k \log n)$, where k is the maximum number of reported crossings by employing binary search like techniques over the label set and interval trees to support visibility queries. Our heuristic is also stated in Algorithm 1.

Algorithm 1: Greedy Heuristic on Site Creations

On site creation (s)**:**
Determine whether s can be routed with a type-o leader.
if (this is possible) **then**
  Proceed with the placement.
else
  Place the label so that to minimize the number of newly introduced crossings.
if (# crossings exceeds # sites) **then**
  Inform the user.
  Run the optimal algorithm.



(a) Before the insertion of the new site.

(b) After the insertion of the new site, which is pointed out by the arrow.

Figure 2: A labeling produced by Algorithm 1.

Figure 2 illustrates the usage of our heuristic. The insertion of the new site (Yorkshire; pointed by an arrow in Figure 2b) has introduced an additional crossing in the resulting labeling of Figure 2b compared

with the one of Figure 2a. Note that in this case, this crossing can be easily eliminated, however in more complicated cases such an elimination may require several labels to be rearranged.

4.2.2 Deletion of Sites

As mentioned earlier, when a site is deleted some space is released. We choose to use this space in order to increase the number of sites that are routed using a type-o leader, if this is possible. We could also use it so that to reduce the number of crossings or combine with these objectives.

Our heuristic regarding the site deletions is stated as follows: Each time a site is deleted, determine whether one of its label's two immediate neighbor labels can be rearranged, so that to be routed using a type-o leader. This can be done in constant time assuming that each label keeps a reference to its two immediate neighbor labels. We proceed by recursively applying the same approach to the rearranged label. Thus, the total time needed to delete a site is at most $O(n)$, where n is number of labels already placed. Clearly, this recursion does not result into an optimal elimination of the leader bends that can be eliminated. However, it is efficient in term of time complexity. This heuristic is also stated in Algorithm 2.

Figure 3 illustrates the usage of our heuristic. The deletion of the site (East Midlands; pointed by an arrow in Figure 3a) result into a labeling with greater number of type-o leaders. However, it did not affect the number of crossings. Note that in general this procedure can lead into a labeling with greater number of leader crossings.

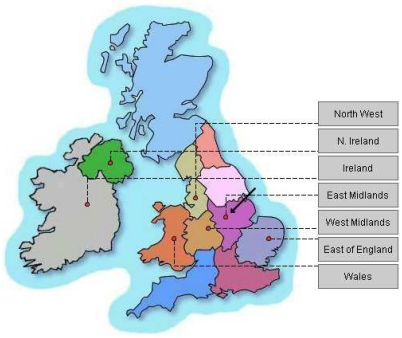
4.2.3 Changing a label's size

Changes on labels' sizes are treated by following similar approaches as the ones for site insertions and deletions. The case where a label is about to shrink is easy. We do not have to consider special issues. We just shrink it, such that its leader do not bend, if it is of type-o. The space that is earned can be exploited as in the case of site deletion. Therefore, such a change demands a linear time effort.

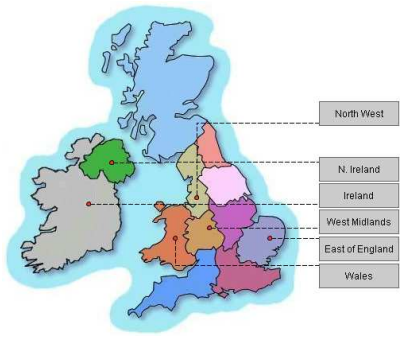
If a label is about to enlarge, then there exist two alternatives. The first one occurs, when there exists enough space (either on top or bellow the label), in order to accommodate this enlargement. In this case, we just enlarge it by focussing on the type of the leader

Algorithm 2: Greedy Heuristic on Site Deletions

On site deletion (s)**:**
l:= the label of site s.
l_t:= the label on top of l.
l_b:= the label bellow l.
Remove l.
if (l_t can be routed using a type-o leader) **then**
  Proceed with the new placement of l_t .
  Recursively, apply the same approach to the two immediate neighbor labels of l_t.
else
  if (l_b can be routed using a type-o leader) **then**
    Proceed with the new placement of l_b .
    Recursively, apply the same approach to the two immediate neighbor labels of l_b.
if (# crossings exceeds # sites) **then**
  Inform the user.
  Run the optimal algorithm.



(a) Before the deletion of the site, which is pointed out by the arrow.



(b) After the deletion of the site.

Figure 3: A labeling produced by Algorithm 2.

that we will use, since such an enlargement may lead into an additional leader of type-*o*. To cope with this case demands a constant time effort. The second alternative occurs when there exists no enough space. In this case, some labels either on top or bellow the label (that is about to enlarge) have to be rearranged. From these two alternatives we choose the one which maximizes the total number of type-*o* leaders. This can be done in linear time.

5 Conclusions

In this paper, as a first step towards solving this problem, we presented results only for the leader bend minimization problem. No results presented regarding the total leader length minimization problem. We also assumed that the user is only allowed to insert and delete sites and change the size of the labels. In more complicated cases, the user may be able to change the sizes of the labels, or even zoom and navigate over the map. No algorithms exist for this model.

References:

[1] J. Ahn and H. Freeman. AUTONAPan expert system for automatic map name placement. In *Proc. International Symposium on Spatial Data Handling (SDH'84)*, pages 544–569, 1984.

[2] M. A. Bekos, M. Kaufmann, K. Potika, and A. Symvonis. Boundary labelling of optimal total leader length. In P. Bozanis and E. Houstis, editors, *10th Panhellenic Conference on Informatics (PCI'05)*, pages 80–89, 2005.

[3] M. A. Bekos, M. Kaufmann, K. Potika, and A. Symvonis. Mutli-stack boundary labeling problems. In S. Arun-Kumar and N. Garg, editors, *Proc. 26th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS2006), LNCS 4337*, pages 81–92, 2006.

[4] M. A. Bekos, M. Kaufmann, K. Potika, and A. Symvonis. Polygons labelling of minimum leader length. In M. Kazuo, S. Kozo, and T. Jiro, editors, *Proc. Asia Pacific Symposium on Information Visualisation (APVIS2006), CRPIT 60*, pages 15–21, 2006.

[5] M. A. Bekos, M. Kaufmann, and A. Symvonis. Labeling collinear sites. In *Proc. Asia Pacific Symposium on Information Visualisation (APVIS2007), IEEE*, pages 45–51, 2007.

[6] M. A. Bekos, M. Kaufmann, A. Symvonis, and A. Wolff. Boundary labeling: Models and efficient algorithms for rectangular maps. In J. Pach, editor, *Proc. 12th Int. Symposium on Graph Drawing (GD'04), LNCS 3383*, pages 49–59, New York, 2005.

[7] M. Formann and F. Wagner. A packing problem with applications to lettering of maps. In *Proc. 7th Annu. ACM Sympos. Comput. Geom.*, pages 281–288, 1991.

[8] S. A. Hirsch. An algorithm for automatic name placement around point data. *The American Cartographer*, 9(1):5–17, 1982.

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

[10] H.-J. Kao, C.-C. Lin, and H.-C. Yen. Many-to-one boundary labeling. In *Proc. Asia Pacific Symposium on Information Visualisation (APVIS2007)*. To appear.

[11] T. Kato and H. Imai. The NP-completeness of the character placement problem of 2 or 3 degrees of freedom. In *Record of Joint Conference of Electrical and Electronic Engineers in Kyushu*, page 1138, 1988. In Japanese.

[12] S. H. Poon, C.-S. Shin, T. Strijk, T. Uno, and A. Wolff. Labeling points with weights. *Algorithmica*, 38(2):341–362, 2003.

[13] M. van Kreveld, T. Strijk, and A. Wolff. Point labeling with sliding labels. *Conputational Geomentry: Theory and applications*, 13:21–47, 1999.

[14] S. Zoraster. The solution of large 0-1 integer programming problems encountered in automated cartography. *Operations Research*, 38(5):752–759, 1990.

[15] S. Zoraster. Practical results using simulated annealing for point feature label placement. *Cartography and GIS*, 24(4):228–238, 1997.