Map Labeling and Its Generalizations

Srinivas Doddi,* Madhav V. Marathe,† Andy Mirzaian,‡ Bernard M.E. Moret,§ and Binhai Zhu¶

Abstract

Map labeling is of fundamental importance in cartography and geographical information systems and is one of the areas targeted for research by the ACM Computational Geometry Impact Task Force. Previous work on map labeling has focused on the problem of placing maximal uniform, axis-aligned, disjoint rectangles on the plane so that each point feature to be labeled lies at the corner of one rectangle. Here, we consider a number of variants of the map labeling problem.

We obtain three general types of results. First, we devise constant-factor polynomial-time approximation algorithms for labeling point features by rectangular labels, where the feature may lie anywhere on the boundary of its label region and where labeling rectangles may be placed in any orientation. These results generalize to the case of elliptical labels. Secondly, we consider the problem of labeling a map consisting of disjoint rectilinear line segments. We obtain constant-factor polynomial-time approximation algorithms for the general problem and an optimal algorithm for the special case where all segments are horizontal. Finally, we formulate a bicriteria version of the map-labeling problem and provide bicriteria polynomial-time approximation schemes for a number of such problems.

Keywords: Approximation algorithms, map labeling, $\mathcal{NP}$-hardness

1 Introduction

Automatic map-making is an important part of Geographic Information Systems (GIS). Although nearly two decades of development have led to some good map-making algorithms, cartographic knowledge and experience remain critical to the production of good maps: "the craft of making maps is still an indispensable ingredient" [BC94]. Map labeling has been targeted by the ACM Computational Geometry Task Force [CGI96] as one of the important areas of research in Discrete Computational Geometry. As pointed out in [CMS95], applications in cartography require three different label-placement tasks: (i) labeling area features (such as countries and oceans); (ii) labeling line (segment) features (such as rivers and roads); and (iii) labeling point features (such as cities and mountain peaks). An efficient algorithm must solve these three label-placement tasks simultaneously. Note that all three tasks share a combinatorial aspect: labels must not overlap; as remarked in [CMS95], this aspect of label placement is independent of the nature of the features being labeled and is perhaps the most basic problem to solve for automating label placement. In this paper we focus on generating non-overlapping label placements for point features and rectilinear (axis-parallel) segment features, with possible extensions sketched or suggested.

Cartographic labeling requires a cartographer to consider many conflicting criteria for labeling the maps, such as location, orientation, shape, size, and typography for each label. In a seminal paper in this

*Dept. of Computer Science, University of New Mexico, Albuquerque, NM 87131, and Los Alamos National Laboratory, srinu@xdiv.lanl.gov; research supported by Los Alamos National Lab.

†Los Alamos National Laboratory, MS K990, Los Alamos NM 87545, madhav@c3.lanl.gov. The work is supported by the Department of Energy under Contract W-7405-ENG-36.

‡Dept. of Computer Science, York University, Toronto, Canada H3A 2A7, andy@cs.yorku.ca; research supported in part by NSERC.

§Dept. of Computer Science, University of New Mexico, Albuquerque, NM 87131, moret@cs.unm.edu; research supported in part by ONR contract N00014-92-C-2144.

¶Dept. of Computer Science, City University of Hong Kong, and Los Alamos National Laboratory, bhz@c3.lanl.gov; research supported by Los Alamos National Lab.

area, Imhof [Im75] illustrates these goals by giving 100 examples of good and bad labeling decisions. As pointed out in [MS91], the following concerns are of particular importance: (i) the degree to which labels overlap with each other and obscure cartographic features; (ii) the degree to which labels are unambiguously and clearly associated with the features they identify; (iii) *a priori* preferences among a canonical set of potential label positions; and (iv) the number of point features left unlabeled. Legibility may take precedence over aesthetic placement, especially for technical maps where every feature must be labeled [FW91].

These considerations lead us to define the *general point-feature map-labeling* problem. An instance of this problem consists of a set of point features and a set of constraints (such as permissible amount of overlap) for placing labels. The goal of the problem is to label each feature so as to satisfy the constraints. On the theoretical side, Formann and Wagner [FW91] studied the problem of labeling a set of n points such that each point is assigned an axis-aligned labeling rectangle, each rectangle is placed so that one of its corners is the point feature it labels, all rectangles have the same size, and the size of the rectangles is maximized. They proved that this problem is $\mathcal{NP}$-complete and that, unless $\mathcal{P} = \mathcal{NP}$, no polynomial-time approximation algorithm can do better than 50% of optimal; moreover, they presented an $O(n \log n)$ time approximation algorithm achieving this bound. Wagner [Wag94] then proved that an approximation algorithm that achieves this bound must take $\Omega(n \log n)$ time. More recently, Wagner and Wolff [WW95, WF95] introduced some heuristics that appear to perform well on small problems. In general, map labeling appears to be a hard problem since it is closely related to the $\mathcal{NP}$-hard *independent set* and *kSAT* problems [KR92]. Other researchers have built automated map-labeling systems since the early 70's, typically using a combination of heuristics such as mathematical programming, gradient descent, etc.; a comprehensive survey and list of references can be found in [CMS93]. Given that map-labeling can be thought of as attempting to meet a set of rules [Im75], several researchers have also attempted to solve the problem using techniques from artificial intelligence and logic programming (see, for instance, [Jo89]); the aforementioned survey also discusses these attempts as does [DF92].

2 Our Results

We study several variants of the general point-feature map-labeling problem. Our results significantly extend and generalize those of [CMS93, FW91] on the complexity and approximability of the map-labeling problem. We consider the following two generalizations of the problem: (i) allowing the labels to be rectangular or elliptical while removing any restriction on their orientation; and (ii) allowing a feature to be anywhere on the boundary of its label region (rather than at a vertex of its labeling rectangle). These generalizations reflect the fact that, in many of today's electronic maps, labels are not restricted to textual matter, but may also be graphical (although even purely textual labels have long been placed non-horizontally in maps). In all of these cases, we retain the objective function proposed by Wagner and his colleagues, namely the size of the uniform labeling areas.

We also note that all previous research in map labeling deals with how to label sites (points), while we mentioned earlier that linear features (rivers, streets, etc.) often need their own labels. In practice, we often need to label rectilinear line segments, such as city streets or VLSI circuits. The labeling area associated with such segments is a rectangle, the length of which is the length of the corresponding segment and the width of which is to be maximized (subject to the constraints of the problem); this area can be placed in one of three positions: above or below a horizontal segment (naturally, left or right of a vertical segment) or across and at the middle of it (i.e., making the segment the mid-edge of the labeling area).

Given the complexity of the map-labeling problem, we investigate the existence of polynomial-time approximation algorithms. We present the first polynomial-time approximation algorithms and approximation schemes for a number of variants of the generalized map-labeling problem. Recall that an approximation algorithm for a maximization problem Π provides a *performance guarantee* of $0 < \rho \leq 1$ if, for every instance I of Π, the solution value returned by the approximation algorithm is within a factor ρ of the optimal value for I. We obtain the following results for the generalized map-labeling problem:

- For labeling a map with uniform squares (in arbitrary orientations), we provide a polynomial-time approximation algorithm with a performance guarantee of $8\sqrt{2}/\cos(\pi/5)$.
- For labeling a map with uniform circles, we provide a polynomial-time approximation algorithm with a performance guarantee of $4(2+\sqrt{3})$.
- For labeling a map with uniform regular polygons, we prove that there exists a constant-factor, polynomial-time approximation algorithm for each type of regular polygon.
- For labeling rectilinear segments with rectangles of uniform width, we prove that there exists a polynomial-time approximation algorithm with a performance guarantee of 2. We also show that the problem can be solved exactly in $\Theta(n \log n)$ time when all segments are horizontal.

Our approximation algorithms for labeling point features are very efficient and easily implemented; all run in $O(n \log n)$ time with small constants.

Going back to the criteria of [MS91], we note that one criterion listed is the number of features left unlabeled—we all have encountered maps with unlabeled features. Yet all the algorithms mentioned above label every feature. By allowing a small number of features to remain unlabeled, we may be able to better label the other features. This approach introduces a trade-off between the quality (size) of the labels placed and the number of unlabeled features. We present a bicriteria framework in which, for n features and any given ϵ, we must find a placement of at least $(1-\epsilon)\cdot n$ labels, each of size at least $(1-c\cdot\epsilon)$ times the optimal labels, for some positive constant c. We present a simple, yet very general technique, based on discretization of the map and its labels, to construct a polynomial-time approximation scheme for this problem and its variants. Since it has been shown in [FW91] that the map-labeling problem cannot be approximated within a factor of 2 unless $\mathcal{P} = \mathcal{NP}$, our bicriteria framework offers one way to overcome the limitations imposed by labeling every feature.

3 Preliminaries

We define formally our two problems for labeling point features and briefly discuss a related tractable problem that we shall use in our approximation algorithms. We give definitions for the decision versions of our problems, in the interest of clarifying the formal computational complexity of these problems; the optimization version is trivially formulated from the decision version.

Definition 1. An instance of the problem of *Map Labeling with Uniform Squares* (MLUS) consists of n points (features) in the plane and a positive integer bound B. The question is whether there exists a placement for n squares, each of side B, such that

- no two squares intersect; and
- each point lies on a square and no two points lie on the same square.

Observe that the solution to the MLUS optimization problem diverges if we have four or fewer points: with four points (in general position), each point can have associated with it an infinite square. Since an infinite square is effectively a quarter of the plane, divergent solutions cannot exist for five or more distinct points.

Definition 2. An instance of the problem of *Map Labeling with Uniform Circles* (MLUC) is given by a set of n points (features) in the plane and a positive integer bound B. The question is whether there exists a placement for n circles, each of radius B, such that

- no two circles intersect; and
- each point lies on a circle and no two points lie on the same circle.

Observe that the solution to the MLUC optimization problem diverges if we have just two points, but must be finite for three or more points, since an infinite circle is effectively a half-plane.

We shall make extensive use of the following well-solved problem.

Definition 3. Given a set S of points, the *k-diameter* of any subset of k points is the maximum distance between any two points in the subset. The *minimum k-diameter* of S, denoted $D_k(S)$, is the smallest value of the k-diameter among all subsets of S of size k.

How to compute the minimum k-diameter was studied by [EE94, DLSS]; they gave an algorithm that returns the value in $O(n \log n)$ time. We make some simple observations about the minimum k-diameter.

Lemma 1. $D_1 \leq D_2 \leq \ldots \leq D_{n-1} \leq D_n$.

Lemma 2. Let $\alpha > 2$ be some constant and draw a circle of radius $\frac{D_k}{\alpha}$ centered at some point $p_i \in S$; then this circle contains at most $k - 1$ points.

Proof: The maximum distance between any two points inside the circle is at most the diameter of the circle, which is $2D_k/\alpha < D_k$. If the circle were to contain at least k points, then these k points would constitute a subset of size k with diameter less than D_k contradicting the definition of D_k.

4 Map Labeling With Uniform Squares

Let L^* denote the size of each square in the optimal solution of the problem MLUS.

Lemma 3. A set of five points with diameter D_5 has optimal labeling squares of size at most $D_5/\cos(\pi/5)$.

Proof: Call the five points a, b, c, d, and e, and assume that the diameter D_5 occurs between points a and b. We then can place a regular pentagon P' of side D_5 that circumscribes all five points. Let L be the size of the largest labeling square for the original five points and L' that for the five vertices of the regular pentagon P'; obviously, we have $L \leq L'$. Symmetry immediately implies that the largest labeling squares for the vertices of P' are arranged around P' in a ring such that the inner side of each labeling square has as its midpoint the vertex of P' that it labels. These inner sides form another regular pentagon; the ratio of the sides of the two pentagons is easily seen to be $\cos(\pi/5)$. Thus L' is $1/\cos(\pi/5)$ times the edge length of P', which is D_5 by construction; thus we have $L \leq L' = D_5/\cos(\pi/5)$.

Theorem 1. A set S of point has maximum labeling squares of size at most $D_5(S)/\cos(\pi/5)$.

Denote the distance between two points $p_i, p_j \in S$ by d_{ij} and denote by C_i the circle centered at point $p_i \in S$ with radius $\frac{D_5(S)}{\alpha}$. By Lemma 2, C_i contains at most four points of S (counting its center). Denote by n_i the number of points of S within C_i; in the following, we assume, without loss of generality, that we have $n_i = 4$. We shall place a square of side $\frac{D_5(S)}{4\alpha\sqrt{2}}$ at each point p_i; note that the largest dimension of this square is its diagonal, which has length $\frac{D_5(S)}{4\alpha}$.

We now proceed to describe the algorithm; since the algorithm effectively places the squares, we state the main result as a theorem and prove it constructively by providing the algorithm. We then analyze the running time of the algorithm.

Lemma 4. Given a set S of points to be labeled with uniform squares, there exists a set of square labels, each of size $L' \geq \frac{D_5(S)}{4\alpha\sqrt{2}}$.

Proof: Our proof is a recursive procedure that labels each point: we select some point p_i and show how to place a square of size L' touching p_i. Assume, without loss of generality, that we have $n_i = 4$ and denote the other three points in C_i by p_j, p_k and p_l. Consider the circle C_i' centered at p_i with radius $\frac{D_5(S)}{2\alpha}$—half the radius of C_i. We distinguish four cases, depending on how many of p_j, p_k, and p_l fall within C_i'.

1. *C_i' only contains p_i.* This case is easy. We can label p_i with a square in any arbitrary position: since p_i is at least two diagonals away from any of its neighbors, its labeling square cannot affect the positioning of labeling squares for its neighbors. Our procedure thus removes p_i, recursively labels all remaining points, then labels p_i.
2. *C_i' contains p_i and one more point.* Let that point be p_j. As in the previous case, note that the positioning of the label for p_i cannot directly affect the positioning of the labels for p_k and p_l. Thus we need only place p_i's label so as to avoid restricting the placement of p_j's label. To do this, we remove p_i, recursively label the remaining points, then label p_i; since only the label of p_j can affect the label of p_i, we can always rotate the label of p_j if needed (if it actually contains p_i, we need to rotate it; but then also, the rotation cannot affect any other labeling squares) and then label p_i itself.
3. *C_i' contains p_i and two more points.* Let these points be p_j and p_k. We further subdivide this case as follows. Let C_i'' be the circle centered at p_i of radius $\frac{D_5(S)}{4\alpha}$, half the radius of C_i' and a quarter of the radius of C_i. We now distinguish three cases, according to the number of points within C_i''.
   (a) *C_i'' only contains p_i.* In that case, we remove p_i, recursively label the remaining points, then come back to label p_i itself. Because p_j and p_k are at least one diagonal away from p_i, their labels cannot include p_i and we can always place a labeling square at p_i without intersecting the labels of p_j and p_k.
   (b) *C_i'' contains p_i and one more point.* Let this additional point be p_j. Thus we have p_i and p_j in C_i'', p_k in C_i' but not C_i'', and p_l in C_i, but not C_i'. If p_l is at least 3 diagonals away from p_i, we can treat this case exactly like sub-case (c) below. Thus assume that p_l is less than three diagonals away. We proceed much as in case (2) above: we remove p_i, recursively label the remaining points, then return to label p_i. The labeling square of p_j might contain p_i; in that case, we rotate it to a position where it does not intersect the labeling square of p_k and does not contain p_i. (Given the constraints defining this case, one can verify that such a position can always be computed.) Then we place a square at p_i; since there are at most two constraints, (the labeling squares of p_j and p_k) this can be done in constant time.

(c) C_i'' *contains* p_i *and one more point.* Let this additional point be p_j. Thus we have p_i and p_j in C_i'', p_k in C_i' but not C_i'', and p_l in C_i, but not C_i'. If p_l is at least 3 diagonals away from p_i, we can treat this case exactly like subcase (c) below. Thus assume that p_l is less than three diagonals away. We proceed much as in case (2) above: we remove p_i, recursively label the remaining points, then return to label p_i. The labeling square of p_j might contain p_i; in that case, we rotate it out of the way (which we can do in such a case without intersecting the labeling square of p_k and p_l Lemma). Then we place a square at p_i; since there are at most two constraints (the labeling squares of p_j and p_k), this is easily done.

(d) C_i'' *contains* p_i *and two more points.* In this case, we remove all three points at once, recursively label all remaining points, and then proceed to label our three points. The only constraint on the labels of p_i, p_j, and p_k is due to the labeling square of p_l. Note that the labeling square of p_l cannot include any of the three points to be labeled and thus need not be altered. It is now a simple matter to place labeling squares for all three points.

4. C_i' *contains* p_i *and three more points.* By the same reasoning as in case 1, the labeling of the four points cannot affect the labeling of any other point of S, because all other points of S are at least two diagonals away. We know that we can label any subset of four isolated points with arbitrarily large squares; in particular, we can label our subset with squares of the desired size. Thus our procedure removes all four points, recursively labels the remaining points, then labels the four points as discussed.

This recursive procedure starts by computing $D_5(S)$, which takes $O(n \log n)$ time. At each step in the recursion, the work done is constant, so that the procedure runs in linear time after determining $D_5(S)$. Overall, then, our approximation procedure runs in $O(n \log n)$ time. Combining these observations with Theorem 1, we get our main result for MLUS.

Theorem 2. The MLUS problem can be approximated to within a factor of $8\sqrt{2}/\cos(\pi/5)$ in $O(n \log n)$ time, where n is the number of features to be labeled.

5 Map Labeling With Uniform Circles (MLUC)

Let $R^\star$ denote the size of the circles in the optimal solution. We derive an upper bound for this size as a function of $D_3(S)$, much in the same fashion as we bounded the size of squares as a function of $D_5(S)$. The basic approach is similar: we consider just three points forming a set of diameter D_3 and bound the size of the circle as a function of D_3; larger sets of points must yield circles that are no larger. By arguments similar to those in the proof of Lemma 3, we can show the following.

Lemma 5. A set of three points with diameter D_3 has optimal labeling circles of size $R^\star \leq (2+\sqrt{3}) \cdot D_3$.

Theorem 3. A set S of points has maximum uniform labeling circles of diameter $R^* \leq (2+\sqrt{3}) \cdot D_3(S)$.

Let C_i denote the circle centered at point $p_i \in S$ with radius $\frac{D_3(S)}{\alpha}$, where $\alpha > 2$ is a constant. By Lemma 2, C_i contains at most two points, including p_i itself; let p_j be the other point. Note that C_i, C_j, and their intersection all contain exactly two points, namely p_i and p_j.

We now present our approximation procedure; as in the MLUS problem, we state a theorem describing our procedure and give the procedure itself as a proof.

Lemma 6. Given a set S of points to be labeled with uniform circles, there exists a set of circular labels, each of size $R' \geq \frac{D_3(S)}{2\alpha}$.

The proof is quite simple. Note that C_i can contain at most two points, say p_i and p_j. We can label these points with circles of diameter $\frac{D_3(S)}{2\alpha}$ that are centered on the line $\overline{p_i p_j}$ and are placed opposite each other, avoiding the central segment $\overline{p_i p_j}$; by construction, these circles cannot intersect circles attached to points outside C_i.

Using the results of [EE94], we can compute $D_3(S)$ in $O(n \log n)$ time. We can determine the nearest neighbor of each point (the p_j for our p_i) in $O(n \log n)$ time using standard techniques. Placing the labeling circles takes constant time per circle. Thus our approximation algorithm runs in $O(n \log n)$ time overall. Combining these observations with Theorem 3, we get our main result about MLUC.

Theorem 4. The MLUC problem can be approximated to within a factor of $4(2+\sqrt{3})$ in $O(n \log n)$ time, where n is the number of features to be labeled.

6 A Bicriteria Approximation Algorithm

We now consider the variant of the map-labeling problem in which a few point features are allowed to remain unlabeled. We present a polynomial-time approximation scheme for this problem. Define a (bicriteria) polynomial-time (α, β)-approximation algorithm for the MLUS problem as a polynomial-time approximation algorithm that finds a placement of at least α circles such that the size of each circle is at least β times the size of a circle in an optimal solution that places circles at each point. Such an algorithm is a bicriteria approximation, governed by α and β.

The basic idea behind our approximation algorithm is to construct a number of geometric intersection graphs and solve the maximum independent set problem for each of the graphs. We then argue that a good approximate solution for the map-labeling problem is given by the solution to one of the graphs. We first consider a restriction of the MLUS problem in which each square must be placed so that its sides are parallel to the axes; call this problem MLUS-AP.

An undirected graph is a *square graph* if and only if its vertices can be put in one-to-one correspondence with uniformly-sized squares in the plane in such a way that two vertices are joined by an edge if and only if the corresponding squares intersect. (We assume that tangent squares intersect.) For any fixed $\lambda > 0$, we say that a square graph is a *λ-precision square graph* if the centers of any two squares are separated by at least λ times the size of a square.

We reduce the MLUS-AP problem to that of finding maximum independent sets for a number of squares graphs. Specifically, given an instance S of MLUS-AP, we construct $O(\log 2D_5(S))$ square graphs, each of size polynomial in S. For each square graph thus created, we obtain an approximate solution to the Maximum Independent Set problem—for which a polynomial-time approximation scheme is known to exist in a variety of geometric graphs [HM+95].

Theorem 5. For any fixed $\epsilon > 0$, given an instance S of n points of MLUS-AP, our algorithm finds a placement of at least $(1-\epsilon) \cdot n$ squares of size at least $\frac{1}{1+\epsilon} \cdot OPT$, where OPT denotes the size of the squares in an optimal solution.

Proof: Since $OPT \leq 2D_5(S)$, there exists some iteration k' such that $(1+\epsilon)^{k'} \leq OPT \leq (1+\epsilon)^{k'+1}$. By the construction of the square graph in Step 2a, it is clear that the optimal solution to the independent set problem for the set of squares $S_{k'}$ has n elements; thus Step 2b finds a placement of $m_{k'} \geq (1-\epsilon) \cdot n$ squares. Since we choose the largest k that gives a placement of at least $(1-\epsilon) \cdot n$ squares, it follows that

- *Input:* A set of points $\{p_1, \ldots, p_n\}$ in the plane and an accuracy requirement $\epsilon > 0$.
- *Output:* A placement of isothetic (axis-parallel) squares $S_1, \ldots, S_m$, such that p_i lies on the boundary of square S_i and the squares S_i are disjoint, $m \geq (1-\epsilon) \cdot n$, and the size of each square is at least $\frac{1}{(1+\epsilon)}$ times the optimal solution.
- *Procedure:*
  1. Let r denote the smallest integer such that $(1+\epsilon)^r \geq 2D_5(S)$.
  2. For $k \leftarrow 1$ to r do
     (a) Construct a square graph, with squares of size $(1+\epsilon)^k - (1+\epsilon)$, as follows. Let $q = \lceil \frac{1}{\epsilon} \rceil$. On each side place marks $\epsilon \cdot 2D_5(S)$ apart and label these marks by indices $n_1, \ldots, n_{4q}$. For each point p_i and each mark n_j, place a copy of a square in four ways so that mark n_j coincides with p_i. Denote the set of squares thus obtained by $S_k = \{S_1, \ldots, S_{4qn}\}$.
     (b) Solve the Maximum Independent Set problem for the set of squares S_k using the algorithm of [HM+95]; let m_k be the size of the independent set returned by the algorithm.
  3. Let k^* denote the largest value of k obeying $m_k \geq (1-\epsilon) \cdot n$.
  4. The solution output by the heuristic consists of the placement in iteration k^*.

the size of the squares in iteration k^* is at least $(1+\epsilon)^{k'} - 2\epsilon(1\epsilon)^{k'}$. The second term in the expression arises due to the discretization of the possible positions at which the square could be placed. Thus we have

$$(1+\epsilon)^{k'} - 2\epsilon(1\epsilon)^{k'} \geq (1+\epsilon)^{k'}(1-2\epsilon) \geq \frac{(1-2\epsilon)}{(1+\epsilon)} \cdot OPT.$$

Theorem 6. For any fixed $\epsilon > 0$, our algorithm runs in $O(n \log D_5(S))$.

Proof: Observe that, for each fixed $\epsilon > 0$, the number of squares in S_k is $\lceil \frac{n}{\epsilon} \rceil = O(n)$. The maximum value of r is $O(\log D_5(S))$. Steps 2a and 2b take $O(n)$ time for each iteration, since, for each fixed ϵ, we obtain a λ-precision square graph with n squares. The algorithm of [HM+95] runs in linear time. All other steps take constant time.

Note that $\log D_5(S)$ is bounded by a polynomial in the input size, since it is the logarithm of a distance between two points given (by their coordinates) in the input.

For the case of the map labeling problem of [FW91], we can further reduce the running time by observing that we have only four possible positions for placing each square. Also note that our algorithms extend to the case when we are allowed to place uniform rectangles at each point feature. Indeed, although the basic idea is quite simple, it is very general and extends to a large number of variants of the map labeling problem, some of which we sketch below.

1. *Arbitrary orientation:* We have already discretized (through the system of marks) the position of the point on the boundary of the labeling square; to handle arbitrary orientation, we also discretize the angle of the labeling square with the horizontal axis. Specifically, we divide the 2π radians in discrete subangles of ϵ, thus yielding $2\pi/\epsilon$ possible angular positions. The algorithm otherwise proceeds as outlined in the case of MLUS-AP, albeit with higher running-time factors.
2. *Circles and other regular polygons:* The algorithm can easily be extended to other regular polygons. (The algorithm of [HM+95] works on many variations of geometric graphs, not just square graphs.)

It can also be applied with a slight modification to circles: we then use the maximum independent set algorithm for unit disk graphs given in [HM+95].

3. *Placement with non-uniform squares:* Assume that different-sized squares can be used under the condition that the ratio of the sides of the largest to the smallest square be bounded (a reasonable aesthetic requirement). In Step 2a, we place squares of different sizes at each point: $(1+\epsilon)^r$ denotes the size of the smallest square and other squares are scaled as required by the problem. In Step 2b we again solve the maximum independent set problem for the induced graph. This time, we use the algorithm given in [HM+95] to find a large independent set for (p,q)*-civilized graphs*.
4. *Placement for vertices of a graph:* Consider a generalization of the map labeling problem in which we are given a graph in the plane and we wish to label the vertices of the graph. The labels must be mutually non-intersecting and must not intersect the the edges of the graph. In such a case, we do the following: every time we construct a square graph, we remove those squares that overlap with any of the edges. The algorithm is otherwise similar and can also be extended to partial overlap of the squares.
5. *Placement for rectilinear line features:* Our approximation algorithm extends to labeling line features in which the line segments are rectilinear. We omit this due to lack of space.

7 Labeling a Rectilinear Map

In this section we study the problem of how to label a rectilinear map. As discussed in Section 2 each line segment can be labeled in one of three possible ways. We say that a rectangle is a *valid label* for a line segment if the rectangle is positioned in exactly one of three possible ways with respect to the line segment; among other things, a valid label has as length the length of the segment it labels and we refer to its other dimension as its width.

Definition 4. An instance of the problem of *Rectilinear Segment Labeling* (RSL) consists of n rectilinear line segments (features) in the plane and a positive integer bound B. The question is whether there exists a placement for n rectangles, each of width B, such that

- no two rectangles intersect; and
- each rectangle is a valid label for a distinct segment.

We first present an optimal $\Theta(n \log n)$ time algorithm for the case when all segments are horizontal and then present an approximation algorithm for the general problem.

7.1 An Optimal Solution for Horizontal Segments

Let the n horizontal segments be denoted s_i, $1 \leq i \leq n$; without loss of generality, we assume that no two segments have the same y-coordinates. The *vertical neighbors* of a segment s_i are those segments that are first hit by s_i when s_i is translated vertically. It is well known that the set of relations "is a vertical neighbor of" (also called a trapezoidal decomposition or a vertical visibility map) can be computed in $O(n \log n)$ time, e.g., through simple scanning (as was first observed by Bentley and Ottmann [BO79]). The resulting map, which we shall call the VVM, is linear in size and can be used for point location in logarithmic time. We shall assume that the map has guards, i.e., segments with a left endpoint left of all n input segments and a right endpoint right of all input segments, with one segment above all others and another below all others.

Let $r_a^W(s)$, $r_b^W(s)$, and $r_m^W(s)$ denote the rectangles of width W placed respectively above, below, and across horizontal segment s, and let $rec_W(s)$ denote the set of all three possible rectangles.

Lemma 7. Let s_i and s_j be two segments such that neither is a vertical neighbor of the other. If the optimal width of labels is W, then $rec_W(s_1)$ will not first intersect $rec_W(s_2)$ and vice versa.

While rather obvious, this lemma has an important corollary: the number of potential intersections among labeling rectangles that we need to consider is $O(n)$. Consider the directed acyclic graph $G = (V, E)$ where each node corresponds to a segment and there is an edge from vertex s_i to s_j if there is a vertical visibility edge between segments s_i and s_j and s_i is below s_j (i.e., in the visibility map make the vertical visibility edges directed upwards). G has n vertices and $O(n)$ edges and can be topologically ordered in $O(n)$ time; let the ordering of the nodes be $s_0, s_2, \ldots, s_{n+1}$, where s_0 and s_{n+1} are the two guards. For each segment s_i, we maintain three variables, W_{i1}, W_{i2}, and W_{i3}, with the following interpretation: after having processed segment s_i, W_{ik} is the largest possible height of a feasible solution among all (transitively closed) predecessors of s_i, subject to the constraint that segment s_i is in state k. Initially we set all W_{ik}'s to infinity. Processing s_j takes $O(1)$ time per incoming edge from a predecessor segment s_i. Using the quantities W_{i1}, W_{i2}, and W_{i3}, we update W_{j1}, W_{j2}, and W_{j3} in $O(1)$ time as follows; we use d_{ij} to denote the vertical distance between s_i and s_j.

- $W_{j1} = \min\{W_{j1}, \max\{x_1, x_2, x_3\}\}$
  where $x_1 = \min\{W_{i1}, d_{ij}\}$, $x_2 = \min\{W_{i2}, 2d_{ij}/3\}$, and $x_3 = \min\{W_{i3}, d_{ij}/2\}$.
- $W_{j2} = \min\{W_{j2}, \max\{y_1, y_2, y_3\}\}$
  where $y_1 = \min\{W_{i1}, 2d_{ij}\}$, $y_2 = \min\{W_{i2}, d_{ij}\}$, and $y_3 = \min\{W_{i3}, 2d_{ij}/3\}$.
- $W_{j3} = \min\{W_{j3}, \max\{z_1, z_2, z_3\}\}$
  where $z_1 = W_{i1}$, $z_2 = \min\{W_{i2}, 2d_{ij}\}$, and $z_3 = \min\{W_{i3}, d_{ij}\}$.

These formulas are self-explanatory. After processing s_{n+1} (the second guard) the optimal solution v_{opt} is equal to W_{n3}. We can thus state the first of two main results about the RSL problem.

Theorem 7. The problem of labeling a set of horizontal segments can be solved optimally in $O(n \log n)$ time, which is optimal in the algebraic decision tree model of computation.

Proof: The upper bound follows easily from the above discussion. We use a linear time reduction from the Min-Gap problem of n reals, which is known to have a lower bound of $\Omega(n \log n)$ under the algebraic decision tree model of computation [BO83], to show that computing the optimal value v_{opt} when all n segments are horizontal, has the same lower bound. The proof is omitted due to lack of space.

7.2 An Approximation Solution for the General Problem

If we allow two of the three possible placements for a label (excluding the placement athwart the segment), then the problem can be modelled as a series of 2SAT problems and solved in $O(n^2)$ time [FW91]. Denote the optimal solution to this restricted version of the problem by v^* and denote the optimal solution to our version (with three choices of placement allowed) by v_{opt}.

Theorem 8. $v_{opt}/v^* \leq 2$.

Proof: Consider the solution of optimal value v_{opt}. For each rectangle in states 1 or 3, shrink its width to half. For each rectangle in state 2, simply remove one of its halves to make it a rectangle in state 1 or 3 with half its original width. Clearly the new solution is a valid labeling of value $v_{opt}/2$ in which all rectangles are in state 1 or 3 only. Since the algorithm finds the best solution under this restriction, we must have $v^* \geq v_{opt}/2$.

It is easy to construct an example where this bound is achieved.

References

[CGI96] ACM Computational Geometry Impact Task Force, "Application challenges to computational geometry," Princeton U. Technical Report TR-521-96,.

[AF84] J. Ahn and H. Freeman, "A program for automatic name placement," *Cartographica* **21 (2 & 3)** (1984), 101–109.

[AHU] A. Aho, J. Hopcroft and J. Ullman, *The Design and Analysis of Computer Algorithms*, Addison Wesley, Reading MA., 1974.

[BO79] J.L. Bentley and T.A. Ottmann, "Algorithms for reporting and counting geometric intersections," *IEEE Trans. Comput.* **C28** (1979), 643–647.

[BC94] D. Beus and D. Crockett, "Automated production of 1:24,000 scale quadrangle maps," *Proc. 1994 ASPRS/ACSM Ann. Convention and Exposition*, Vol. 1, (1994), 94–99.

[BO83] M. Ben-Or, "Lower bounds for algebraic computation trees," *Proc. 15th Ann. ACM Symp. on Theory of Comput.* STOC-83, (1983), 80–86.

[AF84] L. Carstensen, "A Comparison of Simple Mathematical Approaches to the Placement of Spot Symbols," *Cartographica* **24 (3)** (1987), 46–63.

[CMS93] J. Christensen, J. Marks, and S. Shieber, "Algorithms for cartographic label placement," *Proc. 1993 ASPRS/ACSM Ann. Convention and Exposition* Vol. 1, (1993), 75–89.

[CMS95] J. Christenson, J. Marks, and S. Shieber. "An Empirical Study of Algorithms for Point-Feature Label Placement," *ACM Transactions on Graphics,* 1995.

[DF92] J. Doerschler and H. Freeman, "A rule-based system for cartographic name placement," *Commun. ACM* **35** (1992), 68–79.

[DLSS] A. Datta, P. Lenhof, C. Schwarz, and M. Smid, "Static and dynamic algorithms for k-point clustering problems," *Proc. 3rd Workshop on Algorithms and Data Structures* WADS-93, Springer-Verlag LNCS #709, (1993), 265–276.

[EE94] D. Eppstein and J. Erickson, "Iterated nearest neighbors and finding minimal polytopes," *Discrete & Comput. Geom.* **11** (1994), 321–350.

[FW91] M. Formann and F. Wagner, "A packing problem with applications to lettering of maps," *Proc. 7th Ann. ACM Sympos. Comput. Geom.* CG-91, (1991), 281–288.

[GJ79] M. R. Garey and D. S. Johnson, *Computers and Intractability. A Guide to the Theory of $\mathcal{NP}$-Completeness*, Freeman, San Francisco CA, 1979.

[Hi82] S. Hirsh, "An Algorithm for Automatic Name Placement Around Point Data," *The American Cartographer* **9 (1)** (1982), 5–17.

[HM85] D. Hochbaum and W. Maass "Approximation Schemes for Covering and Packing Problems in Image Processing and VLSI," *Journal of ACM,* **32 (1)**, Jan. (1985), pp. 130-136.

[HM+95] H. B. Hunt III, M. V. Marathe, V. Radhakrishnan, S. S. Ravi, D. J. Rosenkrantz and R. E. Stearns, "A Unified Approach to Approximation Schemes for $\mathcal{NP}$- and $\mathcal{PSPACE}$-Hard Problems for Geometric Graphs," *Proc. 2nd European Symp. on Algorithms* ESA-94, (1994), 468–477.

[Im75] E. Imhof, "Positioning names on maps," *The American Cartographer* **2** (1975), 128–144.

[Jo89] C. Jones, "Cartographic name placement with Prolog," *IEEE Computer Graphics and Applications* **5** (1989), 36–47.

[KR92] D. Knuth and A. Raghunathan, "The problem of compatible representatives," *SIAM J. Disc. Math.* **5** (1992), 422–427.

[MS91] J. Marks and S. Shieber, "The Computational Complexity of Cartographic Label Placement," March (1991).

[PS85] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, 1985.

[vR89] J. van Roessel, "An Algorithm for Locating Candidate Labeling Boxes within a Polygon," *The American Cartographer* **16 (3)** (1989), 201–209.

[Wag94] F. Wagner, "Approximate map labeling is in $\Omega(n \log n)$," *Inform. Process. Lett.* **52** (1994), 161–165.

[WW95] F. Wagner and A. Wolff, "Map labeling heuristics: provably good and practically useful," *Proc. 11th Ann. ACM Sympos. Comput. Geom.* CG-95, (1995), 109–118.

[WF95] A. Wolff, "Map Labeling," *Diploma Report, Freie University, Berlin, Department of Mathematics.* (1995).

[Yo72] P. Yoeli, "The Logic of Automated Map Lettering," *The Cartographic Journal* **9 (2)** (1972), 99–108.

[Zo86] S. Zoraster, "Integer Programming Applied to the Map Label Placement Problem," *Cartographica* **23 (3)** (1986), 16–27.