Elastic Labels Around the Perimeter of a Map *

Claudia Iturriaga[1] ** and Anna Lubiw[2]

[1] University of New Brunswick, Canada, citurria@unb.ca
[2] University of Waterloo, Canada, alubiw@uwaterloo.ca

Abstract. In this paper we study the map labeling problem of attaching rectangular labels to points, but with the novelty that our labels are *elastic*, in the sense that the height and width of each rectangle may vary though we require a fixed area. Our main result is a polynomial time algorithm for the *rectangle perimeter labeling problem*, where the points to be labeled lie on the boundary of a rectangular map. This problem is likely to be relevant in Geographical Information Systems (GIS) as maps are displayed dynamically on a computer screen using clipping, panning, and zooming.

1 Introduction

One of the fundamental tasks in cartography is the **labeling** of maps—attaching text to geographic features. In fact, the label placement problem has been identified as a key problem by the ACM Computational Geometry Impact Task Force [1]. Researchers have developed algorithms and heuristics for labeling features that are points, curves, and regions. Wolff and Strijk provide a good bibliography of this area of research [13].

Many of the issues become simpler when the features to be labeled are points because we expect the text to be placed horizontally and close to the associated point. Some examples of point features are cities and towns on a large scale map, and on a small scale map, drill sites, hospitals, electrical power stations and post offices. We want labels that do not overlap and are large enough to be readable. The most common formulation of this problem is the *point-feature label placement problem*: given a set of points in the plane, and an axis-parallel rectangular label associated with each point, place each label with one corner at the associated point such that no two labels overlap.

The point-feature label placement problem is known to be NP-complete [4, 8, 9, 11]. Kučera et al. [10] gave exact sub-exponential time algorithms, though for large problem instances these algorithms are not practical. Heuristics have been proposed for the problem [2, 4, 12, 3]. Formann and Wagner [4] gave a polynomial time algorithm for the case when labels have two candidate positions instead of four.

Traditionally, a label contains just one or two words, e.g. the name of a city. We focus on *text labels* that contain a paragraph or more of information. Such

* Research partially supported by NSERC.

** This work was done while the first author was at the University of Waterloo.

F. Dehne, A. Gupta, J.-R. Sack, and R. Tamassia (Eds.): WADS'99, LNCS 1663, pp. 306-317, 1999
© Springer-Verlag Berlin Heidelberg 1999

labels may contain, for example, descriptions of restaurants and tourist sites.

A paragraph can be formatted to fit into rectangles of different aspect ratios. We can write the paragraph out as one long line, or use two lines and about half the width, or etc. These rectangles have roughly the same area. (See Figure 1.) This leads us to model a text label as an *elastic label* that has fixed area, but varying height and width. We allow height to vary continuously though it is more appropriate for text labels to have only a discrete set of heights, corresponding to the number of lines. Our methods do extend to this case.

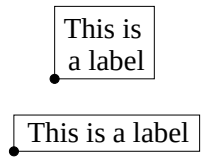

Fig. 1. Text labels.

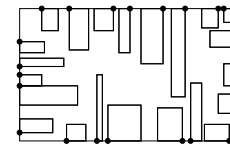

Fig. 2. A solution for an instance of the rectangle perimeter labeling problem.

The *elastic labeling problem* is to choose the height and width of each label, and the corner of the label to place at the associated point, so that no two labels overlap. The elastic labeling problem is an NP-hard problem since it generalizes the point-feature label placement problem. That is, the problem is NP-hard even when there is no elasticity just because of the choice of the corners. In [5] we show that the problem also remains NP-hard even when we have elasticity but no choice about which corner of each label to use. We call this the *one-corner elastic labeling problem*. Note that if we must use the *same* corner of each elastic label the problem can be solved in polynomial time by a sweep algorithm [7].

Since the elastic labeling problem remains NP-hard even when we fix the corners of the labels, we consider a problem with more constraints, but still of practical use. We require that the points lie on the boundary of a rectangular map. This *rectangle perimeter labeling problem* arises, for example, when the perimeter of a map is labeled with information about objects that lie beyond the boundary of the map, e.g. where the roads lead to. Figure 2 shows a solution to an instance of the rectangle perimeter labeling problem. This problem is likely to be relevant in GIS as maps are displayed dynamically on a computer screen using clipping, panning, and zooming.

Our main result is a polynomial time algorithm for the rectangle perimeter labeling problem. We first tackle two subproblems in which the points lie on only two sides of the rectangle, either two adjacent sides (the *two-axis labeling problem*) or two opposite sides (the *two-parallel lines labeling problem*).

The rest of this paper is organized as follows. In section 2, we present the definitions and notation that will be used. In section 3 we give a brief description of the *two-axis labeling problem* presented in [6], and in section 4 we study the *two-parallel lines labeling problem*. Finally, in section 5, we combine these results to solve the general rectangle perimeter labeling problem.

2 Definitions and Notation

An *elastic rectangle* $\mathcal{E}$ is a family of rectangles specified by a quintuplet (p, α, H, W, Q) where p is a point that is a corner of all rectangles in $\mathcal{E}$, α is the area of any rectangle in $\mathcal{E}$ (that is all rectangles in $\mathcal{E}$ have the same area), $H = [h^{\min}, h^{\max}]$ is the range of the height of the rectangles, $W = [w^{\min}, w^{\max}]$ is the range of the width, and $Q \subseteq \{1, 2, 3, 4\}$ is a set of possible positions of p allowed in the family. The value of the position is 1 when p is a bottom left corner, 2 when p is a top left corner, 3 when p is a top right corner, and 4 when p is a bottom right corner.

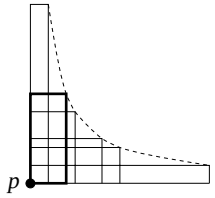

Fig. 3. An elastic rectangle.

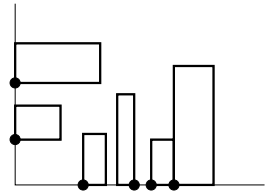

Fig. 4. Two-axis case.

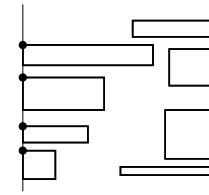

Fig. 5. Two-parallel lines case.

We use the notation $p(\mathcal{E})$, $\alpha(\mathcal{E})$, $H(\mathcal{E})$, $W(\mathcal{E})$, and $Q(\mathcal{E})$ for the parameters of an elastic rectangle $\mathcal{E}$. The point $p(\mathcal{E})$ will be called the *anchor* of $\mathcal{E}$.

When Q is a singleton, the family of rectangles $\mathcal{E}$ is described by a hyperbola segment tracing out the locus of the corner of the elastic rectangle opposite p. Figure 3 shows an elastic rectangle with $Q(\mathcal{E}) = \{1\}$, with the hyperbola as a dashed curve. Note that a discrete set of the rectangles in $\mathcal{E}$ are drawn but we are (for now) considering $\mathcal{E}$ to be continuous.

A *realization* of an elastic rectangle $\mathcal{E}$, denoted E, is a single rectangle from the family—i.e. we must choose a valid height, width, and corner to place at $p(\mathcal{E})$. We say that we *fix* an elastic rectangle when we choose one realization from its family. A *realization* of a set of elastic rectangles means a realization of each elastic rectangle in the set. Such a realization is *good* if the interiors of the chosen rectangles are pairwise disjoint. Given a set of elastic rectangles as input, the *elastic labeling problem* is to find a good realization for the set.

A *one-corner* elastic rectangle is an elastic rectangle with $|Q| = 1$. The *one-corner elastic labeling problem* is the special case of the elastic labeling problem where all the elastic rectangles are one-corner elastic rectangles. This problem is known to be NP-hard [5]. Our paper is about tractable special cases.

3 Two-Axis Labeling Problem

The *two-axis labeling problem* is a variation of the one-corner labeling problem in which the points lie on the positive x and y axes, and the labels lie in the

first quadrant. Figure 4 depicts a good realization for an instance of the two-axis labeling problem. From [6] we have the following results.

Theorem 1. [6] *There is an algorithm for the two-axis labeling problem with running time $O(nm)$, where n and m are the numbers of rectangles in the x and y axes, respectively.*

Later on, in section 5, we will need not only Theorem 1, but the following stronger result. The algorithm uses dynamic programming. Let $\mathcal{S}_{i,j}$ be the subproblem of finding a good realization for the first i elastic rectangles in the x-axis and the first j elastic rectangles in the y-axis. Subproblem $\mathcal{S}_{i,j}$ may have many solutions. What we care about is the height and width of the smallest box enclosing each good realization. The algorithm captures all minimal height-width combinations for this smallest enclosing box. Any good realization in this infinite set has all the elastic rectangles fixed except the last one along the x-axis or the last one along the y-axis. This permits a concise representation of the set of solutions.

4 Two-Parallel Lines Labeling Problem

In this section we give a good algorithm for the *two-parallel lines labeling problem*: find a good realization for a set of one-corner elastic labels that lie between two vertical lines L_1 and L_2, with their anchors lying on these lines.

Figure 5 shows a good realization for an instance of this problem. Figure 6 shows some of the ways that the elastic rectangles can interact with each other.

We use a greedy approach, adding elastic rectangles from bottom to top. We keep the three most recently added rectangles elastic, and fix the earlier ones. We first describe the bottom-to-top ordering of the anchors, and then discuss the general step of the algorithm.

We consider the anchors in each line L_1 and L_2 in order from bottom to top. These two orderings are merged as follows: Initially, let p and q be the bottommost anchors of lines L_1 and L_2, respectively—we take them in either order. More generally, let p and q be the most recently considered anchors of L_1 and L_2, respectively. If p is below q add the next anchor above p on L_1. Otherwise, add the next anchor above q on L_2. If p and q have the same y-coordinate, make either choice. We call this the *smallest-moves-up ordering*. For simplicity, add two dummy anchors with associated 0×0 rectangles at the top of the lines L_1 and L_2 to ensure that all anchors get considered in this ordering. See Figure 7 for an example.

We now describe the general step of the algorithm as it adds elastic rectangles based on the smallest-moves-up ordering of anchors. Three of the added elastic rectangles remain elastic and the others are fixed. If r and s are the current topmost anchors that have been added in each of the two lines, and $y(r) \geq y(s)$, and t is the anchor below r on its line, then the rectangles that remain elastic are those associated with r, s, and t. Because r was added to the ordering to increase the smaller y-coordinate, thus $y(t) \leq y(s)$, and the points form what we

call a *topmost C-shape*. We call r, s, and t the top, middle, and bottom points of the C-shape respectively. In Figure 8, the anchors in bold are in a C-shape. Note that the anchors that form the topmost C-shape are not necessarily the three topmost anchors nor the three most recently added anchors —in particular, though r and s are topmost, t may not be (see the hollow anchor in Figure 8).

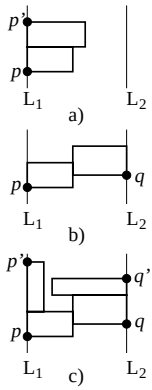

Fig. 6. Facing rectangles.

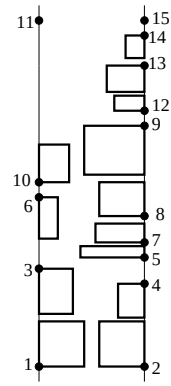

Fig. 7. Smallest-moves-up ordering.

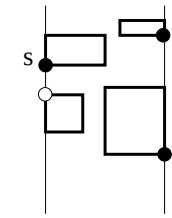

Fig. 8. C-shape.

Consider what happens when we add one more elastic rectangle. Its anchor, q, enters the topmost C-shape. Another one leaves, and we must fix its elastic rectangle. Since the anchor q must be next in smallest-moves-up ordering it must be above s on its line. If q lies above r the new topmost C-shape consists of q, r, s, and we must fix the elastic rectangle at t. If q lies below r then the new topmost C-shape consists of r, q, t, and we must fix the elastic rectangle anchored at s.

We will find it notationally more convenient to have one symbol for the elastic rectangle that leaves the top-most C-shape. We will therefore treat the *previous* step of the algorithm, the one where r, s, t became the top-most C-shape. Let u be the anchor that just left the top-most C-shape. Then u lies under s in its line. If u is above t then s is the newly added point and r, u, t is the old C-shape; and if u is below t then r is the newly added point and s, t, u is the old C-shape.

Let $\mathcal{R}$, $\mathcal{S}$, $\mathcal{T}$, $\mathcal{U}$ be the elastic rectangles at r, s, t, u, respectively. We must fix $\mathcal{U}$. A case-by-case analysis seems necessary. We will be able to maintain the structure that the bottom and middle anchors of the top-most C-shape are bottom anchors of their elastic rectangles (or top anchors of fixed rectangles). This simplifies our task in that we may assume u and t are bottom anchors. On the other hand, we must now be sure to fix $\mathcal{S}$ if s is a top anchor. We will treat the four cases where r and s are top/bottom anchors. We can ignore the relative vertical order of u and t because we can use the following lemma to reduce to the case where the realizations of $\mathcal{U}$ and $\mathcal{T}$ lie side by side.

Lemma 1. *Suppose we have a top-most C-shape where the middle and bottom points are bottom anchors of their elastic rectangles or top anchors of fixed*

rectangles. If the minimum height realization, U, of the bottom-most elastic rectangle, $\mathcal{U}$, lies below the middle anchor of the C-shape, then we can fix $\mathcal{U}$ to U without precluding a good realization for the whole set.

Proof. Outline. We show that if there is a good realization for the whole set that uses the rectangles fixed up to now, then there is a good realization for the whole set using the rectangles fixed up to now and using U.

In all cases we will concentrate on the horizontal line, l_s, going through the point s because we now have full information about the two, three, or four elastic rectangles that are *active below* l_s—i.e., that have realizations intersecting the region below this line. Two elastic rectangles *interact* if they have realizations that intersect. In all cases we must fix $\mathcal{U}$. Its realization must lie below l_s and so—of all the rectangles not yet fixed—it interacts only with the other one, two, or three elastic rectangles that are active below l_s. Because some of these elastic rectangles may interact with as-yet-unseen elastic rectangles, we must fix $\mathcal{U}$ in such a way that we limit as little as possible the choices for these other elastic rectangles. We note here, and later will take for granted, that in choosing realizations for elastic rectangles, we must take care to avoid the rectangles that have already been fixed.

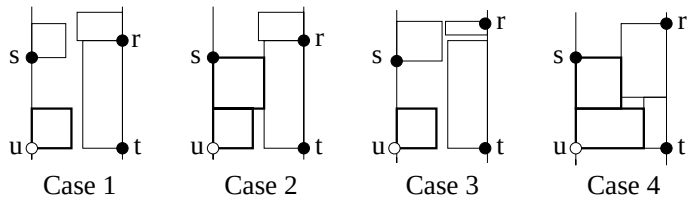

Fig. 9. Fixing elastic rectangle $\mathcal{U}$: the 4 cases.

Case 1. Both r and s are bottom anchors. See Figure 9. We must fix $\mathcal{U}$. $\mathcal{U}$ interacts only with $\mathcal{T}$, since no other elastic rectangle is active below l_s. Because the realizations of $\mathcal{U}$ and $\mathcal{T}$ must lie side by side, we can clearly fix $U \in \mathcal{U}$ of minimum width. To be more formal, if there is a good realization for the whole set using the rectangles fixed up to now, we can replace the realization of $\mathcal{U}$ by U and still have a good realization.

Case 2. r is a bottom anchor. s is a top anchor. See Figure 9. We must fix $\mathcal{U}$ and $\mathcal{S}$. $\mathcal{U}$ interacts only with $\mathcal{T}$ and $\mathcal{S}$, since no other elastic rectangle is active below l_s. If we can find a good realization consisting of $U \in \mathcal{U}$, $T \in \mathcal{T}$ and $S \in \mathcal{S}$ such that all three rectangles lie below the line l_s, then we should use them; fix $\mathcal{U}$ to U and $\mathcal{S}$ to S. ($\mathcal{T}$ can be fixed to T by applying the simplification of Lemma 1.) Otherwise, in any good realization, the rectangle chosen for $\mathcal{T}$ must stick up above l_s, and we fix $\mathcal{U}$ and $\mathcal{S}$ of minimum total width. We claim that this does not preclude a good realization of the whole set:

Claim. If there is a good realization G for the whole set using the rectangles fixed up to now, then we can replace the realizations of $\mathcal{U}$ and $\mathcal{S}$ as above, and still have a good realization.

Proof. Replacing in G the realizations of $\mathcal{U}$, $\mathcal{S}$, and $\mathcal{T}$ to ones that lie below l_s leaves a good realization. In the other case, the realization of $\mathcal{T}$ in G must stick above l_s, and thus must lie beside the realizations of $\mathcal{U}$ and $\mathcal{S}$, so replacing the realizations of $\mathcal{U}$ and $\mathcal{S}$ by ones of minimum total width leaves a good realization.

We will treat cases 3 and 4 together:

Case 3 [and 4]. r is a top anchor. s is a bottom anchor [a top anchor]. See Figure 9. We must fix $\mathcal{U}$ [and $\mathcal{S}$]. $\mathcal{U}$ interacts only with $\mathcal{T}$ and $\mathcal{R}$ [and $\mathcal{S}$] since no other elastic rectangle is active below l_s. If $\mathcal{T}$ must stick up above l_s, i.e. there is no good realization of $\mathcal{U}$ and $\mathcal{T}$ [and $\mathcal{S}$] lying below l_s, then we are back to the same arguments as in case 1 [case 2], and fix $\mathcal{U}$ [and $\mathcal{S}$] of minimum [total] width. As in case 1 [case 2], we can prove that this does not preclude a good realization for the whole set.

Assume then that there is a good realization of $\mathcal{U}$ and $\mathcal{T}$ [and $\mathcal{S}$] lying below l_s. The complication now is that such realizations are not all equally good. In particular, we may wish to use a realization of $\mathcal{R}$ that goes below l_s. If we do, we are surely better off (with respect to elastic rectangles above l_s) taking a realization $R \in \mathcal{R}$ of maximum height. Thus we find the maximum height realization of $\mathcal{R}$ that permits good realizations U, T [and S] of $\mathcal{U}$, $\mathcal{T}$ [and $\mathcal{S}$]. We fix $\mathcal{U}$ to U [and $\mathcal{S}$ to S]. We claim that this does not preclude a good realization of the whole set:

Claim. If there is a good realization G for the whole set using the rectangles fixed up to now, then there is a good realization for the whole set using the rectangles fixed up to now and using the realizations for $\mathcal{U}$ [and $\mathcal{S}$] specified above.

Proof. Let R' be the realization of $\mathcal{R}$ in G. If R' lies above l_s then we can replace the realizations of $\mathcal{U}$, $\mathcal{T}$ [and $\mathcal{S}$]) in G by the ones specified above. Otherwise R' goes below l_s and we replace the realizations of $\mathcal{U}$, $\mathcal{T}$, and $\mathcal{R}$ [and $\mathcal{S}$] in G by the ones specified above. Note that replacing R' by a taller skinnier rectangle does not interfere with any other rectangles above l_s.

This completes the description of one step of the algorithm. See [7] for details on how to implement these four cases in constant time on a real-RAM model of computation by calculating intersections of hyperbolas.

We have proved the correctness of one step of the algorithm. Correctness of the whole algorithm follows by induction. Since each step takes constant time, the whole algorithm runs in linear time. We summarize with the following theorem.

Theorem 2. *There is an algorithm for the two-parallel lines labeling problem with running time $O(n)$ where n is the number of points, and we assume they are given in sorted order along the two lines.*

We observe (since we will need it later) that the algorithm does not actually rely on the fact that the area of an elastic label is constant, but only on the fact that the family of rectangles is captured by a hyperbola segment tracing the locus of the corner of the rectangle opposite the anchor.

5 Rectangular Perimeter Labeling Problem

In this section we combine our algorithms for the two-axis and two-parallel lines labeling problems to solve the general *rectangle perimeter labeling problem*: find a good realization of a set of one-corner elastic labels that lie inside a *boundary rectangle* P with their anchors on the perimeter of P. Figure 2 shows a good realization for an instance of this problem.

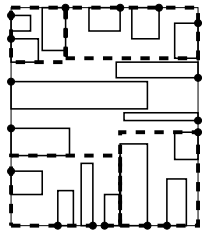

Fig. 10. A corridor.

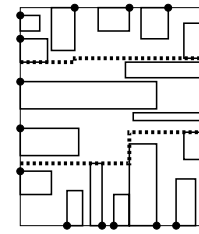

Fig. 11. Jagged horizontal lines.

We will show that it suffices to consider a polynomial number of decompositions of the elastic rectangles into regions in which labels from only two sides of the boundary rectangle compete for space. (It is intuitively clear, for example, that near a corner of the map only the labels from two sides of the map are relevant.) We can then apply our algorithms for the two-axis and two-parallel lines cases in these regions. The dashed lines in Figure 10 illustrate one such decomposition. We describe the decomposition in subsection 5.1, an algorithm to obtain a good realization for a given decomposition in subsection 5.2, and the complete algorithm in subsection 5.3.

5.1 Decomposition

Suppose we have a good realization for the elastic rectangles inside P. Let k be a corner of P. A *corner block* at corner k is a rectangle C contained in P with one corner at k and with the property that any rectangle in the good realization is either completely outside the interior of C or is completely inside C. We disallow $C = P$. We allow C to have height or width zero, but in this case consider C to be a one-dimensional object whose interior is an open line segment.

The idea is that inside a corner block we have an instance of the two-axis labeling problem. What about the area outside the corner blocks?

Two corner blocks *touch* if they are at adjacent corners of P and have some common boundary but disjoint interiors. Thus two touching corner blocks completely cover one side of P (see the two dashed rectangles at the bottom of P in Figure 10). We define a *corridor* to be four corner blocks with disjoint interiors that form two touching pairs covering two opposite sides of P. The area outside the four corner blocks of a corridor touches P at most in two opposite sides and gives rise to an instance of the two-parallel lines problem. Note that it is possible for this area to touch P only on one side, or even on no sides.

For simplicity we define a one-corner elastic rectangle in the interior of P to be *left-base*, *right-base*, *bottom-base*, or *top-base* if its anchor is in the left, right, bottom or top side of P respectively.

Lemma 2. *Any good realization of the elastic rectangles inside P has a corridor.*

Proof. We first prove that there is a jagged horizontal line or a jagged vertical line in the free space between the rectangles. See Fig. 11 for examples. To be more precise, a jagged horizontal line consists of a horizontal line segment with one endpoint on the left side of P, a horizontal line segment with one endpoint on the right side of P, and a vertical line segment (possibly empty) joining their other endpoints. A point in P is *free* if it is either in the interior of P but outside the interiors of the rectangles, or on the perimeter of P but outside the open line segments where the rectangles meet the perimeter of P.

Let T be a top-base rectangle of maximum height. Attempt to construct a free jagged vertical line by walking down one side of T and then continuing this line downward. If it reaches the bottom of P without hitting another rectangle we are done. If we are blocked by a bottom-base rectangle we can walk around it to get what we want. Otherwise we hit a left- or right-base rectangle. Suppose without loss of generality that we hit a left-base rectangle S. Walk along this side of S to the right. If by continuing this line we hit the other side of P, or we hit a right-base rectangle, we get a free jagged horizontal line. Otherwise we hit a bottom-base rectangle B. Note that we cannot hit a top-base rectangle since T has maximum height. Walking down B completes our free jagged vertical line.

It remains to prove that a free jagged line implies a corridor. We will show that if there is a free jagged horizontal line then there is a pair of touching corner blocks covering the top of P and a pair of touching corner blocks covering the bottom of P. Let h be a free jagged horizontal line that is maximal in the sense that the region above it is minimal by containment. We claim that h provides the outline of two touching corner blocks that cover the top of P. Let v be the vertical segment of h. The fact that we cannot shift v over to decrease the area above h means that some rectangle above h touches v. If this is a top rectangle, we get our corner blocks; if it is a left or right rectangle, we get a higher free jagged horizontal line, contradicting the maximality of h.

So far, we have argued about a nice decomposition assuming that we *have* a good realization. How can we turn these ideas around to help us *find* a good realization? We would like to enumerate all the possible corridors, and for each one solve four instances of the two-axis labeling problem for the four corner

blocks and one instance of the two-parallel lines labeling problem for the area outside the corner blocks.

Since in principle there are an infinite number of possible corridors we need to discretize them somehow. What we will do is specify the set of elastic rectangles that are to go into each of the four corner blocks, keeping in mind that either all top rectangles and all bottom rectangles go into corner blocks, or else all left and all right rectangles go into corner blocks. We call this a *corridor partition* of the elastic rectangles. The number of corridor partitions is $O(n^6)$.

What remains is to solve the problem: given a corridor partition of the elastic rectangles, is there a good realization for the elastic rectangles that respects the given partition? We call this the *corridor partition realization problem*, and solve it in the next subsection.

5.2 Corridor Partition Realization Problem

In this section we give an algorithm for the corridor partition realization problem described in the last paragraph of the previous section. The idea is straightforward: use the two-axis labeling algorithm to find the best realization for the elastic rectangles assigned to each corner block, and then use the two-parallel lines algorithm to finish up. The one complication is that there is no best realization for the elastic rectangles assigned to a corner block.

We need to look more closely at the solutions given by the two-axis labeling algorithm. Let k be a corner of P and let S be the set of elastic rectangles assigned by the corridor partition to the corner block at k. Let $\mathcal{R}_1$ be the elastic rectangle in S furthest from k on one axis and $\mathcal{R}_2$ be the elastic rectangle in S furthest from k on the other axis. The two-axis labeling algorithm finds all good realization of S for which the smallest enclosing box is minimal. Any realization in this infinite set is captured by two solutions, one where all the elastic rectangles in S except $\mathcal{R}_1$ are fixed and $\mathcal{R}_1$ remains elastic, and the other where all the elastic rectangles in S except $\mathcal{R}_2$ are fixed and $\mathcal{R}_2$ remains elastic. In either of these two cases we have what we call an *elastic block*: a family of rectangles with one corner at k and the opposite corner on a segment of a hyperbola. An elastic block is in general not an elastic rectangle since it need not have constant area. However, as noted in section 4, the two-parallel lines labeling algorithm works just as well on elastic blocks as on elastic labels. Since each of the 4 corners has 2 elastic blocks as possible solutions, we end up with 8 instances of the two-parallel lines labeling problem to solve each taking $O(n)$ time.

Supposing that we have available the results of running the two-axis labeling algorithm at each corner of P, this algorithm for the corridor partition realization problem takes $O(n)$ time.

5.3 Algorithm

Here is our algorithm for the rectangle perimeter labeling problem: Sort the anchors along each side of the boundary rectangle P. Run the two-axis labeling algorithm once in each corner of P, using all the elastic rectangles on each axis,

at a cost of $O(n^2)$. Then, for each of the $O(n^6)$ corridor partitions, run the corridor partition realization algorithm, using $O(n)$ time. The algorithm has running time $O(n^7)$.

This running time should clearly be improved. In the remainder of this section we outline a way to achieve $O(n^4)$ running time. We believe that further improvements are possible using the fundamental ingredients we have provided so far, namely the algorithms for the two-axis and two-parallel lines versions, and the decomposition into a corridor.

We begin by running the two-axis labeling algorithm in each corner of the boundary rectangle, so we have all the corner solutions available to us. This takes $O(n^2)$ time. We will describe how to search for pairs of touching corner blocks covering the bottom and top sides of P, working from the bottom to the top of P. Our brute force algorithm tried all $O(n^3)$ possible partitions of the elastic rectangles into two potential touching corner blocks covering the bottom of P. We gain some efficiency by concentrated on pairs of touching corner blocks that are minimal by containment of their union. We claim that there are $O(n^2)$ possibilities and that we can find them in $O(n^2)$ time.

For each of these partitions of the elastic rectangles, we solve in each corner to obtain elastic blocks, and then proceed to run the two-parallel lines algorithm up the sides of P. We exploit the fact that this algorithm tries to keep the elastic rectangles down as much as possible. At each of the $O(n)$ steps of this algorithm we check if the remaining elastic rectangles can be formed into two touching corner blocks covering the top of P. We are looking for a pair of corner blocks covering the top of P whose heights are as close as possible to equal. This is what allows us to restrict attention to consecutive pairs of points that arise in the $O(n)$ steps of the two-parallel lines algorithm. We claim that the search for the pair of corner blocks covering the top of P can be done in $O(n)$ time.

The algorithm can probably be improved further. One possible approach is to enhance the two-parallel lines algorithm to capture all solutions, rather than just the "greedy" solution, with the hope of avoiding the repeated calls to this subroutine. Another possibility is to search for the decomposition using some kind of binary search rather than the brute-force search we employed.

6 Conclusions

In this paper we addressed the problem of choosing elastic labels to attach to points on the boundary of a rectangular map. We gave an algorithm to solve this rectangle perimeter labeling problem in polynomial time. Even with improvements, our algorithm, with a running time of $O(n^4)$, is impractical. However, given that so many problems in map labeling are NP-hard, any polynomial time algorithm is a success. We do think that a faster algorithm is possible, and that it must be based on the results we have presented: efficient algorithms for the two special cases where points lie on only two sides of the map, and a basic decomposition result to reduce to these special cases.

The rectangle perimeter labeling problem has potential applications for the display of dynamic maps on the web and in Geographical Information Systems (GIS). When a rectangular window displays only some portion of a map, and that portion can change as the user scrolls and zooms, we need information about what is beyond the boundary of the map currently visible in the window. Text labels around the perimeter of the map in the window would seem useful.

Since font sizes are discrete, and text labels are displayed using some discrete number of lines, a discrete version of the rectangle perimeter problem may be more relevant than the continuous version. Our methods carry over to this case. See [7]. Also in [7] is a solution to the (much easier) variation of the rectangle perimeter labeling problem where the labels lie on the outside of the perimeter of the boundary rectangle.

One natural open problem is the two-corner version of the rectangle perimeter labeling problem, where, for example, a point on the bottom of the boundary rectangle can have its label above and right, or above and left.

References

1. B.Chazelle et al. Application challenges to computational geometry: CG impact task force report. *Technical Report TR-521-96*, Princeton Univ., April 1996.
2. J. Christensen, J. Marks, and S. Shieber. An empirical study of algorithms for point feature label placement. *ACM Transactions on Graphics.* **14** 3 (1995), 203–232.
3. J. Christensen, S. Friedman, J. Marks, and S. Shieber. Empirical testing of algorithms for variable-sized label placement. *Proc. 13th ACM Symp. on Comp. Geom.* (1997), 415–417.
4. M. Formann and F. Wagner. A packing problem with applications in lettering of maps. In *Proc. 7th ACM Symp. on Comp. Geom.* (1991) 281–288.
5. C. Iturriaga and A. Lubiw. NP-hardness of some map labeling problems. *Technical Report CS-97-18.* University of Waterloo, 1997.
6. C. Iturriaga and A. Lubiw. Elastic labels: the two-axis case. In G. Di Battista editor, *Graph Drawing (Proc. GD'97).* vol. 1353 of *LNCS.* Springer-Verlag. (1998), 181–192.
7. C. Iturriaga. Map Labeling Problems, Ph.D. Thesis University of Waterloo, 1999.
8. T. Kato and H. Imai. The NP-completeness of the character placement problem of 2 or 3 degrees of freedom. *Record of Joint Conference of Electrical and Electronic engineers in Kyushu.* (1988) 11–18. In Japanese.
9. D. Knuth and A. Raghunathan. The problem of compatible representatives. *SIAM Disc. Math.* **5** 3 (1992), 422–427.
10. L. Kučera, K. Mehlhorn, B. Preis, and E. Schwarzenecker. Exact algorithms for a geometric packing problem. In *Proc. 10th Symp. Theoret. Aspects Comput. Sci.*, 665 of LNCS, 317–322. Springer-Verlag, 1993.
11. J. Marks and S. Shieber. The computational complexity of cartographic label placement. *Technical Report CRCT-05-91.* Harvard University, 1991.
12. F. Wagner and A. Wolff. A practical map labeling algorithm. *Computational Geometry: Theory and Applications.* (1997) 387–404.
13. A. Wolff and T. Strijk. A map labeling bibliography, 1996. http://www.inf.fu-berlin.de/map-labeling/papers.html.