Boundary Labeling for Rectangular Diagrams

Prosenjit Bose
School of Computer Science, Carleton University, Ottawa, Canada
jit@scs.carleton.ca

Paz Carmi
Department of Computer Science, Ben-Gurion University, Beer-Sheva, Israel
carmip@cs.bgu.ac.il

J. Mark Keil
Department of Computer Science, University of Saskatchewan, Saskatoon, Canada
mark.keil@usask.ca

Saeed Mehrabi
School of Computer Science, Carleton University, Ottawa, Canada
saeed.mehrabi@carleton.ca

Debajyoti Mondal
Department of Computer Science, University of Saskatchewan, Saskatoon, Canada
d.mondal@usask.ca

Abstract

Given a set of n points (sites) inside a rectangle R and n points (label locations or ports) on its boundary, a boundary labeling problem seeks ways of connecting every site to a distinct port while achieving different labeling aesthetics. We examine the scenario when the connecting lines (leaders) are drawn as axis-aligned polylines with few bends, every leader lies strictly inside R, no two leaders cross, and the sum of the lengths of all the leaders is minimized. In a k-sided boundary labeling problem, where $1 \leq k \leq 4$, the label locations are located on the k consecutive sides of R.

In this paper we develop an $O(n^3 \log n)$-time algorithm for 2-sided boundary labeling, where the leaders are restricted to have one bend. This improves the previously best known $O(n^8 \log n)$-time algorithm of Kindermann et al. (Algorithmica, 76(1):225–258, 2016). We show the problem is polynomial-time solvable in more general settings such as when the ports are located on more than two sides of R, in the presence of obstacles, and even when the objective is to minimize the total number of bends. Our results improve the previous algorithms on boundary labeling with obstacles, as well as provide the first polynomial-time algorithms for minimizing the total leader length and number of bends for 3- and 4-sided boundary labeling. These results settle a number of open questions on the boundary labeling problems (Wolff, Handbook of Graph Drawing, Chapter 23, Table 23.1, 2014).

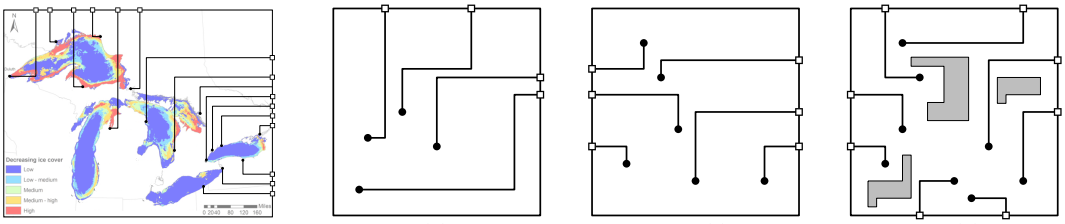
2012 ACM Subject Classification Theory of computation, Theory of computation → Algorithm design techniques, Theory of computation → Computational geometry

Keywords and phrases Boundary labeling, Dynamic programming, Outerstring graphs

Digital Object Identifier 10.4230/LIPIcs.SWAT.2018.12

Related Version See [8], `https://arxiv.org/abs/1803.10812` for the full version of the paper.

Funding Research of Prosenjit Bose and Saeed Mehrabi is supported in part by Natural Sciences and Engineering Research Council of Canada (NSERC). Saeed Mehrabi is also supported by a Carleton-Fields postdoctoral fellowship. Debajyoti Mondal is supported in part by Global Water Futures project (GWF) and Natural Sciences and Engineering Research Council of Canada (NSERC).



(a) (b) (c) (d)

Figure 1 (a) A 1-bend 2-sided boundary labeling (i.e., with *po*-leaders) on a geographic map showing (ice cover on the Great Lakes [14]). (b) A 2-bend 2-sided boundary labeling (i.e., with *opo*-leaders). This example does not have a feasible solution with 1-bend leaders. (c) Boundary labeling in 1-bend opposite 2-sided model. (d) A 1-bend 4-sided boundary labeling in the presence of obstacles.

1 Introduction

Labeling problems appear in a variety of scenarios such as in annotating educational diagrams, wiring schematics, system manuals, as well as in many information visualization and engineering applications. The increasing trend of automation in these areas has motivated the research in labeling algorithms. Crossings among the *leaders* (i.e., the lines connecting labels to the sites), number of bends per leader, and the sum of leader lengths are some important aesthetics of a diagram labeling. To achieve clarity and better readability, all these parameters are often preferred to be kept small.

Many labeling problems are NP-hard [12, 5]. A rich body of research attempts to develop efficient approximation and heuristic algorithms [13, 15, 10, 21, 22], both in the static and the dynamic settings [3, 10]. In this paper we examine a well-known variant of the labeling problem called *b-bend k-sided boundary labeling*, e.g., see Figure 1. The input for this problem is a set of kn sites and kn ports, where the sites lie in the interior of a rectangle R, the ports are located on k consecutive sides of R, and each side contains n ports. Both the sites and ports are represented as points. The goal is to decide whether each site can be connected to a unique port using axis-aligned leaders such that the leaders are disjoint, each leader lies strictly inside R and each leader has at most b bends. If such a labeling exists, then we compute a labeling that optimizes these labeling aesthetics. We examine two such optimization criteria: one is to minimize the sum of the leader lengths, and the other is to minimize the total number of bends.

The strict-containment inside R, bend restrictions and orthogonal constraints impose certain shapes on the leader. An orthogonal leader containing exactly one bend (resp., two bends) is known as a *po-leader* (resp., an *opo-leader*)[1] [17]. We note that there are 1-bend leaders with 135° degrees at the bend, which are knowns as *do-leaders* [2]. Since we are only interested in orthogonal leaders in this paper, we say 1-bend leaders to always mean "*po*-leaders" for the rest of the paper.

Related work. Boundary labeling has been an active area of research in the last decade, e.g., see the surveys [1, 20]. The boundary labeling problem was first introduced by Bekos et al. [6]. They gave $O(n \log n)$-time algorithms to decide labeling feasibility for the 1-bend

[1] The letters 'o' and 'p' stand for 'orthogonal' and 'parallel', respectively. So, an *opo*-leader starts orthogonally at the site, and ends orthogonally at the port.

1-sided and opposite 2-sided models, i.e., the labels are located on two opposite sides of R. In addition, they gave an $O(n^2)$-time algorithm that minimizes the total leader length. For the 2-bend 4-sided model, they could test the feasibility in $O(n \log n)$ time and reduced the length minimization to a minimum-cost bipartite matching problem. Benkert et al. [7] improved Bekos et al.'s [6] result on the 1-bend 1-sided model by devising an $O(n \log n)$-time algorithm for the length minimization. They also considered general cost functions (i.e., beyond Euclidean length), as well as other types of leaders. We refer the reader to [19, 4] for other variants of boundary labeling problem.

The 2-sided model considered by Bekos et al. [6] and Benkert et al. [7] is an opposite-sided model, i.e., ports are placed on two opposite sides of R. This model is different from the adjacent 2-sided model, where the labels are always placed on adjacent sides. The adjacent 2-sided model was first considered by Kindermann et al. [17]. For the 1-bend 2-sided model, they gave an $O(n^2)$-time algorithm to check feasibility, and an $O(n^8 \log n)$-time algorithm for total leader length minimization; to our knowledge, this is the fastest algorithm known for the 1-bend 2-sided model. Note that the labeling problem in this model seems surprisingly more difficult than the corresponding opposite 2-sided model (also mentioned by Kindermann et al. [17]). For the 1-bend 3-sided (resp., 4-sided) model, they gave an $O(n^4)$-time (resp., $O(n^9)$-time) algorithm for checking the labeling feasibility, but they were unable to solve the length minimization problem. They posed this as an open question, i.e., can a minimum-length solution for the 3- and 4-sided boundary labeling be computed in polynomial time? These challenges motivated us to examine the adjacent model in more detail.

Fink and Suri [11] studied the boundary labeling problem in the presence of *obstacles*. In addition to the set of sites, they allowed a set of orthogonal polygons (equivalently, obstacles) to lie inside R. The objective is to minimize the total leader length with the constraint that the leaders must not intersect the obstacles. They gave polynomial-time algorithms for minimizing the total leader length in the 1-sided and opposite 2-sided models, but the running time of these algorithms while using *po*- and *opo*-leaders is fairly high, i.e., $O(n^4)$, $O(n^8)$ for the 1-sided model, and $O(n^9)$, $O(n^{21})$ for the opposite 2-sided model. They also examined the case when the leaders have non-uniform lengths and the leader locations can be chosen, which they proved to be NP-hard.

A different generalization of boundary labeling considers sliding ports, i.e., labels are assigned disjoint intervals on the boundary of R, and a site can be connected to any point in such an interval. In the 1-sided model, Bekos et al. [6] gave an $O(n^2)$-time algorithm that can minimize the total number of bends using *opo*-leader (they did not require the *opo*-leaders to lie strictly inside R). They posed an open question to determine the time complexity for the 3- and 4-sided case. Benkert et al. [7] considered bend minimization with *po*-leaders. They gave an $O(n^2)$-time algorithm for the 1-sided model, and $O(n^8)$-time algorithm for the opposite 2-sided model. The 'Handbook of Graph Drawing' [20] lists a number of open problems related to the minimization of the total number of bends for different variants of boundary labeling.

The 1-, 3- and 4-sided models for the boundary labeling problem are always adjacent models, but a 2-sided model can be either adjacent or opposite. Throughout the paper we will refer to the 'opposite' variant as an 'opposite 2-sided' model.

Our contributions. We give an algorithm for the 1-bend 2-sided boundary labeling problem that minimizes the total leader length in $O(n^3 \log n)$ time (if such a labeling exists). Ours is an adjacent model and uses *po*-leaders, and hence improves the $O(n^8 \log n)$-time algorithm of Kindermann et al. [17]. Since the best known algorithm for the length minimization in the

1-bend opposite 2-sided model takes $O(n^2)$ time [7], our result raises an intriguing question that whether the adjacent boundary labeling model can further be improved to reach (or, even break) the $O(n^2)$ barrier.

We show that many variants of the boundary labeling problems can be related to outerstring graphs, where the minimization of total leader lengths or bends reduces to an optimization problem in those outerstring graphs. We notice that this relation is previously pointed out in a different context [18]. This idea leads us to the following results:

- The first polynomial-time algorithm with a running time of $O(n^6)$ for the 1-bend 3-sided and 4-sided boundary labeling problem that minimize the total leader length. This settles the time-complexity question posed by Kindermann et al. [17].
- Polynomial-time algorithms for minimizing the total leader length or the total number of bends, even in the presence of obstacles. Our algorithms work for both *po*- and *opo*-leaders, as well as for all possible distributions of the ports to the boundary of R, i.e., both adjacent and opposite models. The running time for the opposite 2-sided model is $O(n^6)$ for *po*-leaders and $O(n^9)$ for *opo*-leaders; these improve, respectively, the $O(n^9)$- and $O(n^{21})$-time algorithms of Fink and Suri [11]. This technique can also be applied to the sliding port model, which settles the time-complexity question posed in [6, 20] related to the bend minimization.

2 Computing 1-Bend 2-Sided Boundary Labelings

In this section we give an $O(n^3 \log n)$-time algorithm to find a solution to the 1-bend 2-sided boundary labeling problem. Throughout this section, we assume that the sites and ports are in general position, i.e., no axis-aligned straight line passing through a site intersects a port or another site. Consequently, each leader must have exactly one bend. We thus omit the term '1-bend' in the rest of this section. Moreover, we assume that the ports lie on the top and right sides of the rectangle R.

2.1 Technical Background

Let $R(t), R(b), R(l), R(r)$ be the *top, bottom, left and right sides* of R, respectively. An *xy-separating curve* is an axis-aligned xy-monotone polygonal chain that starts at the bottom-left corner of R and ends at the top-right corner of R. A 2-sided boundary labeling solution is *xy-separated* if there exists an xy-separating curve such that the leaders incident to $R(t)$ (resp., $R(r)$) lie on or above (resp., below) the xy-separating curve.

▶ **Lemma 1** (Kindermann et al. [17]). *If a 2-sided boundary labeling problem has an affirmative solution with 1-bend leaders, then there exists such an xy-separated solution that minimizes the sum of all leader lengths.*

Figure 2(a) illustrates an xy-separated solution of a 2-sided boundary labeling problem. An xy-separated curve is shown in a light-green. Let $\mathcal{I}$ be an instance of a 2-sided boundary labeling problem. Without loss of generality assume that the ports are distributed along the sides $R(t)$ and $R(r)$. Let $ports(R(t))$ (resp., $ports(R(r))$) be the set of ports along $R(t)$ (resp., $R(r)$). A leader is called *inward* if the 90° angle formed at its bend point contains the top-right corner of R. Otherwise, we call the leader an *outward* leader. The leaders incident to ℓ and ℓ' in Figure 2(a), are inward and outward leaders, respectively.

Assume that $\mathcal{I}$ has an affirmative solution $\mathcal{S}$ and let C be a corresponding xy-separating curve. Let $up(C)$ be the polygonal region above C bounded by $R(t)$ and $R(l)$. Similarly, let $right(C)$ be the polygonal region to the right of C bounded by $R(b)$ and $R(r)$. By C_u (resp., C_r) we denote the xy-separating curve that minimizes the area of $up(C_u)$ (resp., $right(C_r)$),

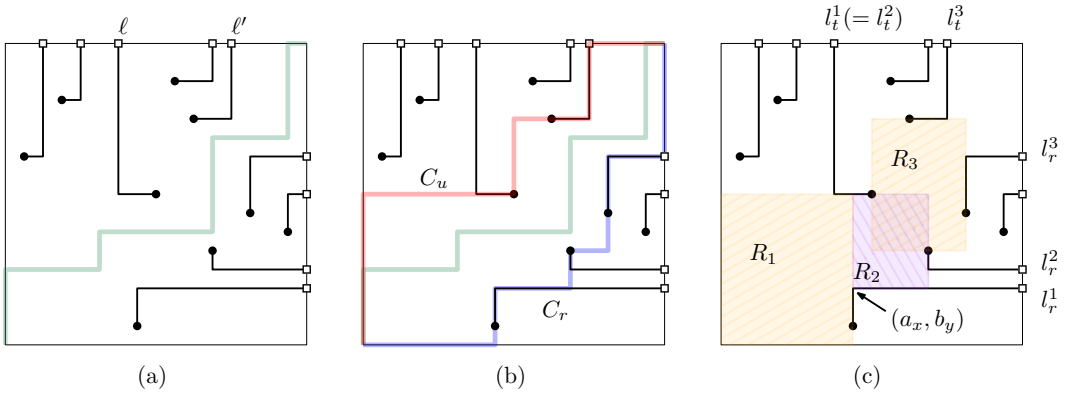


Figure 2 (a) An xy-separated solution to a 2-sided boundary labeling. The xy-separating curve C is shown in light-green. (b) Illustration for the curves C_u and C_r. (c) $\mathcal{R}$.

e.g., see Figure 2(b). For a point p, let p_x and p_y be its x and y-coordinates, respectively. Given C_u and C_r, we define a sequence of rectangles $\mathcal{R} = (R_1, R_2, \ldots, R_k)$ as follows:

- Each rectangle is a maximal rectangle between C_u and C_r.
- The bottom-left corner of R_1 coincides with that of R.
- For $i > 1$, we first consider R_{i-1}. Since R_{i-1} is maximal, the top and right sides of R_{i-1} must be determined by a pair of leaders, e.g., see R_1 in Figure 2(c). Denote these leaders by ℓ_t^{i-1} and ℓ_r^{i-1}, respectively. Let $a \in \ell_t^{i-1}$ be the rightmost point of ℓ_r^{i-1} on the top side of R_{i-1}, and let $b \in \ell_r^{i-1}$ be the topmost point on the right side of R_{i-1}. We define R_i to be the maximal empty rectangle with the bottom-left corner at (a_x, b_y) and the sides bounded by C_u and C_r.

2.2 Algorithm

The idea of the algorithm is to employ a dynamic programming algorithm based on the idea of finding the optimal rectangle sequence $\mathcal{R}$. Note that for any rectangle $R_j \in \mathcal{R}$, we can think of a subproblem $\lambda(R_j)$ that seeks a solution including the leaders ℓ_t^j and ℓ_r^j. More formally, $\lambda(R_j)$ is an instance of the 2-sided boundary labeling problem for which the rectangle $B(R_j)$ corresponding to this problem is determined by the vertical segment of ℓ_t^j, the horizontal segment of ℓ_r^j as well as the top and right sides of the rectangle R; see the gray rectangle in Figure 3(a). It is straightforward to add a dummy rectangle R_0 with corresponding leaders ℓ_t^0 and ℓ_r^0 such that $\lambda(R_0)$ represents the original 2-sided boundary labeling problem; e.g., see Figure 3(b).

Given R_j, we try to find R_{j+1} by checking all possible candidate rectangles. For convenience, we defer the details of finding all candidate rectangles, and focus on the computation of the solution cost (sum of leader length) assuming that we have found R_{j+1}. Figure 3(c) illustrates such a scenario. Let $R^t_{j,j+1}$ be the region bounded by the lines determined by the vertical segments of ℓ_t^j and ℓ_t^{j+1}, the horizontal segment of ℓ_t^j, and $R(t)$. Define $R^r_{j,j+1}$ symmetrically, e.g., see the top of Figure 4(i). Observe that $\lambda(R^t_{j,j+1})$ is a 1-sided boundary labeling problem with leaders ℓ_t^j and ℓ_t^{j+1}. In other words, since R_{j+1} is an empty rectangle, all the ports between ℓ_t^j and ℓ_t^{j+1} must be connected to some site interior to $R^t_{j,j+1}$. We define $\lambda(R^r_{j,j+1})$ symmetrically. It is now straightforward to express the solution of $\lambda(R_j)$ in terms of the solutions of $\lambda(R^t_{j,j+1})$, $\lambda(R^r_{j,j+1})$, and $\lambda(R_{j+1})$.

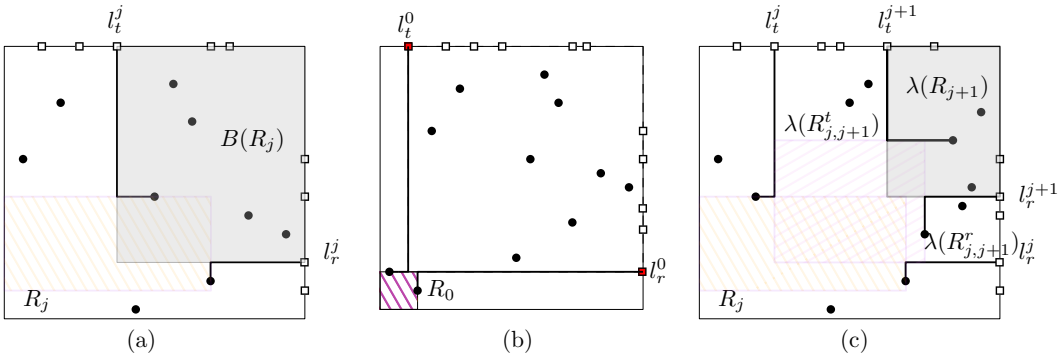


Figure 3 Illustration for the dynamic programming algorithm.

For any leader l, we denote its length by $|l|$. Let $|\lambda(R_j)|$ be the sum of the leader lengths in an optimal solution of $\lambda(R_j)$ (excluding the lengths of ℓ_t^j and ℓ_r^j). Let $ports(B(R_j))$ and $sites(B(R_j))$ be the number of ports and sites interior to $B(R_j)$, excluding those that are incident to ℓ_t^j and ℓ_r^j. We now have the following recursive formula, where $\mathcal{C}$ denotes the set of candidate rectangles.

$$|\lambda(R_j)| = \begin{cases} \infty, & \text{if } ports(B(R_j)) \neq sites(B(R_j)). \\ (|\ell_t^j| + |\ell_r^j|) + & \\ \min\limits_{R_{j+1} \in \mathcal{C}} \{|\lambda(R^t_{j,j+1})| + |\lambda(R^r_{j,j+1})| + |\lambda(R_{j+1})|\}, & \text{otherwise.} \end{cases}$$

Finding candidate rectangles. Given a rectangle R_j, we now describe how to find a set of candidate rectangles that must include R_{j+1}. Recall that we can compute the bottom-left corner (a_x, b_y) of R_{j+1} from R_j. Figures 4(a)–(d) illustrate the scenarios where ℓ_t^j and ℓ_r^j are inward. The point (a_x, b_y) is marked with a cross. We claim that the top side or the right side of R_{j+1} must contain a site (Lemma 3). We will use the following result of Benkert et al. [7] to prove Lemma 3.

▶ **Lemma 2** (Benkert et al. [7]). *For any solution S to a 1-bend 1-sided boundary labeling problem that minimizes the total leader length (possibly with crossings), there exists a crossing-free labeling with the total leader length at most the total leader length of S.*

▶ **Lemma 3.** *The top side or the right side of R_{j+1} must contain a site.*

Proof. Suppose for a contradiction that neither the top nor the right side of R_{j+1} contains a site. We now consider four cases.

Case 1 (both ℓ_t^{j+1} and ℓ_r^{j+1} are inward): In this case the leaders ℓ_t^{j+1} and ℓ_r^{j+1} must intersect (see Figure 4(e)), which contradicts that the underlying solution is crossing-free.

Case 2 (ℓ_t^{j+1} is inward and ℓ_r^{j+1} is outward): If ℓ_r^j is outward, then it must intersect ℓ_r^{j+1} (see Figure 4(f)). Therefore, the leader ℓ_r^j must be inward, as illustrated in Figure 4(g). Note that by our general position assumption, the 'y-intervals' determined by the vertical segments of ℓ_r^j and ℓ_r^{j+1} must overlap. Consequently, by swapping the site assignments, we can obtain a solution (possibly with crossings) with strictly smaller total leader length. Figure 4(i) illustrates such a scenario. By Lemma 2, we can replace this labeling of $\lambda(R_{j,j+1})$ with a crossing free labeling that lies inside $R^r_{j,j+1}$ and does not increase the total leader length, e.g., see Figure 4(j). Note that the total leader length of the resulting solution would be strictly smaller, contradicting that the current solution is optimal.

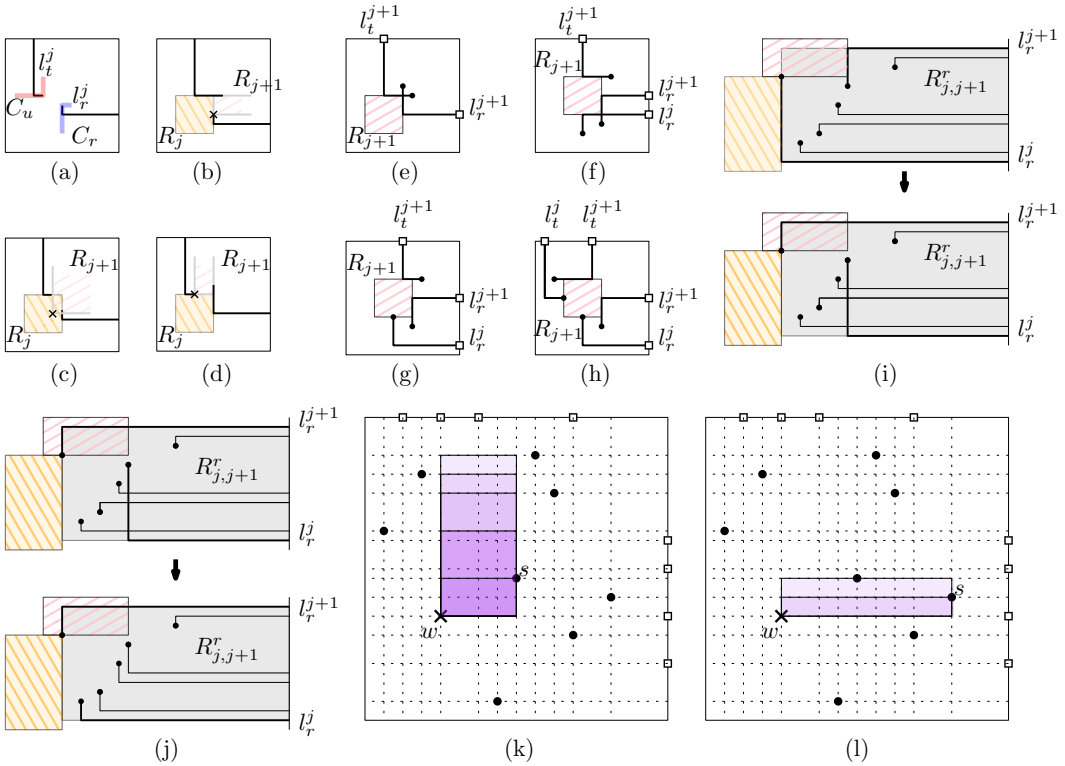


Figure 4 Illustration for (a)–(d) (a_x, b_y), (e)–(j) Lemma 3, and (k)–(l) candidate rectangles.

Case 3 (ℓ_t^{j+1} is outward and ℓ_r^{j+1} is inward): This case is symmetric to Case 2.
Case 4 (both ℓ_t^{j+1} and ℓ_r^{j+1} are outward): We can process this case in the same way as we did in Case 2. ◀

Recall that we know the bottom-left point w of R_{j+1}. We first assume that the right side of R_{j+1} contains a site. For every site s with $s_x > w_x$ and $s_y > w_y$, we consider all possible empty rectangles with bottom-left corner w, right side passing through s and the top side determined by a horizontal line passing through a site above s. Figures 4(k)–(l) illustrate the candidate rectangles for the bottom left point w. We then assume that the top side of R_{j+1} contains a site, and find the candidate rectangles symmetrically. We can now obtain an upper bound on the distinct candidate rectangles.

▶ **Lemma 4.** *The overall number of distinct candidate rectangles is $O(n^3)$.*

Proof. For a particular bottom-left corner w, it may initially appear that there are $O(n^2)$ possible candidate rectangles to explore. But we can prove an $O(n)$ upper bound, as follows.

Let D be the region dominated by w; i.e., for each point $q \in D$, the x and y-coordinates of q are at least as large as those of w. Let $S = \{s_1, s_2, \ldots, s_k\}$ be the set of sites in D (ordered by increasing y-coordinates) such that no site in S is dominated by any other site in D (except possibly for w). We may assume without loss of generality (see Lemma 3) that the right side of R_{j+1} contains a site. Since the proper interior of the rectangle is empty, for each s_i, where $1 \leq i \leq k$, we only need to consider a set of heights $H(s_i)$ that lie between s_i and s_{i+1} (or, between s_i and $R(t)$ when $i = k$). For every pair of sites $\{s, s'\} \in S$, we have the property that neither s nor s' dominates the other. Therefore, we have $H(s) \cap H(s') = \emptyset$, $\sum_i H(s_i) = O(n)$, and thus a linear number of candidate rectangles for w.

The number of possible intersections (i.e., bottom-left corners) among the horizontal and vertical lines passing through the ports and sites is $O(n^2)$. Therefore, the number of distinct candidate rectangles that may appear over the run of the algorithm is $O(n^3)$. ◀

Data structures and time complexity. If we use an $O(n^2) \times O(n^2)$ dynamic programming table and compute each entry by checking $O(n)$ candidate rectangles, then we need at least $O(n^5)$ time. To improve the running time to $O(n^3 \log n)$, we preprocess the input. For every possible matching of a pair of ports (on the same side of R) to a pair of sites, we compute and store the solution to the corresponding 1-sided boundary labeling problem. Since there are $O(n^4)$ such 1-sided problems, and each of them can be answered in $O(n \log n)$ time [7], this takes $O(n^5 \log n)$ time. We first show how to reduce this preprocessing time to $O(n^3 \log n)$.

Consider a subproblem $\lambda(R^t_{j,j+1})$. Such a problem can easily be expressed by the ports and sites incident to ℓ^j_t and ℓ^{j+1}_t. Here we encode $\lambda(R^t_{j,j+1})$ in a slightly different way. We use the parameters p, p', α, β, where p, p' are the ports incident to ℓ^j_t and ℓ^{j+1}_t, α is either ∞ or the y-coordinate of a site that indicates the top side of an "empty rectangle", which is used in our preprocessing, and β is the 'type' of $\lambda(R^t_{j,j+1})$. We will express $\lambda(R^t_{j,j+1})$ as $S(p, p', \alpha, \beta)$. In the following we describe the details of $S(p, p', \alpha, \beta)$.

Note that to solve $\lambda(R^t_{j,j+1})$ affirmatively, we need exactly as many free sites as the number of ports between p and p'. Thus for any subproblem, if the number of free sites and free ports interior to $R^t_{j,j+1}$ do not match, then we can immediately return a negative answer. We assume that the points and ports are stored in an orthogonal range counting data structure (with $O(n \log n)$-time preprocessing) such that given an orthogonal rectangle, one can report the number of ports and points interior to the rectangle in $O(\log n)$ time [9]. We only focus on those instances that have the same number of free sites and ports, and express them in the form $S(p, p', \alpha, \beta)$.

Let s, s' be the sites that are incident to ℓ^j_t and ℓ^{j+1}_t, respectively. By the property of the optimal solution, we may assume that $s_y < s'_y$. We define $\lambda(R^t_{j,j+1})$ as having Type 1, 2, 3 or 4 depending on whether s, s' belongs to $R^t_{j,j+1}$ or not.

Type 1 (both s, s' are outside $R^t_{j,j+1}$): In this case the rectangle determined by the bend points of ℓ^j_t and ℓ^{j+1}_t must be empty (i.e., the gray region in Figure 5(a)). We set α to be ∞, and β to be 1. During the algorithm execution, if $\lambda(R^t_{j,j+1})$ is of Type 1, then we will seek a solution to $S(p, p', \infty, 1)$.

Note that for any instance of the form $S(p, p', \infty, 1)$, we can determine in $O(1)$ time[2] the point s'' such that the rectangle B determined by p, p', s'' contains an equal number of free ports and sites. Note that the solution to the labeling problem inside B will be equivalent to that of $\lambda(R^t_{j,j+1})$. We will precompute the solutions of $S(p, p', \infty, 1)$ so that $\lambda(R^t_{j,j+1})$ can be answered in $O(1)$ time by a table look-up. This general idea of answering a problem $\lambda(\cdot)$ using $S(\cdot)$ applies also to the other types; i.e., Types 2, 3 and 4.

Type 2 (both s, s' are inside $R^t_{j,j+1}$): In this case the rectangle determined by the bend point of ℓ^{j+1}_t and s must be empty; see Figures 5(c). (Notice that the case shown in Figure 5(b) is not possible in an optimal solution because re-routing p' to s and p to s' would result in a feasible solution with a smaller total leader length.) We thus set α to be $y(s')$, and β to be 2. Observe now that given $S(p, p', \alpha, 2)$, we can find both s and s' in $O(1)$ time[2] by counting the number of ports between p and p', and using α.

[2] It is straightforward to preprocess the ports and sites in $O(n^3)$ time in a data structure to answer such queries in $O(1)$ time.

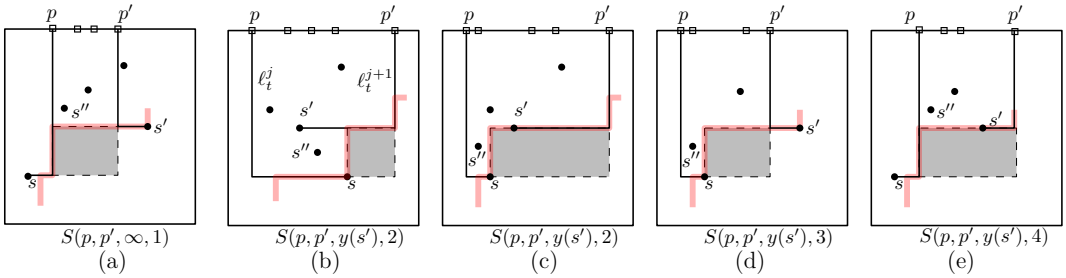


Figure 5 (a)–(e) Illustration for different Types of subproblems.

Type 3 ($s \in R^t_{j,j+1}$ and $s' \notin R^t_{j,j+1}$): In this case the rectangle determined by the bend point of ℓ_t^{j+1} and s must be empty (Figure 5(d)). We thus set α to be $y(s')$, and β to be 3. Given $S(p, p', \alpha, 3)$, we can recover s and s' using the range counting data structures[2]. The same argument holds even when $s_x > p'_x$.

Type 4 ($s \notin R^t_{j,j+1}$ and $s' \in R^t_{j,j+1}$): In this case the rectangle determined by the bend points of ℓ_t^j and ℓ_t^{j+1} must be empty (Figure 5(e)). We thus set α to be $y(s')$, and β to be 4. Given $S(p, p', \alpha, 4)$, we can recover s' using α. Here we do not need to find s since the solution must lie inside the rectangle determined by p, p' and s'.

▶ **Lemma 5.** *The solution to the problems $S(p, p', \alpha, \beta)$ can be computed in $O(n^3 \log n)$ time.*

Proof. Since there are $O(n)$ possible choices for each of p, p', α, and a constant number of choices for β, we have at most $O(n^3)$ subproblems. We can employ a dynamic programming to compute the solution to these problems. The idea is to select the bottommost free point s'' and connect it to a port p'' between p and p'. This splits the problem into two subproblems, which can again be expressed in the form $S(p, p', \alpha, \beta)$. Such a split may generate a new type of subproblem $\mathcal{Q}$, where ℓ_t^j has a shorter height than that of ℓ_t^{j+1}. Since ℓ_t^j was initially incident to s'', we can process $\mathcal{Q}$ as follows: For every pair of ports, we use Benkert et al.'s [7] algorithm to precompute the solution to the boundary labeling problem inside the stripe bounded by the vertical lines through p, p'. If there are k ports between p and p', then we use the topmost k sites in the stripe (if it exists). This preprocessing takes $O(n^3 \log n)$ time. To answer $\mathcal{Q}$, we use the precomputed solution for the corresponding stripe.

Since the number of choices for p'' is at most n, we can compute an entry of the dynamic programming table by a linear number of table look-up. Since the number of entries is $O(n^3)$, the running time is bounded by $O(n^4)$. An involved analysis shows that there is only $O(1)$ candidate choices for p'', and these candidates can be found in $O(\log n)$ time. The full version of the paper [8] includes the details. Since an entry of the dynamic programming table can now be computed using $O(1)$ number of table look-ups, the running time reduces to $O(n^3 \log n)$. ◀

▶ **Theorem 6.** *Given a 1-bend 2-sided boundary labeling problem with $O(n)$ sites and labels, one can find a labeling (if exists) that minimizes the total leader length in $O(n^3 \log n)$-time.*

Proof. Every subproblem $\lambda(R_j)$ can be defined by a pair of leaders, and hence we can define an $O(n^2) \times O(n^2)$ table T to store the solutions to the subproblems. To compute an entry of the table T, we look at a set of candidate rectangles with two nice properties. First, all these rectangles have the same bottom-left corner, and second, none of these rectangles can be a candidate rectangle for any other entry of T. Therefore, the number of 'candidate

rectangle queries' to fill all the entries of T is bounded asymptotically by the number of distinct candidate rectangles, which is $O(n^3)$ (by Lemma 4). Since we do not recompute solutions, and the table look-up takes $O(1)$ time, the total running time is bounded by $O(n^4)$, which dominates the preprocessing time.

Observe that the complexity $O(n^4)$ comes from considering all possible pairs of leaders, whereas only $O(n^3)$ options are relevant (by Lemma 4). Therefore, instead of a table, we can keep the relevant entries in a dynamic binary search tree, which increases the cost for solution look-up to $O(\log n)$, but limits the time for both the memory initialization and look-up queries to $O(n^3 \log n)$. Thus the total running time improves to $O(n^3 \log n)$. ◀

3 Relating Boundary Labeling to Outerstring Graphs

In this section, we reduce the boundary labeling problem to the independent set problem on a class of weighted geometric intersection graphs in the plane called outerstring graphs. We show that if one can discretize a boundary labeling problem such that the number of candidate leaders is a polynomial in n, then our approach will yield a polynomial-time algorithm for the problem.

An *outerstring graph* is an intersection graph of a set of curves in the Euclidean plane that lie inside a polygon such that one of the endpoints of each curve is attached to the boundary of the polygon. Keil et al. [16] gave an $O(N^3)$-time algorithm for the maximum-weighted independent set problem on outerstring graphs. The algorithm requires an outerstring graph as an input, where each curve is given as a polygonal line (i.e., a chain of straight line segments) and N is the number of segments in the representation. We show that by discretizing the boundary labeling problem and assigning an appropriate weight to each candidate leader, one can reduce the boundary labeling problem to the maximum-weighted independent set problem on outerstring graphs. Here, as an example, we show the reduction for the boundary labeling problem using *po*- and *opo*-leaders in the presence of obstacles.

Boundary labeling with orthogonal obstacles. Fink and Suri [11] gave $O(n^9)$ and $O(n^{21})$-time algorithms for the opposite 2-sided boundary labeling with *po*- and *opo*-leaders, respectively. Our approach will yield $O(n^6)$ and $O(n^{12})$-time algorithms for *po*- and *opo*-leaders, respectively, irrespective of the labeling model (opposite, adjacent, or for any port distribution on the boundary). For the opposite 2-sided case, the running time reduces to $O(n^6)$ and $O(n^9)$ (for *po*- and *opo*-leaders, respectively). This will settle the time complexity question of 1-bend 3- and 4-sided boundary labeling [17]. In the rest of this section, we relax the general position assumption and denote n to be the total number of sites and obstacle vertices.

First consider the case of *po*-leaders. Let I be an instance of the boundary labeling problem. Given a site and a port, there is at most one way of connecting them. Let M denote the set of all possible leaders that do not intersect any obstacle. Then $|M| \in O(n^2)$. It is straightforward to compute M in $O(n^3)$ time. Observe that each leader $l \in M$ can be viewed as an outerstring, and let $st(l)$ be the corresponding outerstring. Let $|l|$ be the length of the leader l, and define $x := \max_{l \in M} |l|$ and $y := \min_{l \in M} |l|$. Let C be a number such that $C > nx - (n-1)y > 0$. For each leader $l \in M$, we assign a weight $w(st(l))$ to $st(l)$, where $w(st(l)) := C - |l|$. The following lemma and Keil et al.'s [16] algorithm lead us to the results for *po*-leaders (Theorem 8).

▶ **Lemma 7.** *I has a feasible solution with total leader length L if and only if the corresponding outerstring graph G_I has a feasible solution with total weight $(nC - L)$.*

Proof. A feasible solution S of I with total leader length L gives a feasible solution for G_I with total weight

$$\sum_{l \in S} w(st(l)) = \sum_{l \in S} (C - |l|) = nC - \sum_{l \in S} |l| = (nC - L).$$

We now assume that G_I has a feasible solution S' with total weight $W = (nC - L)$, and show that the corresponding leaders S yields a feasible solution of I of total leader length L. Since S' is an independent set, the leaders in S are crossing-free, as well as no site or port is incident to more than one leader. It now suffices to show that every site is connected to a string, i.e., $|S| = n$ and the total leader length is L. Observe that $W = nC - L \geq nC - nx > nC - (n-1)y - C = (n-1)(C-y)$. If $|S| < n$, then the total leader length is at most $(n-1)x$, and S' has weight at most $(n-1)(C-y)$, which contradicts that $W > (n-1)(C-y)$. Therefore, $|S| = n$, and we have

$$W = \sum_{s \in S'} w(s) = \sum_{s \in S'} (C - |l_i|) = nC - \sum_{l \in S'} |l|.$$

Since $W = (nC - L)$, we have $\sum_{l \in S'} = L$. ◀

▶ **Theorem 8.** *The boundary labeling problem can be solved in $O(n^6)$ time using po-leaders, for both adjacent and opposite sided models, even in the presence of obstacles (where n is the total number of sites and vertices of the obstacles).*

Consider now the case for *opo*-leaders. For opposite 2-sided case, Fink and Suri [11] showed that one can discretize the problem such that if there exists a feasible solution, then there is one where the x-coordinate of the middle segment of every leader lies in the set of all x-coordinates of the sites and obstacle vertices. Therefore, we have $O(n)$ potential leaders for each port-site pair, and thus $O(n^3)$ leaders in total. Hence applying Keil et al.'s [16] algorithm gives a running time of $O(n^9)$.

The discretization of [11] does not apply to the 3- and 4-sided case. However, consider a grid H determined by the axis-aligned lines through the ports, sites and obstacle vertices. For each pair of consecutive parallel lines of H, place a set of n parallel lines in between. Let the resulting grid be H'. If there is a feasible solution to the boundary labeling problem, then for any pair of consecutive parallel vertical lines ℓ, ℓ' (similarly for horizontal) of H, we can have at most n middle vertical segments of the leaders. We thus can distribute them by moving horizontally to the n lines of H' (e.g., see [11]), which does not change the total leader length. By construction, there is no site, port or obstacle vertex between ℓ and ℓ'. Hence such a modification can be performed without introducing any crossing. Since H' is an $O(n^2) \times O(n^2)$ grid and since we have $O(n^2)$ potential leaders for each port-site pair, the number of candidate leaders is $O(n^4)$. Hence applying Keil et al.'s [16] algorithm gives a running time of $O(n^{12})$.

▶ **Theorem 9.** *The adjacent boundary labeling problem can be solved in $O(n^{12})$ time using opo-leaders, even in the presence of obstacles (where n is the total number of sites and vertices of the obstacles). For opposite 2-sided models, the running time reduces to $O(n^9)$.*

Sliding ports and bend minimization. The outerstring-graph approach can also be applied to the sliding port model, where each label is assigned a distinct interval on the boundary of R and a site can be connected to any point of an interval. The goal here is to minimize the total leader length or the number of bends. We only need to discretize the problem such

that the number of strings that we need to consider is a polynomial in n. Define H to be a grid determined by the axis-aligned lines through sites, interval boundaries and obstacle vertices. Construct H' from H by introducing for every pair of consecutive parallel lines of H, a set of $2n$ parallel lines in between.

The grid H' can be used to discretize the problem, as follows. The segments incident to the sites are already on H. Consider now a vertical (similarly for horizontal) segment ℓ that is incident to an interval I, but not incident to any site. Let ℓ' and ℓ' be a pair of consecutive horizontal lines of H such that ℓ lies between them. There can be at most $2n$ horizontal lines between ℓ, ℓ', which we can distribute to the lines of H' by moving vertically (e.g., see [11]). Since there cannot be any site, interval boundary or obstacle vertex between ℓ, ℓ', such a modification neither introduce crossings nor increase the number of total bends. By the construction of H, the boundary of R between ℓ, ℓ' lies in the interval I. Hence ℓ will still be incident to I. Finally, the middle segments of the leaders can be processed in the same way as we did for Theorem 9. It is straightforward to observe that the number of potential strings is a polynomial in n. We can now assign certain weights to these strings such that the maximum-weight independent set of the corresponding outerstring graph yields a minimum-bend solution for the boundary labeling problem.

We first consider the case of *po*-leaders. Let I be an instance of this problem. Consider the set M of outerstrings as before. For each outerstring $st(l) \in M$, we assign the weight $w(st(l))$, where

$$w(st(l)) = \begin{cases} n+2, & \text{if } l \text{ has no bends.} \\ n+1, & \text{if } l \text{ has one bend.} \end{cases} \tag{1}$$

This forms our instance G_I of an outerstring graph on which we solve the maximum-weighted independent set problem by running Keil et al.'s algorithm [16].

▶ **Lemma 10.** *Let I be an instance of the boundary labeling problem with po-leaders. Then I has a feasible solution with k bends if and only if the instance G_I has a feasible solution W with total weight at least $(n^2 + 2n - k)$.*

Proof. Let S be a feasible solution of I. Clearly, the strings corresponding to the leader of S' is a feasible solution for G_I. Let k be the total number of bends in S. Then the weight of S' is $\sum_{l \in S} w(l) \geq (n+1)k + (n+2)(n-k) = n^2 + 2n - k$.

Assume now that G_I has a feasible solution S with weight at least $n^2 + 2n - k$. Let S' be the corresponding set of leaders in I. Since S is an independent set, a port or site can be incident to at most one leader of S. If a site is not connected to any port in S', then at most $(n-1)$ sites are incident to a leader. Since the maximum weight of a leader can be at most $(n+2)$, the weight of S is at most $(n-1)(n+2) = (n^2+n-2)$, which is a contradiction since the weight of S is at least $n^2 + 2n - k > (n^2 + n - 2)$ (because $n \geq k$). Therefore, $|S'| = n$.

It now remains to show that the weight of S' is at most k. Suppose for a contradiction that S' has at least $(k+1)$ *po*-leaders. Therefore, the weight of S is at most $(n+1)(k+1) + (n+2)(n-k-1) = n^2 + 2n - (2k+1) < (n^2 + 2n - k)$, which is a contradiction that the weight of S is at least $(n^2 + 2n - k)$. ◀

Now, we consider the case of *opo*-leaders. Let I be an instance of this problem. Consider the set M of outerstrings as before. For each outerstring $st(l) \in M$, we assign the weight

$w(st(l))$ as follows:

$$w(st(l)) = \begin{cases} \alpha + 3, & \text{if } l \text{ has no bends,} \\ \alpha + 2, & \text{if } l \text{ has one bend,} \\ \alpha + 1, & \text{if } l \text{ has two bends.} \end{cases} \tag{2}$$

Here, $\alpha = 2n$.

▶ **Lemma 11.** *Let I be an instance of the boundary labeling problem with opo-leaders. Then I has a feasible solution with k bends if and only if the instance G_I has a feasible solution W with total weight at least $(\alpha n + 3n - k)$.*

Proof. Let S be a feasible solution of I. Clearly, the strings corresponding to the leader of S' is a feasible solution for G_I. Let k be the total number of bends in S, and let k_1 and k_2 be the number of strings with 1-bend and 2-bends, respectively. Therefore, $k_1 + 2k_2 = k$, and the weight of S' is $\sum_{l \in S} w(l) = (\alpha+2)k_1 + (\alpha+1)k_2 + (\alpha+3)(n - k_1 - k_2) = \alpha n + 3n - k_1 - 2k_2 = \alpha n + 3n - k$.

Assume now that G_I has a feasible solution S with weight at least $(\alpha n + 3n - k)$. Let S' be the corresponding set of leaders in I. Since S is an independent set, a port or site can be incident to at most one leader of S. If a site is not connected to any port in S', then at most $(n-1)$ sites are incident to a leader. Since the maximum weight of a leader can be at most $(\alpha + 3)$, the weight of S is at most $(n-1)(\alpha+3) = (\alpha n + 3n - \alpha - 3)$, which is a contradiction since the weight of S is at least $\alpha n + 3n - k > (\alpha n + 3n - \alpha - 3)$ (because $\alpha = 2n \geq k$). Therefore, $|S'| = n$.

It now remains to show that the leaders of S' has at most k bends. Suppose for a contradiction that S' has at least k_1 *po*-leaders and k_2 *opo*-leaders such that $k_1 + 2k_2 \geq k+1$. Therefore, the weight of S is at most $(\alpha + 2)k_1 + (\alpha + 1)k_2 + (\alpha + 3)(n - k_1 - k_2) = \alpha n + 3n - (k_1 + 2k_2) - (2k_1 + k_2) + (2k_1 + k_2) = \alpha n + 3n - (k_1 + 2k_2)$. Since $k_1 + 2k_2 \geq k + 1$, the weight of S is strictly less than $\alpha n + 3n - k$, which is a contradiction. ◀

By Lemmas 10 and 11, we have the following theorem (which settles two open questions of [20, Table 23.1]).

▶ **Theorem 12.** *A boundary labeling that minimizes the total number of bends can be computed (if exists) in polynomial time for both adjacent and opposite models (with sliding ports, po and opo-leaders), even in the presence of obstacles.*

4 Conclusion

The most natural directions for future research is to improve the time complexity of our algorithm for the 1-bend adjacent 2-sided model. A number of intriguing questions follow: Can we find a non-trivial lower bound on the time-complexity? Is the problem 3-sum hard or, as hard as 'sorting $X + Y$'? Can we check the feasibility in near-linear time? It would also be interesting to find fast approximation algorithms for boundary labeling problems.

References

1 Alexander Wolff and Tycho Strijk. The map-labeling bibliography. `http://i11www.ira.uka.de/map-labeling/bibliography/`. Online; accessed 10 February, 2018.

2 Lukas Barth, Andreas Gemsa, Benjamin Niedermann, and Martin Nöllenburg. On the readability of boundary labeling. In *23rd International Symposium Graph Drawing and Network Visualization (GD 2015), Los Angeles, CA, USA*, pages 515–527, 2015.

3 Lukas Barth, Benjamin Niedermann, Martin Nöllenburg, and Darren Strash. Temporal map labeling: A new unified framework with experiments. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS)*, pages 23:1–23:10. ACM, 2016.

4 Michael A. Bekos, Sabine Cornelsen, Martin Fink, Seok-Hee Hong, Michael Kaufmann, Martin Nöllenburg, Ignaz Rutter, and Antonios Symvonis. Many-to-one boundary labeling with backbones. *J. Graph Algorithms Appl.*, 19(3):779–816, 2015.

5 Michael A. Bekos, Michael Kaufmann, Martin Nöllenburg, and Antonios Symvonis. Boundary labeling with octilinear leaders. *Algorithmica*, 57(3):436–461, 2010.

6 Michael A. Bekos, Michael Kaufmann, Antonios Symvonis, and Alexander Wolff. Boundary labeling: Models and efficient algorithms for rectangular maps. *Comput. Geom.*, 36(3):215–236, 2007.

7 Marc Benkert, Herman J. Haverkort, Moritz Kroll, and Martin Nöllenburg. Algorithms for multi-criteria boundary labeling. *J. Graph Algorithms Appl.*, 13(3):289–317, 2009.

8 Prosenjit Bose, Paz Carmi, J. Mark Keil, Saeed Mehrabi, and Debajyoti Mondal. Boundary labeling for rectangular diagrams. *CoRR*, abs/1803.10812, 2018. URL: `https://arxiv.org/abs/1803.10812`.

9 Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer, Berlin Heidelberg, 2008.

10 Srinivas Doddi, Madhav V. Marathe, Andy Mirzaian, Bernard M. E. Moret, and Binhai Zhu. Map labeling and its generalizations. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 148–157, 1997.

11 Martin Fink and Subhash Suri. Boundary labeling with obstacles. In *Proceedings of the 28th Canadian Conference on Computational Geometry (CCCG)*, pages 86–92, 2016.

12 Michael Formann and Frank Wagner. A packing problem with applications to lettering of maps. In *Proceedings of the Seventh Annual Symposium on Computational Geometry (SoCG)*, pages 281–288. ACM, 1991.

13 Herbert Freeman. An expert system for the automatic placement of names on a geographic map. *Inf. Sci.*, 45(3):367–378, 1988.

14 GLEAM. `http://www.greatlakesmapping.org/`. Online; accessed 10 February, 2018.

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

16 J. Mark Keil, Joseph S. B. Mitchell, Dinabandhu Pradhan, and Martin Vatshelle. An algorithm for the maximum weight independent set problem on outerstring graphs. *Comput. Geom.*, 60:19–25, 2017.

17 Philipp Kindermann, Benjamin Niedermann, Ignaz Rutter, Marcus Schaefer, André Schulz, and Alexander Wolff. Multi-sided boundary labeling. *Algorithmica*, 76(1):225–258, 2016.

18 Benjamin Niedermann, Martin Nöllenburg, and Ignaz Rutter. Radial contour labeling with straight leaders. In *2017 IEEE Pacific Visualization Symposium (PacificVis 2017), Seoul, South Korea*, pages 295–304, 2017.

19 Martin Nöllenburg, Valentin Polishchuk, and Mikko Sysikaski. Dynamic one-sided boundary labeling. In *18th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems (GIS)*, pages 310–319, 2010.

20 Alexander Wolff. Graph drawing and cartography. In Roberto Tamassia, editor, *Handbook of graph drawing and visualization*, chapter 23, pages 697–736. CRC Press, 2014.

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

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