Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 0, no. 0, pp. 0–0 (0)

Many-to-One Boundary Labeling

Chun-Cheng Lin[1] *Hao-Jen Kao*[1] *Hsu-Chun Yen*[1,2,*]

[1]Department of Electrical Engineering
National Taiwan University, Taipei, Taiwan 106, ROC
{sanlin, khr}@cobra.ee.ntu.edu.tw yen@cc.ee.ntu.edu.tw

[2]Department of Computer Science
Kainan University, Taoyuan, Taiwan 338, ROC

Abstract

In *boundary labeling*, each point site is uniquely connected to a label placed on the boundary of an enclosing rectangle by a *leader*, which may be a rectilinear or straight line segment. To our knowledge, all the results reported in the literature for boundary labeling deal with the so-called one-to-one boundary labeling, i.e., different sites are labelled differently. In certain applications of boundary labeling, however, more than one site may be required to be connected to a common label. In this case, the presence of crossings among leaders often becomes inevitable. Minimizing the total number of crossings in boundary labeling becomes a critical design issue as crossing is often regarded as the main source of confusion in visualization. In this paper, we consider the crossing minimization problem for *multi-site-to-one-label boundary labeling*, i.e., finding the placements of labels and leaders such that the total number of crossings among leaders is minimized. We show the crossing minimization problem to be NP-complete under certain one-side and two-side labeling schemes. Subsequently, approximation algorithms or heuristics are derived for the above intractable problems.

Article Type	Communicated by	Submitted	Revised
regular paper	Seok-Hee Hong	June 2007	February 2008

* Corresponding author. Research supported in part by NSC Grant 96-2221-E-002-027, and Research Grant 95-EC-17-A-02-S1-049, Taiwan.

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 points, lines, or areas) 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 [13]. A detailed bibliography and survey on map labeling can be found in [21], [17], respectively. ACM Computational Geometry Impact Task Force [7] has identified label placement as an important area of research. The majority of map labeling problems are known to be NP-complete [11, 14] in general. (The interested reader is also referred to [19, 20] for various approximations and heuristics for map labeling.)

Map labeling problems are classified by the following three kinds of graphical features according to their dimensions, namely, point features, line features, and area features. For example, in a geographical map, a city (resp., river and lake) is typically represented by a point (resp., line and area) feature. Note that a point or a line feature label is normally located next to the associated object, while an area feature label is usually placed within the boundary of the feature to be labeled.

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 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.

For the work reported in [1, 2, 3, 4] regarding boundary labeling, each label is uniquely associated with a site (point feature). In practice, however, it is not uncommon to see more than one site to be associated with the same label. Such examples include the religion distribution in each state of a country, the language distribution of the world, or the association or organization composed of some countries in the world, etc. In view of the above, in this paper we investigate the *multi-site-to-one-label boundary labeling* (a.k.a. *many-to-one boundary labeling*) in which the mapping from sites to labels is a many-to-one function, i.e., more than one site is allowed to be connected to a common label and each site is connected only by a leader. Unlike the conventional boundary labeling, this kind of labeling inevitably leads to a high possibility of crossings among leaders in the drawing. Therefore, an important objective for many-to-one boundary labeling is to find the placements of labels and leaders such that the total number of crossings among leaders is minimized. Aside from minimizing the total number of crossings, we also consider the issue of minimizing the total leader length under the framework of many-to-one boundary labeling in this paper.

Labeling key components of a motherboard is an example used in [5] for illustrating the usefulness of the technique of one-to-one boundary labeling. For motherboards used in servers or parallel computers, it is common to find multiple copies of components such as CPUs, chipsets, memory DIMMs, I/O ports, expansion slots, and so on, on the same motherboard. In this case, placing labels along the sides of a motherboard involves connecting multiple sites to a single label, suggesting an example to which many-to-one boundary labeling can apply. Figure 1 gives the boundary labeling of the ASUS KFN5-Q/SAS motherboard in a many-to-one fashion. For comparison purpose, the motherboard is also labeled by two other approaches in Figure 2, where *area labeling* in Figure 2(a) places a text label within the boundary of each object, and *legend labeling* in Figure 2(b) attaches an assigned number to each object of the same component, and places a legend table with those numbers as well as the text information of their corresponding components on the right side of the motherboard.

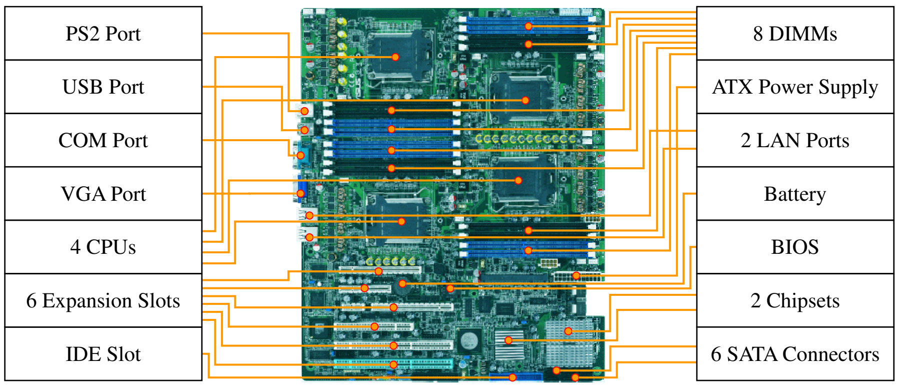
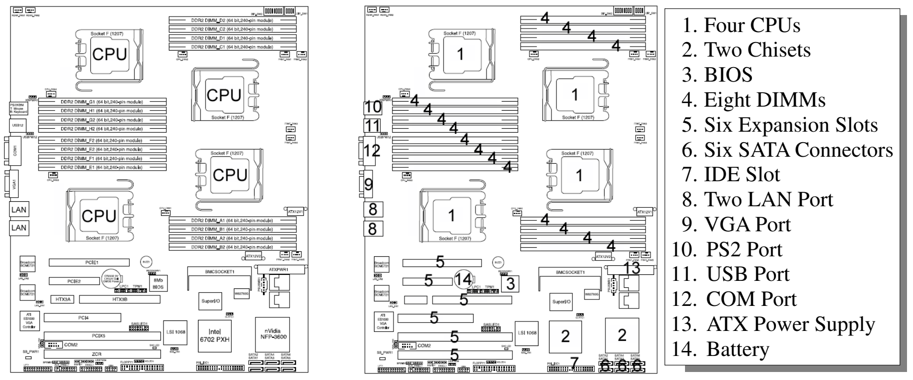


Figure 1: An example of many-to-one boundary labeling.

In comparison with Figure 2(a) (where text labels are of different sizes causing some of them to be too tiny to read), Figure 1 displays clearer and more readable text labels on such a dense motherboard, though more space is required. If the motherboard is colored (as shown in Figure 1), then the area labeling which places lots of redundant texts inside the motherboard tends to cause unnecessary confusion in visualization. As for Figure 2(b), the legend labeling can be viewed as an alternative to many-to-one boundary labeling. In practice, choosing the right labeling scheme often depends on the application, and an integrated solution might turn out to be better in some cases.

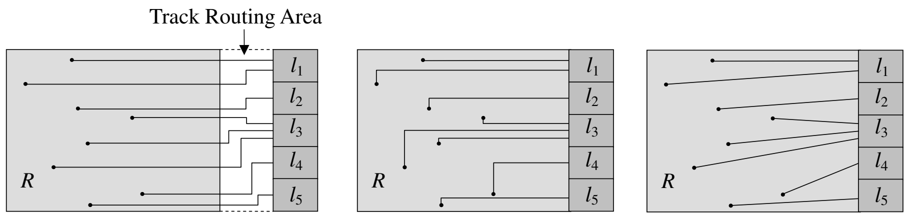
Conventionally, boundary labeling is identified as *k-side labeling with type-t leaders* (where $k \in \{1, 2, 4\}$ and $t \in \{opo, po, s\}$) if the labels are allowed to be



(a) Area labeling. (b) Legend labeling.

Figure 2: Other labeling approaches.

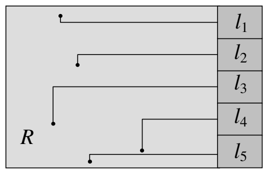
attached to the k sides of the enclosing rectangle R by only type-t leaders. The parameter t specifies the way a leader is drawn to connect a site to a label. The *opo*, *po*, and *s* stand for *orthogonal-parallel-orthogonal*, *parallel-orthogonal* and *straight-line*, respectively, which can easily be understood from the examples given in Figures 3 (a), (b) and (c). 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 3 (a).



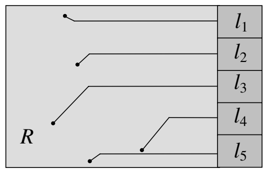
(a) Type-*opo* leaders (b) Type-*po* leaders (c) Type-*s* leaders

Figure 3: Illustration of leaders.

Very recently, in order to improve one-side one-to-one boundary labeling with type-*po* leaders, Benkert et al. [6] introduced a new notation of so-called type-*do* leaders, in which the *do* stands for *diagonal-orthogonal*. As shown in Figure 4, the only difference between type-*po* and type-*do* leaders is that the leader starts with a diagonal segment of fixed angle oriented towards the label. They suggested to apply type-*do* leaders to producing smoother shapes of leaders such that it becomes easier to comprehend the assignment from sites to labels. Intuitively, we observe from Figure 4 that the model of type-*do* leaders can obtain shorter total leader length than that of type-*po*. Therefore, they investigated the problem of minimizing the total leader length, without any crossings of leaders.



(a) Type-*po* leaders



(b) Type-*do* leaders

Figure 4: Comparison between type-*do* and type-*do* leaders in one-to-one boundary labeling.

As listed in Table 1, the main contributions of this paper include:

1. Crossing minimization problems for both one-side and two-side many-to-one labeling with type-*opo* leaders are proved to be NP-complete (Sections 3 and 4). We also design approximation algorithms to cope with such intractable problems.

2. Crossing minimization problems for one-side and two-side many-to-one labeling with type-*po* leaders are proved to be NP-complete (Sections 5 and 6). Heuristic algorithms with satisfactory experimental results are also given for these problems.

3. In Section 7, we discuss the many-to-one labeling with the objective of minimizing the total leader length to be solvable in polynomial time, along a similar line of the work of [1].

Table 1: The main contributions of this paper.

objective	number of sides	leader type	time complexity	solution
Minimize the crossing number	one	*opo*	NPC	approximation
Minimize the crossing number	two	*opo*	NPC	approximation
Minimize the crossing number	one	*po*	NPC	heuristic
Minimize the crossing number	two	*po*	NPC	heuristic
Minimize total leader length	any	*opo*, *po*	P	following [1, 4]

A wide variety of variants of one-to-one boundary labeling have been proposed and studied from an algorithmic viewpoint in the literature. Table 2 summarizes those that are related to our work. By comparing Table 1 with Table 2, it is interesting to note that in the one-to-one case, minimizing the total leader length while respecting the no-crossing constraint is always tractable, whereas in the many-to-one case, minimizing the crossing number becomes intractable. Also note that the total leader length minimization problem remains solvable in polynomial time even in the many-to-one case.

Table 2: Variants of one-to-one boundary labeling problems and their complexities.

objective	number of sides	leader type	time complexity	reference
Minimize total leader length*	one, four	s	$O(n^{2+\epsilon})$	[3, 4]
Minimize total leader length*	one	opo	$O(n \log n)$	[3, 4]
Minimize total leader length*	two	opo	$O(n^2)$	[3, 4]
Minimize total leader length*	four	opo	$O(n^2 \log^3 n)$	[1, 4]
Minimize total leader length*	one, two	po	$O(n^2)$	[3, 4]

* All the problems are subject to the constraint that there are no crossings among leaders; the positions of ports are fixed; each label is of uniform (maximum) size.

In one-to-one boundary labeling, it is always possible to find a layout without crossings among leaders; in the many-to-one case, however, leader crossings are inevitable in general. This disparity is exactly the reason why minimizing the number of crossings is the most critical issue in many-to-one boundary labeling.

2 Preliminaries

A *k-side type-t many-to-one boundary labelled map* (or *k-side type-t map*, for short), where $k \in \{1, 2, 4\}$ and $t \in \{opo, po, s\}$, is $\mathcal{M} = (P, L, n_1, n_2, n_3, n_4, f)$ where

- $P = \{p_1, p_2, ..., p_N\}$, $p_i \in \mathbb{R}^2, 1 \leq i \leq N$, is the set of sites (points) on the plane enclosed in a rectangle R,
- L is the set of *labels* with $|L| = n_1 + n_2 + n_3 + n_4$,
- $n_1, n_2, n_3, n_4 \in \mathbb{N}$ are the numbers of *labels* to the *East*, *West*, *South*, and *North*, respectively, of the axis-parallel rectangle enclosing all sites in P,
- $f : P \rightarrow L$ is an onto function which assigns each site in P to a label in L. Note that f is a many-to-one function in general.

W.l.o.g., we assume that for k=1 (resp., 2), labels are only placed on the *East* side (resp., *East* and *West* sides) of the enclosing rectangle of P. Hence, $n_2 + n_3 + n_4 = 0$ for $k = 1$ and $n_3 + n_4 = 0$ for $k = 2$. On the other hand, labels can be placed on the four sides of the rectangle when $k = 4$. The parameter t, $t \in \{opo, po, s\}$, refers to the type of *leaders* allowed for connecting sites to labels. The opo, po, and s stand for *orthogonal-parallel-orthogonal*, *parallel-orthogonal* and *straight-line*, respectively. See Figure 3 for these three types of leaders. For notational convenience, we refer to the *East*, *West*, *South*, and *North* sides to be the 1st, 2nd, 3rd, and 4th sides throughout the rest of this paper. One should also note that every label l is connected with $|f^{-1}(l)|$ sites; hence, l has to have $|f^{-1}(l)|$ ports to which the sites are connected. Although f^{-1} is not a function, the slight abuse of notation is simply for convenience of understanding. For simplicity, we assume that there are no two sites with the same x- or y-coordinates, and the locations of ports of each label are predefined (see label l_3 with three ports in Figure 3 (a)). Note that if part of a leader is overlapped with a certain other leader in a boundary labeling, then the overlapping can be removed by slightly adjusting the location of port of one of the two leaders. Therefore, it is reasonable to assume that leaders never overlap. In addition, the leader connecting site u to label l is denoted by ul.

A *boundary labeling* of a map $\mathcal{M}$ is a sequence of labels $(l_1^1, ..., l_1^{n_1}, l_2^1, ..., l_2^{n_2}, l_3^1, ..., l_3^{n_3}, l_4^1, ..., l_4^{n_4})$ such that $\forall 1 \le i \le 4, 0 \le j \le n_i, l_i^j \in L$. W.l.o.g., we assume that all the labels are different. Intuitively speaking, $l_i^1, ..., l_i^{n_i}$ is the sequence of labels along the i-th side. W.l.o.g., for $i = 1$ and 2 (i.e., East and West sides, resp.) a top-down ordering is assumed; for $i = 3$ and 4 (i.e., South and North sides, resp.) a left-to-right ordering is assumed. Figure 5 illustrates a 4-side type-s boundary labeling. For simplicity, we assume labels (drawn as rectangles) along the same side to be of uniform and maximum size; hence, the ordering of labels along each side is sufficient to determine the exact positions of labels. As f is a many-one function in general, there might be several sites connecting to the same label. For example, three sites are connected to label l_3 in Figures 3(a) and 3(b). It is easy to observe from that to minimize the number of crossings (or the total leader length) in the case of type-*opo* leaders show in Figure 3(a) (resp., type-*po* leaders show in Figure 3(b)), the ordering of ports at which the three leaders touch label l_3 (drawn as a rectangle) must respect the ordering (in increasing order) of the y-coordinates (resp., x-coordinates) of the three sites connected to label l_3. The *crossing number* is the number of crossings among leaders in a drawing.

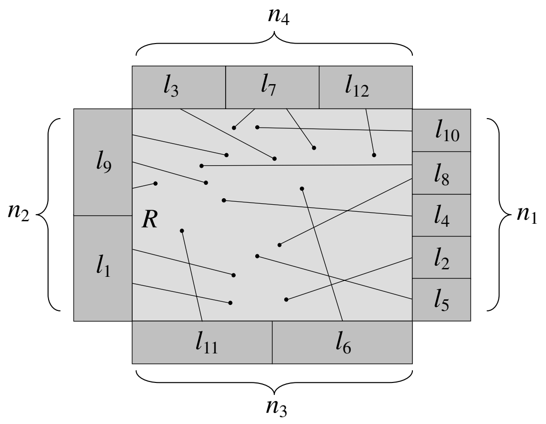


Figure 5: A four-side many-to-one labeling with type-s leaders.

One of the optimization problems considered in this paper is as follows:

The Crossing Problem for k-Side Many-to-One Labeling with Type-t (CPkML-t, for short): Given a k-side type-t map $\mathcal{M}$, determine a boundary labeling for $\mathcal{M}$ so that the crossing number is minimized.

Before deriving our main results, we first recall three NP-complete problems, namely, the *Decision Crossing Problem* (DCP) [8], the *Max-Bisection Problem* [22], and the *Min-Bisection Problem* for directed graphs [10] that are closely related to our problems.

Consider a two-layered network $G = (L_0, L_1, E)$ consisting of disjoint sets L_0 and L_1 of nodes and a set $E \subseteq L_0 \times L_1$ of edges. Assume that the nodes in L_0 and L_1 appear on the vertical lines $x = 0$ and $x = 1$, respectively, and the edges in E are straight line segments joining two nodes from L_0 and L_1 respectively. A drawing of G is generated by assigning each node $v \in L_i$, $i = 0, 1$, to a y-coordinate $y_i(v)$. Hence, two edges uv and tw, where $u, t \in L_0$ and $w, v \in L_1$, cross in the drawing if and only if $(y_0(u) - y_0(t))(y_1(v) - y_1(w)) < 0$. Let the number of crossings in a drawing of G specified by y_0 and y_1 be denoted by $cross(G, y_0, y_1)$. In fact, the crossing number is affected only by the ordering of the nodes of L_0 and L_1, not by their precise positions; so y_0 and y_1 are said the *ordering* of L_0 and L_1, respectively. The DCP is stated as follows:

The Decision Crossing Problem (DCP)
Instance: A two-layered network $G = (L_0, L_1, E)$, an ordering y_0 of L_0, and an integer M.
Question: Is there an ordering y_1 of L_1, so that $cross(G, y_0, y_1) \leq M$?

The *Max-Bisection Problem* is stated as follows. So far the best approximation ratio for the Max-Bisection problem is 1.431 [22].

The Max-Bisection Problem: Given an undirected graph $G = (V, E)$ with non-negative weights $w_{i,j}$ for each edge in E (and $w_{i,j} = 0$ if $(i, j) \notin E$), partition the nodes in V into two sets S and $V \setminus S$ of equal cardinality so that $w(S) := \sum_{i \in S, j \in V \setminus S} w_{i,j}$ is maximized.

Contrary to the Max-Bisection Problem, the *Min-Bisection Problem* is the problem of computing a bisection for an input graph G so that $w(S)$ is minimized. Note that if the graph is directed, then the objective of the problem is to minimize $w(S) := \sum_{(i,j) \in (S, V \setminus S)} w_{i,j}$. The problem for undirected graphs is known to be NP-hard [12], and the problem for directed graphs can be easily shown to be NP-hard as well [10]. For the case of undirected graphs, an $O(\log^{1.5} n)$-approximation algorithm is known [9]. In practice, Kernighan-Lin heuristic (a.k.a., K-L heuristic) [15] is a well-known algorithm for handling the problem for undirected case. As for the case of directed graphs, Feige and Yahalom [10] showed that the Min-Bisection Problem for directed graphs is not approximable at all.

3 The Crossing Problem for One-Side Many-to-One Labeling with Type-*opo* Leaders

In this section, we show that the crossing problem for one-side many-to-one labeling with type-*opo* leaders (CP1ML-*opo*) is NP-complete (Subsection 3.1). We also give an approximation algorithm guaranteed to yield a solution which is less than or equal to three times the optimal solution (Subsection 3.2). Note that in the restricted case in which each label is associated with at most two sides, an analysis used in [16] can be applied to showing the algorithm presented in Subsection 3.2 to be optimal.

3.1 CP1ML-*opo* is NP-complete

Consider the case where all the labels are placed on the East side of rectangle R which encloses all the sites in the given map. Recall that we assume that there are no two sites with the same x- or y- coordinates. In addition, since every leader goes from a site through the right borderline of rectangle R orthogonally, there is no crossing between leaders inside rectangle R. We can arbitrarily adjust the x-coordinate of the bend of every type-*opo* leader in the track routing area (see Figure 3 (a)) so that two leaders cross only when the y-coordinate order of their corresponding sites is different from that of their corresponding labels. That is, the crossing number is affected only by the y-coordinate orders of sites and labels, not by their x-coordinate orders. As a result, if every type-*opo* leader is replaced by a straight line segment and all the sites are on a vertical line, then the problem under consideration is similar to the DCP except our problem allows more than one site to be connected to a common label. In what follows, we show the concerned problem to be NP-complete. In the case where every leader is replaced by a straight line segment and every site is placed on a vertical line, the decision version of the problem can be captured by the *decision many-to-one crossing problem* (DMCP) as follows:

The Decision Many-to-One Crossing Problem (DMCP)
Instance: A two-layered network $G = (L_0, L_1, E)$ where the mapping from nodes in L_0 to nodes in L_1 is a many-to-one function, an ordering y_0 of L_0, an integer M.
Question: Is there an ordering y_1 of L_1, so that $cross(G, y_0, y_1) \leq M$?

Theorem 1 *DMCP is NP-complete.*

Proof: It is clear that DMCP is in NP because we can guess an ordering of L_1 and then check if the crossing number is no more than M in polynomial time.

It remains to show the NP-hardness, which is established by a reduction from DCP as follows.

DCP differs from DMCP only in the restriction that each node in L_0 is connected only by an edge. From a DCP instance $G = (L_0, L_1, E)$, M (note that M refers to both a part of the instance of DCP and the number of crossings

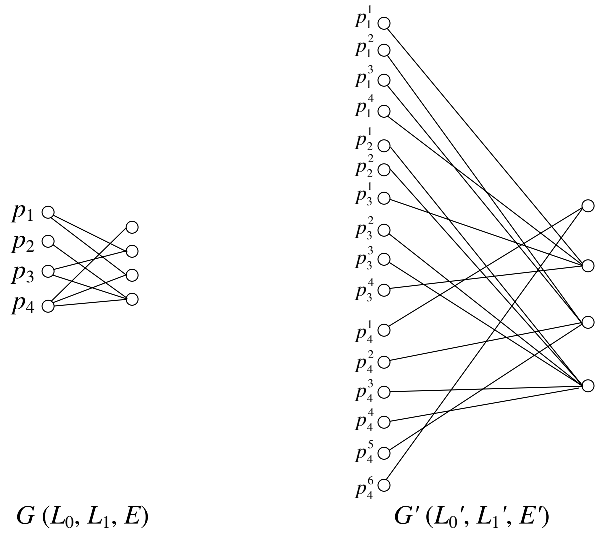


Figure 6: An example reducing from DCP to DMCP.

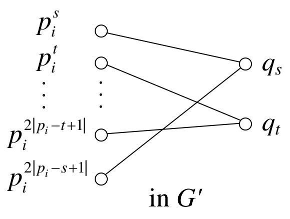


Figure 7: The first category of crossings.

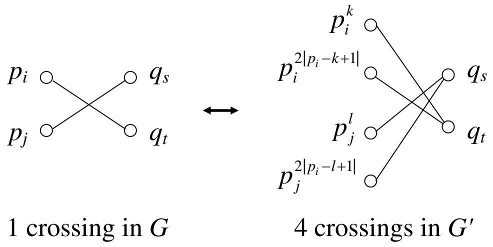


Figure 8: The second category of crossings.

of DCP), we construct a DMCP instance $G' = (L'_0, L'_1, E')$, M' as follows. Let $L'_1 = L_1$. We denote the node with the i-th maximal y-coordinate in L_0 by p_i, and $\{l_i^1, l_i^2, ..., l_i^k\}$ is the set of the nodes in L_1 to which p_i is connected, i.e., there are k edges connecting p_i to k nodes in L_1. For each node p_i in L_0, we create a set of $2k$ nodes, say $P_i = \{p_i^1, p_i^2, ..., p_i^{2k}\}$, in L'_0, where p_i^j has the j-th maximal y-coordinate among all. Then for $j = 1$ to k, we connect p_i^j and p_i^{2k-j+1} to l_i^j. That is, we connect p_i^1 and p_i^{2k} to l_i^1, p_i^2 and p_i^{2k-1} to l_i^2, ... , and p_i^k and p_i^{k+1} to l_i^k. An example is illustrated in Figure 6.

We denote the cardinality of P_i by $|P_i|$ and the number of the nodes in L_0 by $|L_0|$. Let M' be

$$\sum_{i=1}^{|L_0|} 2\binom{|P_i|/2}{2} + 4M$$

We show that there exists an ordering y'_1 of L'_1 such that $cross(G', y'_0, y'_1) \leq M'$ if and only if there exists an ordering y_1 of L_1 such that $cross(G, y_0, y_1) \leq M$.

The crossings in G' can be divided into the following two categories: the edge incident to a node in P_i crosses the edge incident to a node 1) in the same P_i or 2) in P_j for $i \neq j$. For the first category of crossings, as shown in Figure 7, for $i = 1, ..., |L_0|$, for any two pairs of nodes in P_i, $(p_i^s, p_i^{2|p_i|-s+1})$ and $(p_i^t, p_i^{2|p_i|-t+1})$, $s \neq t$, there are two crossings regardless of the order of q_s and q_t, and hence there are $2\binom{|P_i|/2}{2}$ crossings for all permutations of selecting 2 from $\{p_i^1, p_i^2, ..., p_i^{|P_i|}\}$. So the crossing number for the first category is $\sum_{i=1}^{|L_0|} 2\binom{|P_i|/2}{2}$.

For the second category of crossings, as shown in Figure 8, the edge $p_i q_t$ crosses the edge $p_j q_s$ in G if and only if there are the four crossings shown in the right of Figure 8 in G', regardless of the order of q_s and q_t. Therefore, there are $4M$ crossings for the second category in G' if and only if G has M crossings. □

Based on the above, we have the following corollary.

Corollary 1 *CP1ML-opo is NP-complete.*

Proof: (Sketch) To yield the lower bound, it suffices to show a reduction from DMCP to CP1ML-*opo*.

Consider an instance of DMCP, i.e., a two-layered network $G = (L_0, L_1, E)$, ordering y_0 of L_0. With those information, Algorithm 1 constructs a one-side type-*opo* map $\mathcal{M}$.

In what follows, we discuss all the possible cases of whether any two leaders, say $p_a l_b$ and $p_c l_d$, cross in $\mathcal{M}$. W.l.o.g., we assume that the y-coordinate of p_a is greater than that of p_c; leaders $p_a l_b$ and $p_c l_d$ are not straight-line segments (i.e., there are bends on $p_a l_b$ and $p_c l_d$). Note that, in a boundary labeling for type-*opo* leaders, we only need to consider the crossings of leaders in track routing area because all the type-*opo* leaders go from a site through the right borderline of rectangle R orthogonally so that there are no crossings inside rectangle R. We discuss the following two cases: the bends of type-*opo* leaders $p_a l_b$ and $p_c l_d$ are drawn in 1) the same category; 2) different categories. In the first case, w.l.o.g.,

Algorithm 1 DMCP-TO-CP1ML-*opo*

Input: A two-layered network $G = (L_0, L_1, E)$ where the mapping form nodes in L_0 to nodes in L_1 is a many-to-one function, ordering y_0 of L_0, ordering y_1 of L_1 (e.g., see Figure 9(a)).

Output: A one-side type-*opo* map $\mathcal{M}$ (e.g., see Figure 9(b)).

```
1: Construct a one-side type-opo map $\mathcal{M}$ where every site represents every
   node in $L_0$; the locations of sites are determined so that the $y$-coordinate
   order of sites is the same as ordering $y_0$ of $L_0$ while the $x$-coordinate order
   of sites is determined arbitrarily; every label represents every nodes in $L_1$;
   the $y$-coordinate order of labels is the same as ordering $y_1$ of $L_1$.
2: Vertically slice the track routing area of map $\mathcal{M}$ into two regions, where the
   right (resp., left) region is denoted by $T_1$ (resp., $T_2$), as shown in Figure 9(b).
3: Now we connect each leader in $\mathcal{M}$ representing each edge in $E$. According to
   the locations of two endpoints of each leader, all the leaders are classified into
   two categories: the first category is denoted by $C_1$, where the corresponding
   site of every leader has larger $y$-coordinate than its corresponding label port;
   the second category is denoted by $C_2$, which includes the leaders that are
   not in $C_1$.
4: Sort the labels connected by the leaders in each category according to their
   $y$-coordinates.
5: Now we draw every leader as follows. In order not to induce the crossings
   among the leaders connected to a common label, one should notice that the
   $y$-coordinate increasing ordering of the ports at which the leaders touch a
   label must respect the $y$-coordinate increasing ordering of the sites connected
   to the label. As shown in Figure 9(b), the bends of all the type-opo leaders
   in $C_1$ (resp., $C_2$) are drawn in region $T_1$ (resp., $T_2$), and the $x$-coordinate
   increasing order of the bends of the leaders in $C_1$ (resp., $C_2$) respects the
   $y$-coordinate increasing order (resp., $y$-coordinate decreasing order) of their
   corresponding sites.
```

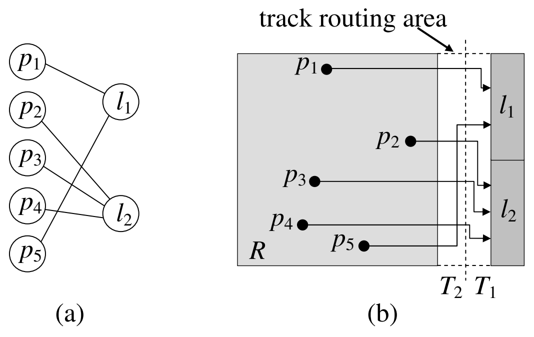


Figure 9: (a) A instance of two-layered network. (b) The boundary labeling corresponding to (a).

we assume the bends of type-*opo* leaders $p_a l_b$ and $p_c l_d$ to be drawn in region T_1. Since from Step 5 of Algorithm 1 the x-coordinate increasing order of the bends of leaders $p_a l_b$ and $p_c l_d$ respects the y-coordinate increasing order of sites p_a and p_c, hence, the two leaders must be drawn as Figure 10(a1) or 10(a2). In the second case, w.l.o.g., we assume the bends of leaders $p_a l_b$ and $p_c l_d$ to be drawn in regions T_1 and T_2, respectively. By Step 5 of Algorithm 1, the two leaders must be drawn as Figure 10(b1), 10(b2), or 10(b3). We observe from Figure 10 that there is one crossing between leaders $p_a l_b$ and $p_c l_d$ if and only if the y-coordinate increasing order of sites p_a and p_c differs from that of labels l_b and l_d. Note that by Algorithm 1 there is at most one crossing between leaders $p_a l_b$ and $p_c l_d$.

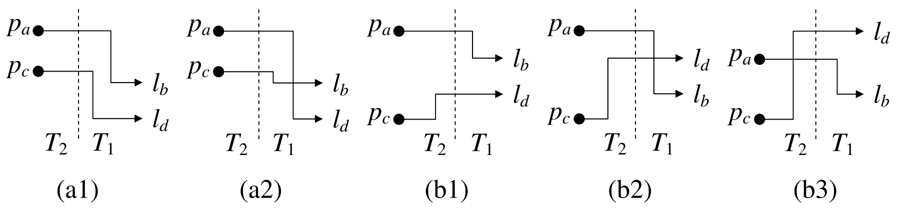


Figure 10: (a) All the possible cases where bends of two leaders are drawn in region T_1. (b) All the possible cases where the bend of leader $p_a l_b$ (resp., $p_c l_d$) is drawn in region T_1 (resp., T_2).

In light of the above, two edges (leaders) $p_a l_b$ and $p_c l_d$ in G (in $\mathcal{M}$) cross if and only if the y-coordinate increasing order of nodes (sites) p_a and p_c differs from that of nodes (labels) l_b and l_d. Hence, we obtain that two edges $p_a l_b$ and $p_c l_d$ in G cross if and only if there is one crossing between two leaders in $\mathcal{M}$. □

3.2 An approximation algorithm

Our approximation algorithm is similar to the so-called *median algorithm* proposed by Eades and Wormald in [8]. The idea of the median algorithm is to place the labels in "*median order*". For convenience and simplicity, we view the labeling problem as finding an ordering y_1 of L_1 in a two-layered network $G = (L_0, L_1, E)$ such that the crossing number is as small as possible. Define N_u of $u \in L_1$ as the nodes $\{v_1, v_2, ..., v_j\} \in L_0$ incident to u. The median order is, for each node $u \in L_1$, to choose the median of the y-coordinates of the neighbors of u as the y-coordinate of u. Precisely, if $N_u = \{v_1, v_2, ..., v_j\}$ with $y_0(v_1) < y_0(v_2) < y_0(v_3) < ... < y_0(v_j)$, then define $med(u) = y_0(v_{\lfloor j/2 \rfloor})$. The median algorithm sets $y_1(u) = med(u)$ for each node $u \in L_1$, and separates the two nodes with the same median by an infinitesimal amount.

The crossing number in the output of the median algorithm is denoted by $med(G, y_0)$, while the crossing number in the output of the optimal labeling is denoted by $opt(G, y_0)$.

Theorem 2 *For all two-layered networks G where the mapping from L_0 to L_1 is a many-to-one function and for all orderings y_0, $med(G, y_0) \leq 3opt(G, y_0)$.*

Proof: Along a similar line of the proof of [8], our proof is given as follows.

Define c_{uv} of $u, v \in L_1$ as the number of crossings that the edges incident to u make with the edges incident to v when $y_1(u) < y_1(v)$. More formally, for $u \neq v \in L_1$,

$$c_{uv} = |\{\{su, tv\} \subseteq E : y_0(s) > y_0(t)\}|$$

and $c_{uu} = 0$. The degree of u is denoted by d_u. For proving the theorem , we claim that if $u, v \in L_1$, and $med(u) \leq med(v)$, then $c_{uv} < 3c_{vu}$. Divide the edges incident with u and v into 4 groups α, β, γ, and δ, where

$$\alpha = \{wu : y_0(w) \leq med(u)\}, \beta = \{wv : y_0(w) \geq med(v)\},$$
$$\gamma = \{wv : y_0(w) < med(v)\}, \delta = \{wu : y_0(w) > med(u)\},$$

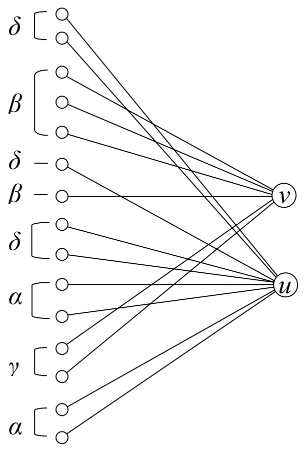


Figure 11: An example for the groups α, β, γ, and δ.

An example is illustrated in Figure 11. Let $a = |\alpha|$, $b = |\beta|$, $c = |\gamma|$, $d = |\delta|$. If $med(u) = y_1(u) \leq med(v) = y_1(v)$, then edges in α cannot cross edges in β. Furthermore, the number of crossings between edges in α and edges in γ is at most ac, and similarly for crossings between edges in β and edges in δ. Also, the number of crossings between edges in γ and edges in δ is at most cd. So

$$c_{uv} \leq ac + bd + cd.$$

Furthermore, if u and v are placed so that $y_1(u) > y_1(v)$, then edges in α must cross edges in β; hence

$$c_{vu} \geq ab.$$

The claim can be discussed by four cases where d_u and d_v are odd or even. In the following we only consider the case where d_u and d_v are both odd; the other cases are similar and simpler.

$$
\begin{aligned}
& a = \frac{d_u+1}{2}, d = \frac{d_u-1}{2}, b = \frac{d_v+1}{2}, c = \frac{d_v-1}{2} \\
& \Rightarrow c_{uv} \le ac + bd + cd \\
& \qquad\quad = \frac{d_u+1}{2} \times \frac{d_v-1}{2} + \frac{d_v+1}{2} \times \frac{d_u-1}{2} + \frac{d_v-1}{2} \times \frac{d_u-1}{2} \\
& \qquad\quad \le \frac{3}{4}(d_u+1)(d_v+1), \\
& \quad c_{vu} \ge ab = \frac{d_u+1}{2} \times \frac{d_v+1}{2} = \frac{1}{4}(d_u+1)(d_v+1) \\
& \Rightarrow c_{uv} \le 3c_{vu}
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
& med(G, y_0) = \sum_{med(u) \le med(v); y_1(u) < y_1(v)} c_{uv} \\
& \le \sum_{u,v \in L_1} 3\min\{c_{vu}, c_{uv}\} \text{ (Since } c_{uv} \le 3c_{vu} \text{ and } c_{uv} \le 3c_{uv}) \\
& \le 3 \sum_{u,v \in L_1} \min\{c_{vu}, c_{uv}\} \le 3opt(G, y_0)
\end{aligned}
$$

□

It should be noticed that finding improved approximation algorithms for the CP1ML-*opo* problem remains an interesting open question.

An experimental result using the approximation algorithm described earlier is given in Figure 12, which illustrates the distribution of some wildlife animals in Taiwan, where leaders are drawn by Algorithm 1. Intuitively many-to-one boundary labeling is more suitable for static maps, for which the leaders allow the user to easily trace the corresponding label of each site. When the number of labels gets larger, such an advantage becomes more obvious.

4 The Crossing Problem for Two-Side Many-to-One Labeling with Type-*opo* Leaders

From our previous result that the crossing problem is NP-complete for CP1ML-*opo*, the intractability result clearly holds for CP2ML-*opo* as well. (CP1ML-*opo* is a special case of CP2ML-*opo* with $n_2 = 0$.) In this section, we consider CP2ML-*opo* under the restriction that $n_1 = n_2$ (i.e., the East and West sides contain the same number of labels). The reason why such a restriction makes sense is given as follows. Recall that we assume labels along the same side to be of equal size. If $n_1 = n_2$ (e.g., see Figure 13(a)), then labels on both sides are of equal size, which may give us a high degree of balance in visibility because labels on two sides can be viewed as a reflectional symmetry along a vertical axis, regardless of leaders and sites. On the other hand, if there is a significant

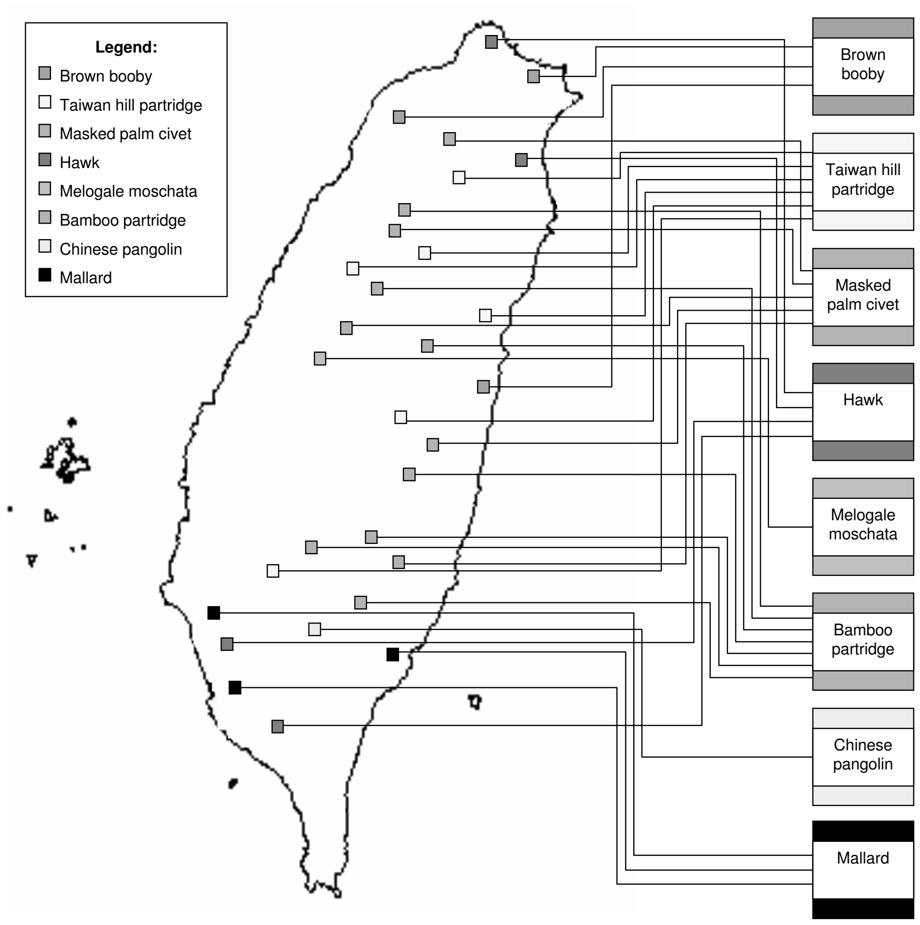


Figure 12: The distribution of some animals in Taiwan, which is represented by one-side many-to-one labeling with type-*opo* leaders.

difference in the number of labels on the two sides, then the texts inside the labels along the denser side may not be readable as Figure 13(b) shows.

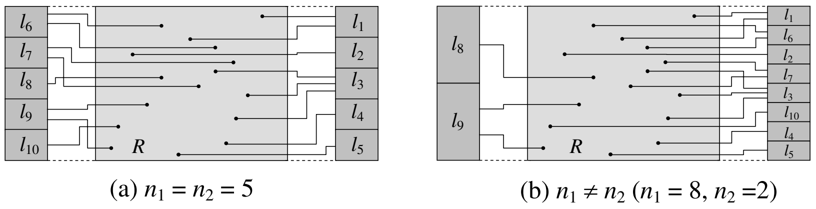


Figure 13: Two boundary labeling layouts with type-*opo* leaders for the same map.

Note that if the number of labels is not even, we can just add a dummy label to make the number even. In this section, we first show the concerned crossing problem to be NP-complete, and then provide an approximation algorithm for the intractable problem.

4.1 CP2ML-*opo* is NP-complete even when $n_1 = n_2$

Since the labeling in this section applies type-*opo* leaders (which is the same as the previous section), hence the crossing number is influenced only by the differences between the y-coordinate orderings of sites and either the labels in the right side or the labels in the left side, respectively. Therefore, the problem can be modeled as an analogy of three-layered network, of which definition is given as follows. A three-layered network $G = (L_0, L_L, L_R, E)$ consists of three disjoint sets L_0, L_L, and L_R of nodes and a set $E \subseteq L_0 \times L_L \cup L_0 \times L_R$ of edges. Assume that the nodes in L_0, L_L, and L_R appear in the vertical lines $x = 0$, $x = -1$, and $x = 1$, respectively. Similar to the definition of two-layered network, y_0, y_L, y_R can be defined. Notice that the crossing number in a three-layered network is influenced by altering the orderings y_L and y_R or swapping the nodes in L_R with those in L_L. In what follows, we show the concerned labeling problem to be NP-complete. The decision version of the problem can be stated as follows:

The Decision Three-Layered Many-to-One Crossing Problem with $|L_L| = |L_R|$ (D3MCP)
Instance: A three-layered network $G = (L_0, L_L, L_R, E)$ with $|L_L| = |L_R|$ where the mapping from the nodes in L_0 to the nodes in $L_R \cup L_L$ is a many-to-one function, an ordering y_0 of L_0, an integer M.
Question: Can we find the orderings y_L of L_L and y_R of L_R and swap some nodes in L_R with those in L_L, so that the crossing number is no more than M?

Theorem 3 *D3MCP is NP-complete.*

Proof: It is clear that D3MCP is in NP; to prove its NP-hardness, the reduction from the DMCP (mentioned in Theorem 1) is established as follows. Starting with a DMCP instance $G = (L_0, L_1, E)$, y_0, M, we construct a D3MCP instance $G' = (L'_0, L'_L, L'_R, E')$, y'_0, M'. Denote $|L_0|$ by N and $|L_1|$ by n. Since the mapping from L_0 to L_1 is many-to-one, $N \geq n$. Assume $L_0 = \{p_1, p_2, ..., p_N\}$ with $y_0(p_1) > y_0(p_2) > ... > y_0(p_N)$ and $L_1 = \{r_1, r_2, ..., r_n\}$. Let L'_0 be L_0 and L'_R be L_1, so $|L'_R| = n$. Then create n nodes, say $l_1, l_2, ..., l_n$ with $y'_L(l_1) > y'_L(l_2) > ... > y'_L(l_n)$, in L'_L, i.e., $|L'_L| = n$. For each node l_i in L'_L, we add $2\binom{N}{2} + 2$ nodes connected to l_i into L'_0, i.e., $|L'_0| = N + n(2\binom{N}{2} + 2)$, as follows. For each node l_i in $\{l_1, l_2, ..., l_{n-1}\}$, we add $\binom{N}{2} + 1$ nodes between $y'_0(p_{i-1})$ and $y'_0(p_i)$ in L'_0 and $\binom{N}{2} + 1$ nodes between $y'_0(p_i)$ and $y'_0(p_{i+1})$ in L'_0, and then connect those nodes to l_i. For the node l_n, we add $\binom{N}{2} + 1$ nodes between the $y'_0(p_{n-1})$ and $y'_0(p_n)$ in L'_0 and $\binom{N}{2} + 1$ nodes below $y'_0(p_N)$ in L'_0, and then connect those nodes to l_n. The illustration is given in Figure 14. Let $M' = M$.

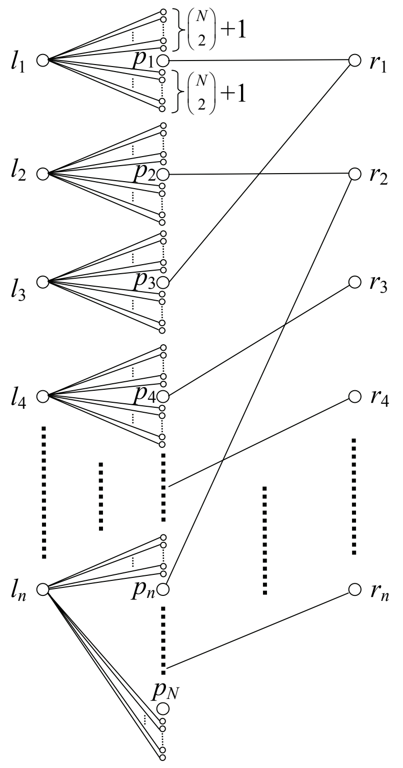


Figure 14: $G'(L'_0, L'_L, L'_R, E')$ with $|L'_0| = N + n(2\binom{N}{2} + 2)$ and $L'_R = L_1$.

In the above construction, for $1 \leq i \leq n$, the node labeled by l_i (resp., r_i) is called a type-l (resp., type-r) node. A node u in L'_R is said to be *swapped* with a node v in L'_L if u and v are placed at $y'_L(v)$ in L'_L and $y'_R(u)$ in L'_R, respectively. Similarly, a node in L'_L swapped to a node in L'_R also can be defined.

In what follows, we prove that $G(L_0, L_1, E)$ has at most M crossings if and only if $G'(L'_0, L'_L, L'_R, E')$ has at most M crossings.

Suppose that G has at most M crossings. From the above construction of G' (see also Figure 14), we observe that crossings in G' only occur on the right-hand side (i.e., between L'_0 and L'_R), so the crossing number of G' is equivalent to that of G. That is, the crossing number of G' is also at most M.

Conversely, suppose that G' has at most M crossings. If $M > \binom{n}{2}$, then the crossing number of G with arbitrary ordering y_0 must be less than M, because it must be no more than $\binom{n}{2}$.

If $M \leq \binom{n}{2}$, it implies that in G' the edges incident to a type-l node cannot cross those incident to any other type-l node as well as those incident to a type-r node; otherwise, it leads to at least $\binom{N}{2}+1 \geq \binom{n}{2}+1$ crossings – impossible. Note that in the converse direction each type-l node in G' may be placed in either L'_L or L'_R. No matter how the type-l nodes in G' are placed, the y-coordinate ordering of the nodes in $\{l_1, l_2, ..., l_n\}$ cannot be modified; otherwise, w.l.o.g., assuming that the y-coordinate of l_i is less than that of l_j for $i < j$, there are at least $\binom{N}{2}+1$ crossings between the edges incident to l_i and either those incident to l_j (if l_i and l_j are placed on the same side) or those incident to r_j (if l_i and l_j are placed on different sides) – impossible. Therefore, w.l.o.g., we assume that the nodes consecutively appear from the topmost to the bottommost in L'_R and L'_L, respectively, are

$$l_1, l_2, ..., l_{i_1}, r_{[i_1+1]}, r_{[i_1+2]}, ..., r_{[i_2]}, l_{i_2+1}, l_{i_2+2}, ..., l_{i_3}, ..., r_{i_{[2k]}}, ..., l_{i_{2k+1}}, ...$$

and

$$r_{[1]}, r_{[2]}, ..., r_{[i_1]}, l_{i_1+1}, l_{i_1+2}, ..., l_{i_2}, r_{[i_2+1]}, r_{[i_2+2]}, ..., r_{[i_3]}, ..., l_{i_{2k}}, ..., r_{i_{[2k+1]}}, ...,$$

where $r_{[1]}, r_{[2]}, ..., r_{[n]}$ is a permutation of $\{r_1, r_2, ..., r_n\}$ (in which $r_{[i]}$ is the i-th node of the permutation) so that l_i and $r_{[i]}$ have the same y-coordinate on different sides in G', for every $i \in \{1, ..., n\}$. Define $R_j = \{r_{[i_{j-1}+1]}, r_{[i_{j-1}+2]}, ..., r_{[i_j-1]}, r_{[i_j]}\}$. Note that the edges incident to every type-r node in R_j only cross those incident to the other type-r nodes in R_j; otherwise, at least $\binom{N}{2}+1$ crossings are generated – impossible. As a result, we can swap all the type-l nodes in L'_R with their same y-coordinate type-r nodes in L'_L, without producing new crossings. Now we obtain a new three-layered network where all the type-l nodes are placed in L'_L, and it has the same crossing number as the original three-layered network G'. That is, the crossing number of the new three-layered network is at most M. After deleting all the type-l nodes of the new three-layered network and their adjacent edges, a two-layered network with at most M edge crossings can be obtained. □

Similar to Algorithm 1, we can draw the bends of leaders in the two track routing areas on two sides so that two edges in G cross if and only if there is one crossing between two leaders (corresponding to the two edges in G) in $\mathcal{M}$. As a result, we have the following corollary.

Corollary 2 *CP2ML-opo is NP-complete even when* $n_1 = n_2$.

4.2 Approximation algorithm with equal number of labels on two sides

In this subsection, we propose an approximation algorithm for the CP2ML-*opo* in the case where the number of labels on the East side is equal to that on the West side, i.e., $n_1 = n_2 = n/2$, n is even.

The approximation algorithm is given in Algorithm 2. Initially in Step 1, all the n labels are placed in the median order on the East side of R, and the leaders are drawn by Algorithm 1. Let L_0 represent such a layout. The idea is to choose $n/2$ labels from the East side and then move them to the West side to minimize the crossing number. When moving a label from the East side to the West side, its y-coordinate is retained and all of its leaders are redrawn in a way that they are reflectionally symmetrical to their original drawings. Suppose we let $\{l_1, ..., l_n\}$ be the set of the n labels, and with respect to a given layout L, $w^L_{i,j}$ be the number of corssings among leaders connecting sites to l_i and l_j. It is not hard to observe the following:

(1) If in layout L, l_i and l_j are on the opposite side of R, then $w^L_{i,j}$=0;

(2) If ogiginally l_i and l_j are on the same side in layout L, and L' is obtained from L by moving l_i and l_j simultaneously to the other side, then $w^{L'}_{i,j} = w^L_{i,j}$.

Next, we construct a complete weighted graph $G = (V, E)$ $(V = \{1, ..., n\})$ in Step 2, where node $i \in V$ corresponds to label l_i, and the weight $w_{i,j}$ of edge (i, j) equals $w^{L_0}_{i,j}$. Suppose we partition V into two disjoint sets S and $V \setminus S$ of equal cardinality, and define

$$\overline{w}(S) := \sum_{i,j \in S} w_{i,j} + \sum_{i,j \in V \setminus S} w_{i,j}.$$

Due to Observations (1) and (2) above, it is easy to see that $\overline{w}(S)$ is exactly equal to the total number of crossings in the layout obtained from the initial layout L_0 by moving all the labels in $V \setminus S$ to the West. As a result, our objective in boundary labeling can be equated with finding a cut so that $\overline{w}(S)$ is minimized – an instance of the so-called Max-Bisection problem.

Even though the Max-Bisection problem is known to have a 1.431-approximation solution [22], the algorithm cannot be applied to our problem directly because the computation of the approximation ratio in the case where $\overline{w}(S)$ is zero is incorrect. As a result, our algorithm needs to check in advance whether there exists a partition such that $\overline{w}(S)$ is zero. In Step 3, we first create a new graph G' that is a duplicate of G, delete the edges with $w_{i,j} = 0$ in G', and then check if the new G' can be decomposed into some bipartite graphs (where the number of the bipartite graphs is denoted by m, $m \geq 1$). One can easily verify that the check can be implemented in polynomial time. Note that any bipartite subgraph of G' (where the weight of any pair of nodes on the same side is zero) can easily be transformed into a boundary labeling without any crossings – just

Algorithm 2 APPROX-CP2ML-*opo*

```
1: Consider a boundary labeling where all the $n$ labels $\{l_1, l_2, ..., l_n\}$ are placed in the median order on the East side of $R$, and the leaders are drawn by Algorithm 1.
2: Construct a complete weighted graph $G = (V, E)$ where each node $i$ in $V$ corresponds to a label $l_i$; the weight $w_{i,j}$ of edge $(i, j)$ equals the number of crossings of the leaders connected to $l_i$ and $l_j$ in the boundary labeling constructed in Step 1.
3: Check if $\overline{w}(S) = \sum_{i,j \in S} w_{i,j} + \sum_{i,j \in V \setminus S} w_{i,j}$ is zero as follows. Let $G' = G$. Delete the edges with $w_{i,j} = 0$ in $G'$. Check if $G'$ can be partitioned into some bipartite graphs (where the number of these bipartite graphs is denoted by $m$, $m \geq 1$). If not (i.e., $\overline{w}(S) \neq 0$), then do Step 4; otherwise, call Procedure 3 to check if the $m$ bipartite graphs can be combined into a new bipartite graph consisting of two disjoint sets with equal cardinality $n/2$. If Procedure 3 returns false (i.e., $\overline{w}(S) \neq 0$), then do Step 4; otherwise (i.e., $\overline{w}(S) = 0$), denote the two disjoint node sets of the combined bipartite graph by $S$ and $V \setminus S$, and then do Step 5.
4: Using $G$ as the input, run the 1.431-approximation algorithm of the Max-Bisection Problem to partition $V$ into two sets $S$ and $V \setminus S$.
5: Output a labeling where the labels corresponding to nodes in $S$ (resp., $V \setminus S$) are placed in the median orders of their adjacent sites on the East (resp., West) side; the leaders are drawn by Algorithm 1.
```

Procedure 3 COMBINE_BIPARTITE(G')

Input: an n-node graph G' consisting of m bipartite graphs.
Output: return true if the m bipartite graphs can be combined into a new bipartite graph consisting of two disjoint sets with equal cardinality $n/2$; else return false.

```
1: We construct an $n \times n \times m$ table $T$. Every entry in table $T$ is initially assigned to zero. Note that each entry only can be 0 or 1.
2: We give the $m$ bipartite graphs in $G'$ an arbitrary ordering. For $j \in \{1, \cdots, m\}$, let the $j$-th bipartite graph be denoted by $A_j \times B_j$ (i.e., $A_j$ and $B_j$ are the two disjoint sets of the $j$-th bipartite graph) and the number of nodes in $A_j$ (resp., $B_j$) be denoted by $n_A^j$ (resp, $n_B^j$). Initially, in table $T$, both entries $T(n_A^1, n_B^1, 1)$ and $T(n_B^1, n_A^1, 1)$ are assigned to 1.
3: For $j = 2, 3, ..., m$, entry $T(p, q, j) = \max\{T(p - n_A^j, q - n_B^j, j - 1), T(p - n_B^j, q - n_A^j, j - 1)\}$ for any $p$, $q \in \{1, 2, ...n\}$. That is, $T(p, q, j) = 1$ iff the first to the $j$-th bipartite graphs can be combined into a new bipartite graph of two disjoint sets with sizes $p$ and $q$, respectively.
4: If entry $T(n/2, n/2, m)$ is 1, then return true; otherwise, return false.
```

placing two disjoint node sets of the bipartite graph on different sides of R and then applying the median heuristic and Algorithm 1 to drawing them.

That is, if G' cannot be decomposed into bipartite graphs, then the crossing number is nonzero, no matter how to place the labels on two sides. Hence, in the case, we can apply the 1.431-approximation algorithm of the Max-Bisection Problem. Otherwise (i.e., G' can be decomposed into some bipartite graphs), we use Procedure 3 (dynamic programming) to combine those bipartite graphs into a new bipartite graph consisting of two disjoint sets with equal cardinality $n/2$. If the output of Procedure 3 is true, then G' can be merged into a bipartite graph consisting of two disjoint sets with equal cardinality $n/2$, so the boundary labeling without any leader crossings can be obtained; otherwise, there does not exist any boundary labeling with zero leader crossing. Subsequently, if the crossing number is nonzero, then Step 4 uses graph G as the input of the 1.431-approximation algorithm of the Max-Bisection Problem to partition V into two sets S and $V \setminus S$. Finally, Step 5 outputs a boundary labeling where the labels corresponding to S (resp., $V \setminus S$) are placed in the median order on the East (resp., West) side, and the leaders are drawn by Algorithm 1.

Unlike heuristic approaches, Algorithm 2 provides an approximation solution with a guaranteed approximation ratio for the intractable problem CP2ML-*opo*. The approximation ratio depends to the map structure as the following result indicates:

Theorem 4 *For the CP2ML-opo in the case where* $n_1 = n_2$*, there exists a* $3(1 + \frac{0.301}{c-1})$*-approximation algorithm, where c is defined as follows:*
if $\sum_{i,j \in V; w_{i,j}>0} 1 \leq \frac{n^2}{4}$*, then* $c = \frac{\sum_{i,j \in V} w_{i,j}}{\sum_{i,j \in V} w_{i,j} - \min\{w_{i,j} | w_{i,j} > 0\}}$.
if $\sum_{i,j \in V; w_{i,j}>0} 1 > \frac{n^2}{4}$*, then* $c = \frac{\sum_{i,j \in V} w_{i,j}}{\sum_{k=1}^{n^2/4} w_k}$
where we sort $\{w_{i,j} | i,j \in V\}$ *from minimum to maximum and rename them as* $\{w_k\}$ *in which* w_k *is the k-th minimum among all.*

Proof: In Algorithm 2, since Step 3 outputs zero if and only if the optimal crossing number is zero, it suffice to show that the output of Step 4 is as required. Let MS_{OPT} denote the optimal solution of the Max-Bisection problem. As the optimal crossing number is nonzero, we discuss the following two cases:

1. If the number of edges with positive weights is less than or equal to the number of edges of a complete bipartite graph consisting of two disjoint sets with equal cardinality $n/2$, i.e., $\sum_{i,j \in V; w_{i,j}>0} 1 \leq (\frac{n}{2})^2$, then MS_{OPT} is not able to partition all the edges with positive weights since the optimal crossing number is nonzero. Therefore,

$$MS_{OPT} \leq \sum_{i,j \in V} w_{i,j} - \min\{w_{i,j} | w_{i,j} > 0\}$$

$$\Longrightarrow \frac{\sum_{i,j \in V} w_{i,j}}{MS_{OPT}} \geq \frac{\sum_{i,j \in V} w_{i,j}}{\sum_{i,j \in V} w_{i,j} - \min\{w_{i,j} | w_{i,j} > 0\}} = c$$

2. In the case where $\sum_{i,j \in V; w_{i,j}>0} 1 > (\frac{n}{2})^2$, MS_{OPT} only can cut at most $(\frac{n}{2})^2$ edges, so

$$MS_{OPT} \leq \sum_{k=1}^{n^2/4} w_k$$
$$\Longrightarrow \frac{\sum_{i,j \in V} w_{i,j}}{MS_{OPT}} \geq \frac{\sum_{i,j \in V} w_{i,j}}{\sum_{k=1}^{n^2/4} w_k} = c$$

Obviously, $c > 1$. Denote the output of Step 4 of Algorithm 2 by c_{APX}, and the minimal crossing number by c_{OPT}. Then

$$\begin{aligned} \frac{c_{APX}}{c_{OPT}} &\leq \frac{\sum_{i,j \in V} w_{i,j} - \frac{MS_{OPT}}{1.431}}{\sum_{i,j \in V} w_{i,j} - MS_{OPT}} \\ &= 1 + \frac{0.301}{\frac{\sum_{i,j \in V} w_{i,j}}{MS_{OPT}} - 1} \\ &\leq 1 + \frac{0.301}{c-1} \end{aligned}$$

Recall that we place all the labels in the median order on the East side in Step 1 of Algorithm 2. Since the median algorithm is a 3-approximation algorithm for one-side many-to-one labeling with *opo*-leaders, we get a $3(1 + \frac{0.301}{c-1})$-approximation algorithm. □

It should be noticed that finding improved approximation algorithms for the CP2ML-*opo* problem remains an interesting open question.

With respect to the Taiwan map with the same sites and labels as in Figure 12, we consider the CP2ML-*opo* in the case where $n_1 = n_2$, i.e., we require that the numbers of labels on both sides are the same. The experimental result of Algorithm 2 is given in Figure 15, which gives us a higher degree of balance in visibility in comparison with Figure 12.

5 The Crossing Problem for One-Side Many-to-One Labeling with Type-*po* Leaders

In this section, we investigate the crossing problem for one-side many-to-one labeling with type-*po* leaders (CP1ML-*po*).

5.1 CP1ML-*po* is NP-complete

Different from Section 3, the x-coordinate ordering of sites plays an important role in the problem discussed in this section, and hence the problem cannot be represented as a two-layered network. The decision version of this problem is stated as follows:

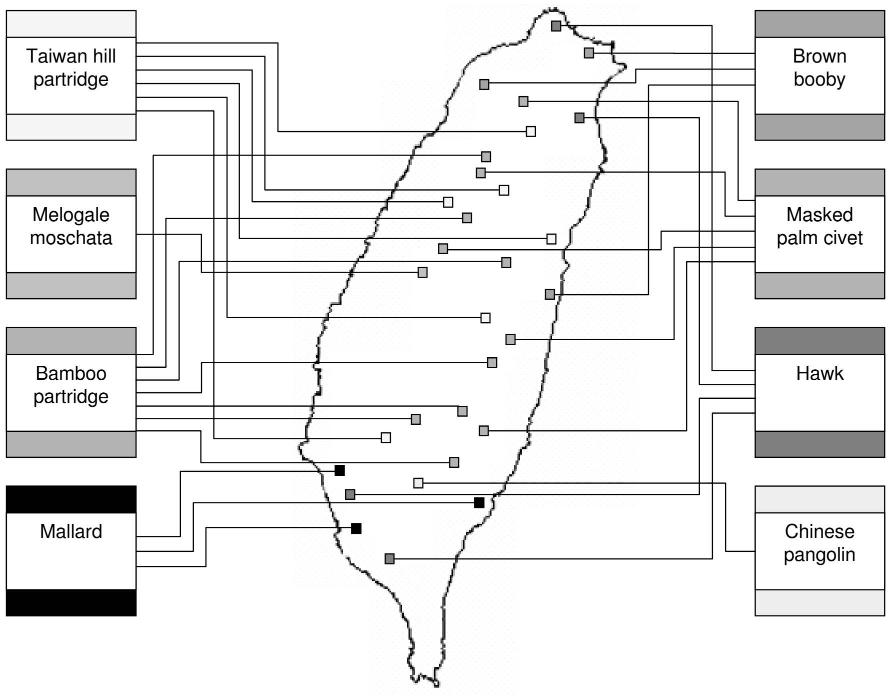


Figure 15: The distribution of some animals in Taiwan, which is represented by two-side many-to-one labeling with type-*opo* leaders.

The Decision Crossing Problem for One-Side Many-to-One Labeling with *po*-Leaders (DCP1ML-*po*)
Instance: A one-side type-*po* map $\mathcal{M} = (P, L, n, 0, 0, 0, f)$, an integer M.
Question: Is there a boundary labeling for $\mathcal{M}$ such that the crossing number is no greater than M?

Theorem 5 *DCP1ML-po is NP-complete.*

Proof: Obviously this problem is in NP because we can guess an ordering of labels and then check if the crossing number is no more than M. Next we show that this problem is NP-hard. We can reduce the DMCP mentioned in Section 3 to a special case of DCP1ML-*po* where the y-coordinate of any site is smaller than that of the lowest port of the lowest label, as illustrated in Figure 16. One can easily see that in order not to induce the crossings of the leaders connected to a common label, the y-coordinate increasing ordering of the ports at which the leaders tough to the common label must respect the x-coordinate decreasing ordering of the sites connected to the label.

In this case, two leaders cross only when the x-coordinate increasing order of any two sites is different from the y-coordinate decreasing order of their corresponding labels (recall that in Section 2 once the position of label is decided, its

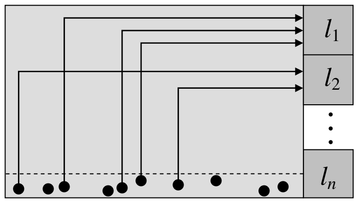


Figure 16: A special case of DCP1ML-*po* where the y-coordinate of any site is smaller than that of the lowest port of the lowest label. Note that some of the edges are not shown in the figure.

port positions also can be decided such that the crossing number is the smallest). Recall that in DMCP, two edges cross only when the y-coordinate increasing order of any two nodes in L_0 is different from the y-coordinate increasing order of their corresponding nodes in L_1. Obviously, the special case of DCP1ML-*po* and DMCP are equivalent as counting the crossing number. Therefore, DCP1ML-*po* is NP-hard. □

5.2 A heuristic

We give a polynomial time heuristic for the labeling problem CP1ML-*po*. Consider a one-side type-*po* map $\mathcal{M} = (P, L, n, 0, 0, 0, f)$, where $L = \{l_1, l_2, ..., l_n\}$ and $f^{-1}(l_i)$ is the set of the sites connected to the common label l_i. Our greedy-based heuristic is described in Algorithm 4. As we are concerned about the reduction of crossings of type-*po* leaders and the placements of labels on the East side of R, it is more natural to start the algorithm from the x-coordinates of sites rahter than from the y-coordinates of sites. Therefore, Steps 1 and 2 preprocess the coordinates of all the sites, and record the leftmost site of those connected to each label.

Recall that we assume the labels along the same side to be of uniform and maximum size; the sizes and possible positions of the n labels on the East side are known. We can image that there are n possible label positions on the East side of the map to which we allocate the n labels, and the y-coordinate ordering of the n labels is sufficient to determine the exact positions of the n labels.

The idea behind our algorithm is that ideally the type-*po* leader between the leftmost site and the topmost or bottommost label leads to fewer crossings with the leaders connected to other labels. Thus, each iteration of Step 3 places the label with the leftmost unconnected sites at the topmost or bottommost unallocated label position on the East side according to whichever situation leads to fewer crossings, while the leaders are drawn by Procedure 5. Note that the leaders connected to a common label do not cross if they are drawn by Procedure 5.

In order to better understand the algorithm, some immediate steps of Algorithm 4 for an example are given in Figure 17. In the leftmost map of Figure 17, Step 2 records the leftmost site in $f^{-1}(l'_i)$ as $p_i^{1'}$ for $i = 1$ to 4; Step 3(a) con-

Algorithm 4 HEURISTIC-CP1ML-$po(G)$

```
1: For $i = 1$ to $n$, compute and record the $x$-coordinate and the $y$-coordinate
   increasing orders of the sites in $f^{-1}(l_i)$, and let $p_i^1$ be the site in $f^{-1}(l_i)$
   with the smallest $x$-coordinate.
2: $\{p_1^1, p_2^1, ..., p_n^1\}$ are sorted according to their $x$-coordinates, and rewritten as
   $\{p_1^{1'}, p_2^{1'}, ..., p_n^{1'}\}$ with $x(p_1^{1'}) < x(p_2^{1'}) < ... < x(p_n^{1'})$. Let $l'_i$ denote the label
   to which $p_i^{1'}$ is connected.
3: Consider that there are $n$ possible label positions on the East side of the
   map, to which we allocate labels $l'_1, l'_2, ..., l'_n$ as follows:
   (a) $l'_1$ is placed at the bottommost label position.
   (b) Assume that the label positions of $l'_1, l'_2, ..., l'_j$ have been determined for
       some $j < n$. Note that in this case there are $(n - j)$ unallocated label
       positions remaining on the East side of the map. For $i = j + 1$ to
       $n$, compute the crossing number when label $l'_{j+1}$ is placed at the top-
       most (resp., bottommost) unallocated label position and the leaders
       connected to label $l'_{j+1}$ are drawn by Procedure 5. Subsequently, la-
       bel $l'_{j+1}$ is placed at the topmost or the bottommost unallocated label
       position according to whichever case leads to fewer crossings.
```

Procedure 5 DRAWING_TYPE-*po*_LEADERS (label l_i)

```
Input: We are given the positions of the ports of label $l_i$ and the sites in $f^{-1}(l_i)$.
Output: The type-po leaders between the sites in $f^{-1}(l_i)$ and the ports of $l_i$
are drawn.
1: See also Figure 18 for an example. Denote the ports of label $l_i$ as
   $\{b_1, b_2, ..., b_m\}$ with $y(b_1) > y(b_2) > ... > y(b_m)$, where $m$ is the number
   of the ports of label $l_i$.
2: Assume that the leaders connected to ports $b_1, b_2, ..., b_j$ have been drawn for
   some $j < m$. For $k = j + 1$ to $m$,
   (a) if there exists at least one unconnected site with $y$-coordinate greater
       than or equal to that of port $b_k$, we connect to port $b_k$ the rightmost
       unconnected site with $y$-coordinate greater than or equal to that of
       port $b_k$;
   (b) otherwise, we connect to port $b_k$ the leftmost unconnected site with
       $y$-coordinate less than that of port $b_k$.
```

nects the sites in $f^{-1}(l'_1)$ to the bottommost label position on the East side of the map. Subsequently, if the sites in $f^{-1}(l'_2)$ are connected to the topmost label position (resp., the second label position to the bottom) and their adjacent leaders are drawn by Procedure 5, then they induce no (resp., one) crossing. Therefore, as shown in the second map of Figure 17, label l'_2 is placed at the topmost label position, and their adjacent leaders are drawn by Procedure 5. Similarly, labels l'_3 and l'_4 are placed, and their adjacent leaders are drawn, as shown in the rightmost map of Figure 17.

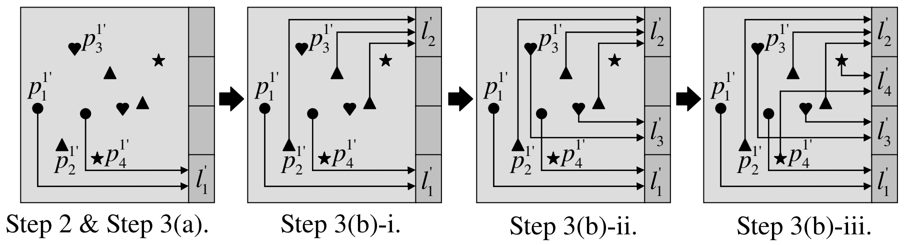


Figure 17: An example of executing Algorithm 4, where the sites connected to a common label are represented by the same icon.

In order to understand how to use Procedure 5 to draw the type-*po* leaders between the sites and the ports with fixed positions, we give an example shown in Figure 18. In the first iteration of Step 2 of Procedure 5, $y(a_1) \geq y(b_1)$ and $y(a_2) \geq y(b_1)$, but we have $x(a_1) > x(a_2)$, so that we connect a_1 to b_1 (Step 2(a) of Procedure 5). By using the same discussion, the leaders of $b_2 - b_5$ are drawn. Subsequently, while b_6 is concerned, since there is no unconnected site a_i for any i so that $y(a_i) \geq y(b_6)$, we connect to port b_6 the leftmost unconnected site a_6 (Step 2(b) of Procedure 5). As for b_7, $y(a_7) \geq y(b_7)$ and $y(a_8) \geq y(b_7)$, but we have $x(a_7) > x(a_8)$, so that we connect a_7 to b_7. Similarly, the leaders of b_8 and b_9 can be drawn.

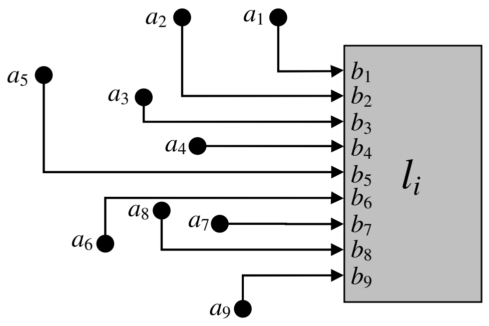


Figure 18: An example of illustrating the type-*po* leaders drawn by Procedure 5.

One can easily verify the correctness of Procedure 5 by considering the following simple invariant: after port b_k is connected in Step 2 of Procedure 5, it always guarantees that the later leaders connected to ports in $\{b_{k+1}, b_{k+2}, ..., b_m\}$ do not cross the existing leaders connected to ports in $\{b_1, b_2, ..., b_k\}$.

Recall that N is the number of sites and $N \geq n$. Because the x-coordinate and the y-coordinate orders of the sites in $f^{-1}(l_i)$ are obtained in Step 1 of Algorithm 4, each iteration of Step 2 in Procedure 5 can be implemented in constant time, and hence, Procedure 5 runs in time $O(N)$. If the coordinate orders of sites are not preprocessed, Procedure 5 runs in time $O(N \log N)$. Note that Procedure 5 (of which objective is to find the one-to-one labeling with no leader crossings) is different from the $O(N^2)$-time algorithm of [3, 4] used for finding the one-to-one labeling with no leader crossings as well as the minimal total leader length. Therefore, Algorithm 4 runs in time $O(N^2)$, because Steps 1 and 2 (resp., Steps 3 computing the crossing number) can be implemented in time $O(N \log N)$ (resp., $O(N^2)$).

In what follows, we implement the algorithm and give some experimental results. The implementation is in C language, and runs on a Pentium M 1.5G Hz PC with 768 MB memory. The experimental results of three problem sets are given in Figure 19, where the top (resp., middle; bottom) bar chart shows the minimal crossing numbers and the crossing numbers of our outputs for the problems with fixed eight (resp., nine; ten) labels and random numbers of sites. Note that the coordinates of the input sites for each problem are different because they are generated randomly. The running times associated with the experiments in Figure 19 are given in Table 3, where each problem is solved in less than one second. Comparing the minimal crossing numbers with the outputs of our algorithm in Figure 19, our algorithm yields layouts of reasonably good quality. It is of interest to further investigate other experimental settings and the performances on practical problems.

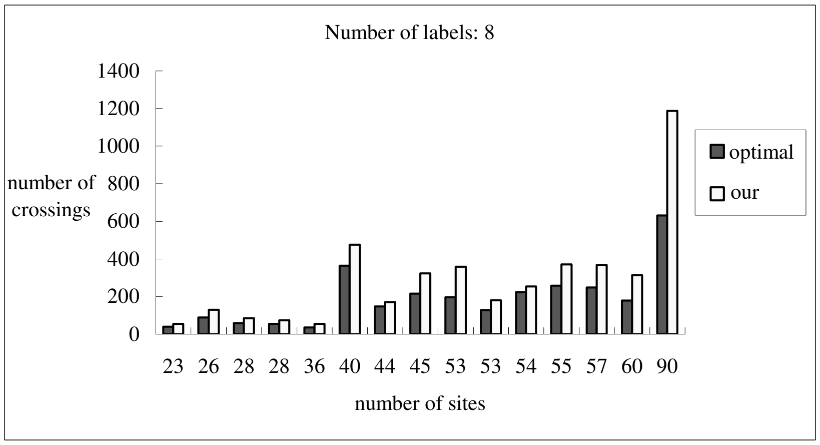
6 The Crossing Problem for Two-Side Many-to-One Labeling with Type-*po* Leaders

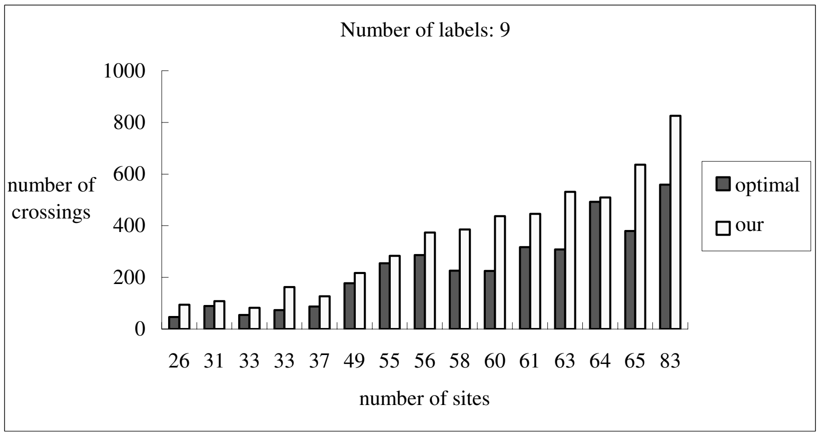
Again, from the result in the previous section, CP2ML-*po* is NP-complete (CP1ML-*po* is a special case of CP2ML-*po* with $n_2 = 0$). In this section, we investigate CP2ML-*po* under the restriction that $n_1 = n_2$. We first show the crossing problem to be NP-complete, then a heuristic is given.

6.1 CP2ML-*po* is NP-complete even when $n_1 = n_2$

Similar to Section 5, the decision version of DCP2ML-*po* is stated as follows:

The Decision Crossing Problem for Two-Side Many-to-One Labeling with *po*-Leaders (DCP2ML-*po*)
Instance: A two-side type-*po* map $\mathcal{M} = (P, L, n_1, n_2, 0, 0, f)$, an integer M.
Question: Is there a boundary labeling for $\mathcal{M}$ such that the crossing number is no greater than M?





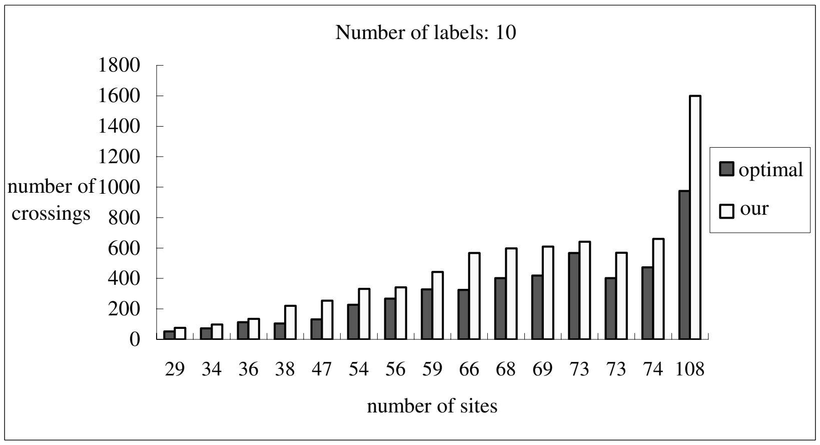


Figure 19: The experimental results for DCP1ML-*po*.

Table 3: Running time for Figure 19.

number of labels = 8		number of labels = 9		number of labels = 10	
number of sites	running time (millisecond)	number of sites	running time (millisecond)	number of sites	running time (millisecond)
23	20.3	26	31.3	29	39.0
26	26.6	31	43.8	34	50.0
28	34.4	33	46.9	36	53.5
28	34.4	33	46.9	38	60.9
36	50.5	37	62.4	47	92.6
40	67.2	49	119.8	54	137.5
44	87.5	55	146.9	56	142.1
45	89.1	56	155.6	59	162.5
53	120.7	58	158.5	66	212.5
53	120.7	60	164.0	68	225.0
54	124.5	61	165.6	69	234.3
55	126.6	63	170.7	73	238.6
57	148.4	64	182.8	73	238.6
60	171.5	65	195.3	74	256.2
90	367.2	83	309.5	108	553.8

Theorem 6 *DCP2ML-po is NP-complete even when $n_1 = n_2$.*

Proof: Obviously the problem is in NP because we can guess an ordering of labels and then check if the crossing number is smaller than M. Next we show that this problem is NP-hard. This proof is similar to the one of DCP1ML-*po*. We can reduce the DMCP mentioned in Section 3 to a special case of DCP2ML-*po* where the y-coordinate of any site is smaller than that of the lowest port among all label ports, as illustrated in Figure 20. In order not to induce any crossing among the leaders connected to a common label, one should notice that the x-coordinate increasing order of sites should respect the y-coordinate decreasing (resp., increasing) order of ports while the label is placed on the East (resp., West) side, e.g., see the leaders connected to label l_{n_2+1} (resp., l_{n_2}) in Figure 20. Therefore, we reasonably assume that the leaders connected to a common label never cross. Also recall that leaders never overlap because overlapping can be easily removed by adjusting the positions of ports slightly (e.g., see the leaders connected to labels l_{n_2} and l_{n_2+1} in Figure 20).

We define the *circular ordering* of labels, which sorts all labels according to their y-coordinates from bottom to up on the West side of map and then from up to bottom on the East side of map. The details are given as follows. As shown in Figure 20, we assign l_1 to the lowest label on the West side, l_2 to the second lowest label on the West side, and so on until we assign l_{n_2} to the topmost label on the West side. Next we assign l_{n_2+1} to the topmost label on the East side, l_{n_2+2} to the second topmost label on the East side, and so on until we assign $l_{n_2+n_1}$ to the lowest label on the East side.

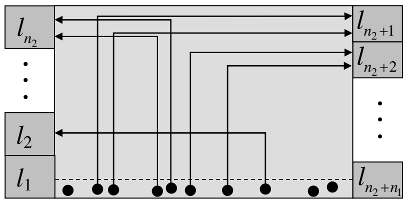


Figure 20: A special case of DCP2ML-*po* where the y-coordinate of any site is smaller than that of the lowest port among the ports of all labels. Note that some of the edges are not shown in the figure.

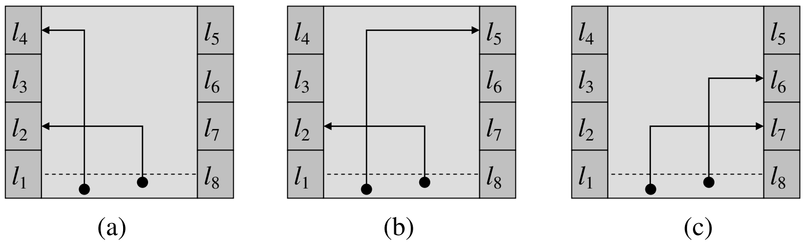


Figure 21: A crossing is produced when the x-coordinate increasing order of two sites does not respect the circular ordering of their corresponding labels, in the case where the two sites are connected to (a) the West side, (b) different sides, and (c) the East side.

In this case, we observe that two leaders cross only when the x-coordinate increasing order of any two sites is different from the circular ordering of their corresponding labels, as shown in Figure 21. Recall that in DMCP, two edges cross only when the y-coordinate increasing order of any two nodes in L_0 is different from that of their corresponding nodes in L_1. Obviously, the special case of DCP2ML-*po* and DMCP are equivalent as counting the crossing number. Therefore, DCP2ML-*po* is NP-hard even when $n_1 = n_2$. □

6.2 A heuristic

In this subsection, we devise a polynomial time heuristic for the labeling problem CP2ML-*po*. Our heuristic needs the algorithm for solving the Min-Bisection Problem for directed graphs, which is not approximable [10]. To our knowledge, so far there has not been any heuristic for solving the Min-Bisection Problem for directed graphs. Our heuristic is based on the K-L heuristic [15], which was designed for solving the Min-Bisection Problem for undirected graphs in practice. Consider a two-side type-*po* map $\mathcal{M} = (P, L, n_1, n_2, 0, 0, f)$, where $L = \{l_1, l_2, ..., l_n\}$ and $f^{-1}(l_i)$ is the set of the sites connected to the common label l_i. Our heuristic is described in Algorithm 6. In order to better understand how Algorithm 6 works, see also an example shown in Figure 22.

Algorithm 6 is explained as follows. In Step 1, we construct a weighted directed graph $G = (V, E)$, where each node in V represents a label in L and each arc (l_i, l_j) has a weight which is defined as follows:

$$w_{i,j} = \sum_{x(a)>x(b),a\in f^{-1}(l_i),b\in f^{-1}(l_j)} 1.$$

One should notice that $w_{i,j}$ is different from $w_{j,i}$. The intuition is that $w_{i,j}$ is penalized by one unit if label l_i (resp., l_j) is placed on the West (resp., East) side but site $a \in f^{-1}(l_i)$ (resp., $b \in f^{-1}(l_j)$) is located close to the East (resp., West) side. That is, $w_{i,j}$ can roughly estimate the number of the crossings induced by placing l_i and l_j on the West and the East sides, respectively.

Subsequently, Steps 2 applies the K-L heuristic in [15] to partitioning graph G into two subsets A and $B = V \setminus A$ with equal cardinality so that $w(A, B) = \sum_{(l_i,l_j)\in A\times B} w_{i,j}$ is as small as possible. The objective of Step 2 is to partition all the labels into two groups (A and B) with equal size so that the crossing number ($w(A, B)$) is as small as possible when the two groups (A and B) of sites are placed on the West and the East sides of the map, respectively. Finally, Step 3 applies Algorithm 4 to determining the positions of the labels corresponding to the nodes in A and B on the West and the East sides of the map, respectively, and drawing all the type-*po* leaders.

Algorithm 6 HEURISTIC-CP2ML-*po*(G)

```
1: Construct a weighted directed graph $G = (V, E)$, where every node in $V$ corresponds to a label and each arc $(l_i, l_j)$ has a weight $w_{i,j} = \sum_{x(a)>x(b),a\in f^{-1}(l_i),b\in f^{-1}(l_j)} 1$. Note that $w_{i,j} \neq w_{j,i}$.
2: We apply the K-L heuristic of [15] to partitioning graph $G$ into two subsets $A$ and $B = V \setminus A$ with equal sizes so that $w(A, B) = \sum_{(l_i,l_j)\in A\times B} w_{i,j}$ is as small as possible.
3: Apply Algorithm 4 to determining the placements of the labels corresponding to the nodes in $A$ and $B$ on the West and the East sides of the map, respectively, as well as the drawings of the type-po leaders.
```

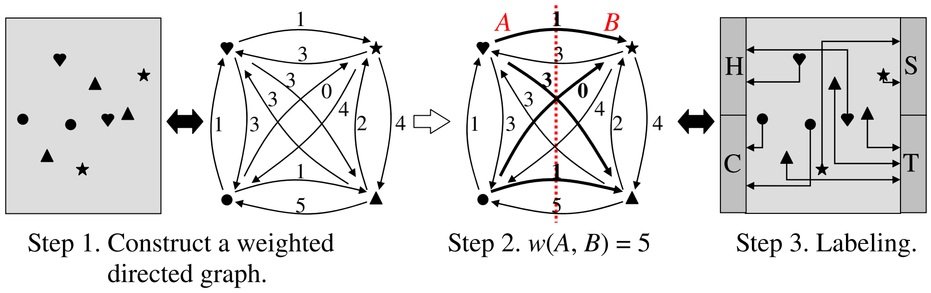


Figure 22: An example for illustrating how Algorithm 6 is executed.

As for the time complexity of Algorithm 6, Step 1 can be implemented in time $O(N^2)$ in the following way. Initially, an $N \times N$ table is used to record all the $w_{i,j}$ values. We iterate each site p_a, of which label is called l_i, so the total number of iterations is N. In each iteration, we consider each of the sites rather than p_a, which is denoted by p_b and its corresponding label is called l_j. If $x(p_a) > x(p_b)$, then $w(i,j) := w(i,j) + 1$. If $x(p_b) > x(p_a)$, then $w(j,i) := w(j,i) + 1$. Therefore, Step 1 can establish the weighted directed graph in time $O(N^2)$.

In addition, Step 2 runs in time $O(N^2 \log N)$ [15], and Step 3 (Algorithm 4) runs in time $O(N^2)$. As a result, Algorithm 6 runs in time $O(N^2 \log N)$.

In what follows, we implement the algorithm and give some experimental results to compare the minimal crossing number and the output of Algorithm 6. The experimental setting is the same as that used in Subsection 5.2. The experimental results of two problem sets are given in Figure 23, where the top (resp., bottom) bar chart shows the minimal crossing numbers and the crossing numbers of the outputs of our algorithm for the problems with fixed ten (resp., eleven) labels and random numbers of sites. In view of the figure, our algorithm outputs a layout of good quality.

7 Discussions and Conclusions

In this section, we discuss the problem with the objective of minimizing the total leader length in Subsection 7.1, as well as the problems for type-s leaders in Subsection 7.2. Some conclusions with future work are given in Subsection 7.3.

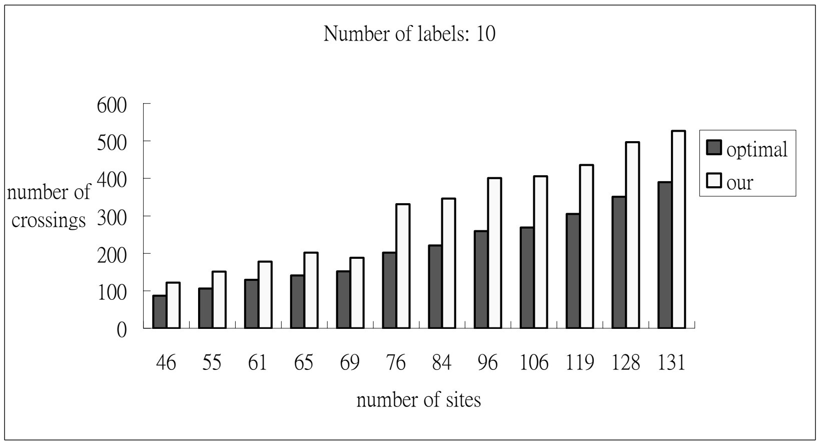
7.1 The leader length problem

In our earlier discussion, we have investigated the many-to-one boundary labeling with the objective of minimizing the crossing number, regardless of the total leader length. An alternative objective might be to minimize the total leader length, regardless of the total crossing number. The concerned problem can be stated as follows:

The Leader Length Problem for Many-to-One Labeling (LLPML): Given a k-side type-t map $\mathcal{M} = (P, L, n_1, n_2, n_3, n_4, f)$, where $t \in \{po, opo\}$, find a boundary labeling for $\mathcal{M}$ such that the total leader length is as small as possible, regardless of the total crossing number.

In what follows, we show LLPML to be solvable in polynomial time using an approach originally used in [1]. The algorithm works regardless of the leader type (*opo* or *po*) and the boundary type (one-side, two-side, or four-side).

As crossings among leaders are allowed, leaders can be routed arbitrarily. Consider a representative label l_i for some $i \in \{1, \cdots, n\}$. It is easy to see that if the position of labels l_i is determined, then routing the leaders connected to labels l_i never affects the length of any other leader. It implies that in the LLPML solution the sum of lengths of the leaders connected to label l_i must be



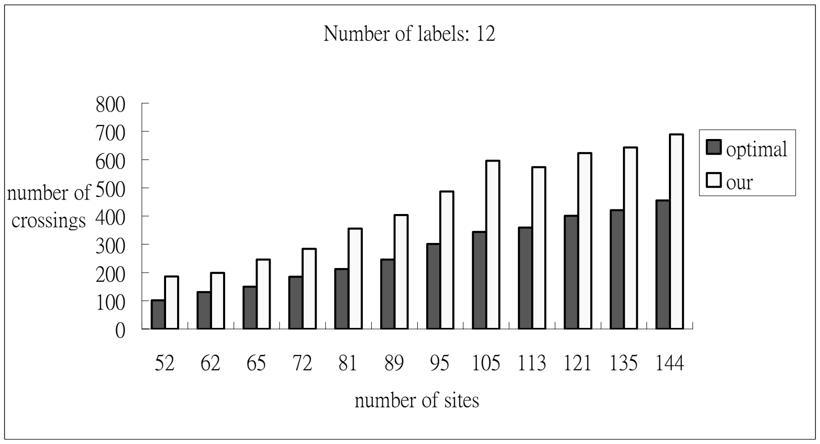


Figure 23: The experimental results for DCP2ML-*po*.

the minimal. Therefore, we first consider how to route the leaders connected to label l_i such that the sum of those leaders is the minimal.

Note that the positions of ports of label l_i are determined as the position of label l_i is determined. The shortest length of a type-*opo* or type-*po* leader between a site u and a port p of label l_i can be measured by *Manhattan distance*, i.e., the shortest distance between site u and port p which is measured along axes at right angles. For example, in a plane with site u at (x_1, y_1) and port p at (x_2, y_2), the leader length is $|x_1 - x_2| + |y_1 - y_2|$.

Let N_i denote the number of sites in $f^{-1}(l_i)$, for $i \in \{1, \cdots, n\}$. Hence, $N_1 + \cdots + N_n = N$. It is easy to observe that the leader length minimization problem of routing the type-*opo* (resp., type-*po*) leaders connected to label l can be viewed as the leader length minimization problem for one-side one-to-one boundary labeling (where each port of label l_i is viewed a label in the one-to-one boundary labeling), and hence can be solved in time $O(N_i \log N_i)$ (resp., $O(N_i^2)$) [4]. Note that the leaders connected to label l_i never cross one another when the sum of their lengths is the minimal [4].

Subsequently, we consider how to determine the position of each label. Since the number of labels placed on each side is given, the sizes of labels on each side are known (fixed). Hence, there are n possible fixed label positions around the map, in which each position just accommodates one of the n labels. In order to allocate the n labels to the n possible label positions around the map, we construct a weighted bipartite graph $A \times B$ in which

- each node a_i, $i \in \{1, \cdots, n\}$, in A represents label l_i;
- each node b_i, $i \in \{1, \cdots, n\}$, in B represents a possible label position around the map;
- for any $i, j \in \{1, \cdots, n\}$, the weight of edge $a_i b_j$ is the sum of lengths of the leaders connected to l_i in the case where label l_i is placed at the position represented by b_j.

In the case of type-*opo* (resp., type-*po*) leaders, the weighted bipartite graph can be constructed in time $O(nN \log N)$ (resp., $O(nN^2)$). In what follows, we explain the case of type-*opo* leaders; the other case is similar. Recall that the sum of lengths of the leaders connected to label l_i can be computed in time $O(N_i \log N_i)$. Since label l_i has n possible label positions, the weights of the n edges in $A \times B$ connected to node a_i in A can be computed in time $O(nN_i \log N_i)$. Hence, the weights of all the edges in $A \times B$ can be computed in time $O(n \times (N_1 \log N_1 + \cdots + N_n \log N_n)) \leq O(n \times (N_1 \log N + \cdots + N_n \log N)) = O(n \times (N_1 + \cdots + N_n) \log N) = O(nN \log N)$.

Finally, like the work in [1], the LLPML solution can be obtained by finding the minimum weighted matching in $A \times B$, which can be solved in time $O(n^2 \log^3 n)$ [18]. As a consequence, LLPML can be solved in polynomial time.

It is interesting to compare and contrast the results for LLPML under type-*po* leaders with those listed in Table 2, which deals with one-to-one boundary labeling. LLPML for all cases are tractable; however, only the one-side and two-side one-to-one boundary labeling problems (i.e., excluding the four-side one) are tractable. The disparity comes from the fact that unlike one-to-one boundary labeling which requires crossing-free leaders, LLPML allows crossings among leaders.

7.2 The problems for type-s leaders

About the problems for type-s leaders, one may guess that our NP-hardness proofs of the problems for type-*opo* leaders (which are reduced from two- or three- layered networks) might be used to show the intractable problems for type-s leaders. However, it is impossible because in the case of type-s leaders both the x- and y- coordinates of the sites matter, but in the case of type-*opo* leaders only the y-coordinates of the sites matter.

7.3 Conclusions with future work

We have shown the crossing problems for both one-side and two-side many-to-one labeling with type-*opo* leaders as well as type-*po* leaders to be NP-complete. For such intractable problems, we have also given approximation algorithms or heuristics with satisfactory performances. In the future, we plan to investigate the crossing problems for four-side many-to-one labeling with either type-*opo* or type-*po* leaders. It is also interesting to investigate the case where sites are of more complicated shape (such as lines, rectangles) and size, which tend to cause a large number of crossings.

Acknowledgements

The authors thank the anonymous referees for comments that improved the content as well as the presentation of this paper.

References

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

[2] M. Bekos, M. Kaufmann, K. Potika, and A. Symvonis. Polygons labelling of minimum leader length. In *Proceedings of Asia Pacific Symposium on Information Visualisation 2006 (APVIS2006)*, volume 60 of *CRPIT*, pages 15–21, 2006.

[3] M. Bekos, M. Kaufmann, A. Symvonis, and A. Wolff. Boundary labeling: models and efficient algorithms for rectangular maps. In *Proceedings 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. Bekos and A. Symvonis. BLer: A boundary labeller for technical drawings. In *Proceedings of the 13th International Symposium on Graph Drawing (GD 2005)*, volume 3843 of *LNCS*, pages 503–504, 2006.

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

[7] 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.

[8] P. Eades and N. C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11:379–403, 1994.

[9] U. Feige and R. Krauthgamer. A polylogarithmic approximation of the minimum bisection. *SIAM Review*, 48(1):99–130, 2006.

[10] U. Feige and O. Yahalom. On the complexity of finding balanced oneway cuts. *Information Processing Letter*, 87(1):1–5, 2003.

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

[12] M. R. Garey and D. S. Johnson. *Computers and Interactability. A Guide to the Theory of NP-Completeness.* A Series of Books in the Mathematical Sciences. Freemann And Company, 1979.

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

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

[15] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2):291–307, 1970.

[16] X. Munoz, W. Unger, and I. Vrt'o. One sided crossing minimization is NP-hard for sparse graphs. In *Proceedings of the 9th International Symposium on Graph Drawing (GD 2001)*, volume 2265 of *LNCS*, pages 115–123, 2002.

[17] 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.

[18] P. M. Vaidya. Geometry helps in matching. *SIAM Journal on Computing*, 18(6):1201–1225, 1989.

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

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

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

[22] Y. Ye. A .699-approximation algorithm for max-bisection. *Mathematical Programming*, 90(1):101–111, 2001.