New Algorithms for Two-Label Point Labeling*

Zhongping Qin[1], Alexander Wolff[2], Yinfeng Xu[3], and Binhai Zhu[4]

[1] Dept. of Mathematics, Huazhong University of Science and Technology, Wuhan, China, and Dept. of Computer Science, City University of Hong Kong. zqin@cs.cityu.edu.hk
[2] Institute of Mathematics and Computer Science, Ernst Moritz Arndt University, Greifswald, Germany. awolff@mail.uni-greifswald.de
[3] School of Management, Xi'an Jiaotong University, Xi'an, China. yfxu@xjtu.edu.cn
[4] Dept. of Computer Science, City University of Hong Kong, and Montana State University, Bozeman, MT 59717, USA. bhz@cs.montana.edu

Abstract. Given a label shape L and a set of n points in the plane, the 2-label point-labeling problem consists of placing $2n$ non-intersecting translated copies of L of maximum size such that each point touches two unique copies—its labels. In this paper we give new and simple approximation algorithms for L an axis-parallel square or a circle. For squares we improve the best previously known approximation factor from $\frac{1}{3}$ to $\frac{1}{2}$. For circles the improvement from $\frac{1}{2}$ to ≈ 0.513 is less significant, but the fact that $\frac{1}{2}$ is not best possible is interesting in its own right. For the decision version of the latter problem we have an NP-hardness proof that also shows that it is NP-hard to approximate the label size beyond a factor of ≈ 0.732. As their predecessors, our algorithms take $O(n \log n)$ time and $O(n)$ space.

1 Introduction

Label placement is one of the key tasks in the process of information visualization. In diagrams, maps, technical or graph drawings, features like points, lines, and polygons must be labeled to convey information. The interest in algorithms that automate this task has increased with the advance in type-setting technology and the amount of information to be visualized. Due to the computational complexity of the label-placement problem, cartographers, graph drawers, and computational geometers have suggested numerous approaches, such as expert systems [1, 8], zero-one integer programming [22], approximation algorithms [6, 11, 19, 20], simulated annealing [4] and force-driven algorithms [13] to name only a few. The ACM Computational Geometry Impact Task Force report [3] denotes label placement as an important research area. Manually labeling a map is a tedious task that is estimated to take 50 % of total map production time.

* This research was conducted during a visit of Z. Qin and Y. Xu to City University of Hong Kong and of A. Wolff to Hong Kong University of Science and Technology. Our work was supported by NSF of China, grant No. 19731001, and by the Hong Kong RGC CERG grants CityU-1103/99E and HKUST-6144/98E.

In this paper we deal with a relatively new variant of the general label placement problem, namely the 2-label point-labeling problem. It is motivated by maps used for weather forecasts, where each city must be labeled with two labels that contain the city's name and, say, its predicted temperature.

The 2-label point-labeling problem is a variant of the 1-label problem that allows sliding. Sliding labels can be attached to the point they label anywhere on their boundary. They were first considered by Hirsch [13] who gave an iterative algorithm that uses repelling forces between labels in order to eventually find a placement without or only few intersecting labels. Van Kreveld et al. gave a polynomial time approximation scheme and a fast factor-2 approximation algorithm for maximizing the number of points that are labeled by axis-parallel sliding rectangular labels of common height [18]. They also compared several sliding-label models with so-called fixed-position models where only a finite number of label positions (or *label candidates*) per point is considered, usually a small constant like four [4, 11, 19]. Another generalization was investigated in [6, 21], namely arbitrarily oriented sliding labels.

Point labeling with circular labels, though not as relevant for real-world applications as rectangular labels, is a mathematically interesting problem. The 1-label case has already been studied extensively [6, 7, 17]. For maximizing the label size, the currently best approximation factor is $\frac{1}{3.6}$ [7].

The 2- or rather multi-label labeling problem was first considered by Kakoulis and Tollis who presented two heuristics for labeling the nodes and edges of a graph drawing with several rectangles [14]. Their aim was to maximize the number of labeled features. One of their algorithms is iterative, the other uses a maximum-cardinality bipartite matching algorithm that matches cliques of label candidates with the elements of the graph drawing that are to be labeled. They do not give any runtime bounds or approximation factors.

For the two problems that we will consider in this paper, namely maximizing the size of axis-parallel square and circular labels, two per point, Zhu and Poon gave the first approximation algorithms [20]. They achieved approximation factors of $\frac{1}{4}$ and $\frac{1}{2}$ for square and circle pairs, respectively. Both algorithms rely on the fact that there are disjoint regions around all (pairs of) input points into which the labels can be safely placed. Recently Zhu and Qin improved the result for pairs of square labels to a factor of $\frac{1}{3}$ [21]. They exploit the structure of a graph that has a node for each input point and an edge for each pair of points closer than $\frac{2}{3}$ times an upper bound for the maximum label size.

In this paper we give new and simple approximation algorithms for the 2-square and the 2-circle point-labeling problem. For squares we improve the approximation factor of Zhu and Qin's algorithm from $\frac{1}{3}$ to $\frac{1}{2}$. For circles we present an algorithm with an approximation factor of $\frac{1}{1+\cos 18^\circ} \approx 0.513$. Here the improvement over Zhu and Poon's factor-$\frac{1}{2}$ approximation algorithm is less significant, but the fact that $\frac{1}{2}$ is not best possible is interesting in its own right. For the decision version of the 2-circle labeling problem we have an NP-hardness proof that also shows that it is NP-hard to approximate the label size beyond a factor of ≈ 0.732. However, due to space limitations, we cannot give the proof here. Other than all previous approximation algorithms for 2-label placement,

our new algorithms do not necessarily have to compute an upper bound for the maximum label size explicitly. We keep the $O(n \log n)$ time and $O(n)$ space bounds of the previous algorithms.

For the 2-square labeling problem the improved approximation factor is made possible by restricting the search to a subset of the solution space. Within this subset, optimal solutions can be computed easily and their labels are at most by the above mentioned constant factors off the maximum label size. The special case that we solve optimally is the following: we label points with rectangles of height-width ratio 2 of maximum size in one of four positions. Our algorithm is the first *point*-labeling algorithm that solves the size maximization problem for more than two label positions optimally in polynomial time. So far such algorithms have only been know for labeling axis-parallel line segments [16]. Our algorithm also improves the approximation factor of the only known algorithm [12] for Knuth and Raghunathan's Metafont labeling problem [15] from $\frac{1}{3}$ to $\frac{1}{2}$.

Throughout this paper we consider labels being topologically *open*, and we define the *size* of a solution to be the diameter in the case of circular labels and the length of the shorter label edge in the case of rectangular labels. We refer to a label placement as *feasible* if no two labels intersect and as *optimal* if additionally labels have the largest possible size. We will only consider $n > 2$.

2 Two-Square Labeling

Definition 1 (2-square point-labeling problem). *Given a set P of n points in the plane, find a set of $2n$ axis-parallel, uniform, non-intersecting, maximum-size open squares, such that each point touches two unique squares.*

The uniqueness contraint does not forbid a point to touch more than two squares, but ensures there is a function that assigns to each point exactly two squares that touch it.

The first approximation algorithm for the 2-square point-labeling problem was suggested by Zhu and Poon [20]. This algorithm labels pairs of points in order of increasing distance. The labels are placed to the left of the left point and to the right of the right point—except when the two points lie on a vertical, see Figure 1 (a) and (b), respectively. In other words, the algorithm does not really place two square labels at each point but one rectangle of height-width ratio 2 or 1/2. The rectangle is attached to its point in the midpoint of either of its long edges, i.e. the algorithm uses only *four* of the infinitely many possible label positions.

We use this observation as follows. First we device an $O(n \log n)$-time algorithm for the 1-rectangle 4-position point-labeling problem, where all points are labeled with maximum-size rectangles in one of the four discrete positions depicted in Figure 2. Then we show how an optimal solution for the 2-square labeling problem can be transformed into a solution of the 1-rectangle labeling problem by using rectangles of half the square size, see Figure 3. Thus the 1-rectangle labeling algorithm already yields a factor-$\frac{1}{2}$ approximation algorithm for the 2-square labeling problem.

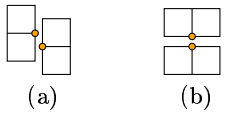


Fig. 1. Label placement of the 2-square labeling algorithm of Zhu and Poon.

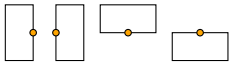


Fig. 2. Label candidates for 1-rectangle 4-position labeling.

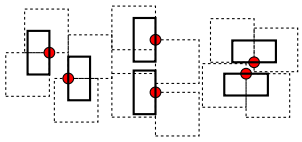


Fig. 3. Mapping a 2-square labeling to a 1-rectangle 4-position labeling of half the size.

Apart from the improved approximation factor, our approach has the advantage that we do not have to compute an upper bound for the maximum label size explicitly. Both previous algorithms [20, 21] first have to determine such an upper bound before they can actually place labels whose size depends on this upper bound. Zhu and Poon use $D_{3,\infty}$, the minimum over the diameters of all 3-subsets of the input points, as an upper bound for the maximum label size. $D_{3,\infty}$ can be computed in $O(n \log n)$ time [5]. Instead of computing $D_{3,\infty}$ we preprocess the input by computing relevant adjacency information, i.e. for each input point we find a constant number of rectilinear nearest neighbors (in the L_∞-metric). For this task a simple $O(n \log n)$ algorithm is known [10]. Of course $D_{3,\infty}$ can be computed in linear time from nearest neighborhood data, but we make better use of this information by solving the following problem optimally.

Definition 2 (1-rectangle 4-position point-labeling problem). *Given n points in the plane, find a set of n congruent, axis-parallel, non-intersecting, maximum-size open rectangles of height-width ratio 2 or 1/2, such that each point touches a unique rectangle.*

The decision version of this problem is the question whether a set of points can be labeled with congruent rectangles of a *given* size. If we encode the four label candidates of each point p by *two* Boolean variables p_1 and p_2 as in Figure 4, we can construct a 2-SAT formula that is equivalent to the decision version of our problem. Thus it can be solved in time and space linear in the number of clauses [9]. Here the number of clauses is at most three times the number of pairs of intersecting label candidates, which must be computed beforehand. A similar strategy has been applied to encode label positions for labeling rectilinear line segments with rectangles of maximum height [16].

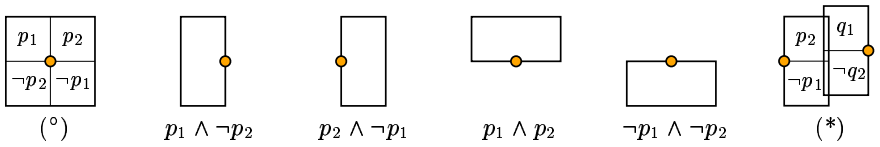


Fig. 4. We encode the four label positions of a point p by the values of two Boolean variables p_1 and p_2 (°). A label intersection (*) can then be read as $\neg((p_2 \wedge q_1) \vee (p_2 \wedge \neg q_2) \vee (\neg p_1 \wedge \neg q_2))$. This equals the 2-SAT formula $(\neg p_2 \vee \neg q_1) \wedge (\neg p_2 \vee q_2) \wedge (p_1 \vee q_2)$.

In order to device an algorithm that maximizes the rectangle size and runs in $O(n \log n)$ time, we use the same strategy as two existing algorithms. The algorithms $AS4a$ [11] and B [19] also try to maximize the label size. They solve the 1-square 4-position point-labeling problem, where labels are restricted to uniform *squares*, and each label must be placed such that one of its *corners* coincides with the point it labels. Like in our problem each point has four label candidates. However, in their problem the candidates only intersect at their borders, while in ours each candidate is completely contained in (the closure of) two others. This is why we can solve our problem optimally, while $AS4a$ and B are factor-$\frac{1}{2}$ approximation algorithms.

Algorithm B proceeds as follows. First it uses the above mentioned simple sweep-line algorithm to detect a constant number of rectilinear nearest neighbors for each point [10]. Then B computes a list of *conflict sizes* from the set N of pairs of neighboring points. A conflict size is a label size for which two label candidates touch but do not contain any input points. Algorithm B does a binary search on the list of conflict sizes. This is sufficient since the intersection graph of the label candidates does not change between two conflict sizes. For each conflict size, B solves the decision problem in three steps. First the algorithm extracts the intersection graph for the current label size from N. Then it uses certain (partly heuristical) rules to simplify the intersection graph until each point has at most *two* label candidates left. Finally B makes a 2-SAT clause for each edge of the intersection graph and tries to find a satisfying truth assignment for the resulting 2-SAT formula. If one exists, it corresponds to a label placement and the search is continued with a greater label size, otherwise with a smaller one.

Asymptotically B does not need more space than the size of N, which is linear in n. The binary search consists of $O(\log n)$ tests. Again due to the linear size of N, each test takes $O(n)$ time. This adds up to $O(n \log n)$ time in total.

Our algorithm for 1-rectangle 4-position point labeling differs from B only in that it is not necessary to reduce the number of candidates of each point from four to two, since we can immediately encode our four label candidates by a 2-SAT formula. All that remains to show is that as for 1-square 4-position point labeling it is sufficient to consider the conflict sizes induced by a constant number of rectilinear nearest neighbors per point.

Lemma 1. *Let r_{opt} be the size of an optimal solution for the 1-rectangle 4-position point-labeling problem. Then for any label size $\rho \leq r_{\text{opt}}$ the label candidates of a point $p \in P$ can intersect only candidates of the 17 points $q \in P \setminus \{p\}$ that are closest to p in the L_∞-metric.*

Since this observation is analogous to [11, Lemma 1], we omit the proof here. The algorithm $AR2$ in that paper, however, needs $O(n \log n)$ time for the *decision* version of a problem similar to our 1-rectangle 4-position labeling problem. (There, only two square labels per point are allowed.) Lemma 1 yields the time and space complexity of our 1-rectangle 4-position labeling algorithm:

Lemma 2. *An optimal solution of the 1-rectangle 4-position point-labeling problem can be computed in $O(n \log n)$ time using linear space.*

Theorem 1. *A feasible solution of the 2-square point-labeling problem of at least half the optimal size can be computed in $O(n \log n)$ time using linear space.*

Proof. Given Lemma 2, we only have to show that an optimal solution of the 1-rectangle 4-position labeling problem always represents a feasible solution of the 2-square labeling problem of at least half the optimal size. In other words, if r_{opt} and s_{opt} are the sizes of optimal solutions of the 1-rectangle and the 2-square labeling problem for the same point set P, then we have to prove $r_{\text{opt}} \geq s_{\text{opt}}/2$. To show this we map a hypothetical optimal solution of the latter problem, i.e. a set $\mathcal{S}_{\text{opt}}$ of $2n$ squares of size s_{opt} into a feasible solution of the former problem, namely a set $\mathcal{R}$ of n rectangles of size $s_{\text{opt}}/2$. Since the labels in $\mathcal{R}$ cannot be larger than r_{opt}, the size of labels in an optimal solution of the 1-rectangle labeling problem, we have $r_{\text{opt}} \geq s_{\text{opt}}/2$.

The mapping is simple: for each point p we choose from its four label candidates the rectangle R_p such that a rectangle of twice the size of R_p (with p as scaling center) has the largest area of intersection with the two square labels of p in $\mathcal{S}_{\text{opt}}$. For our proof it does not matter that R_p is not uniquely defined.

It remains to show why R_p does not intersect the label R_q of some point $q \in P \setminus \{p\}$. Let $s_{\text{opt}} = 1$; the instance can always be scaled such that this is true. Recall that all labels in $\mathcal{S}_{\text{opt}}$ and $\mathcal{R}$ are topologically open. For $A \subseteq \mathbb{R}^2$ let $\overline{A}$ be the topological closure of A. For $p = (x_p, y_p)$ define the line segments $v(p)$ and $h(p)$ as the intersection of an open unit disk with the vertical and the horizontal through p, respectively. Let T_p and B_p be the top- and bottommost square labels of p in $\mathcal{S}_{\text{opt}}$; if their y-coordinates are the same, let T_p be the leftmost. Let $S_p = T_p \cup B_p$. We will use the same notations for q.

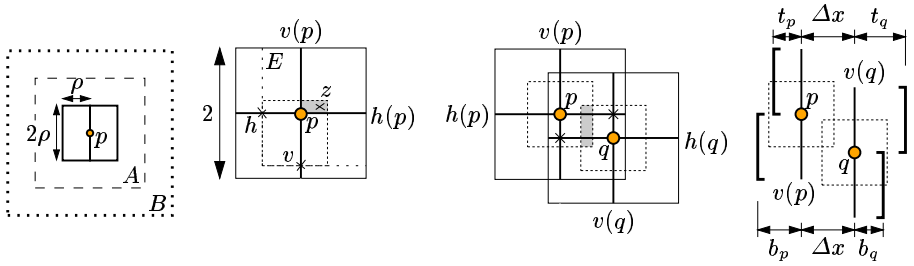
Observe that $v(p) \subseteq \overline{S_p}$ or $h(p) \subseteq \overline{S_p}$, otherwise there are points $h \in h(p) \setminus \overline{S_p}$ and $v \in v(p) \setminus \overline{S_p}$ that delimit region E in Figure 5 (b). Since E must contain S_p completely, we would have $T_p \cap B_p \neq \emptyset$.

Now suppose $R_p \cap R_q \neq \emptyset$. Then the distance $d_\infty(p, q)$ of p and q in the L_∞-metric is less than 1 since all points in R_p (R_q) have distance less than $\frac{1}{2}$ from p (q). For this reason $v(p)$ and $h(q)$ as well as $v(q)$ and $h(p)$ intersect, see Figure 5 (c). Since S_p and S_q do not intersect by definition, the observation above yields that $v(p)$ is contained in $\overline{S_p}$ and $v(q)$ in $\overline{S_q}$ or $h(p)$ is contained in $\overline{S_p}$ and $h(q)$ in $\overline{S_q}$. W.l.o.g. we may assume the former, otherwise we rotate the whole instance by 90° around p. Thus T_p lies above p, and B_p below p; the same holds for q. We can also assume that $x_p \leq x_q$, otherwise we mirror our instance.

Let $\Delta x = x_q - x_p$. Then $0 \leq \Delta x < 1$. Let t_p (b_p) be the horizontal distance between the *left* edge of T_p (B_p) and $v(p)$, see Figure 5 (d). Similarly, let t_q (b_q) be the horizontal distance between the *right* edge of T_q (B_q) and $v(q)$.

Now a simple packing argument will yield the contradiction. Since T_p and T_q do not intersect, we have $t_p + \Delta x + t_q \geq 2$. Since B_p and B_q do not intersect, we have $b_p + \Delta x + b_q \geq 2$. These inequalities sum up to $\Sigma := t_p + t_q + b_p + b_q \geq 4 - 2\Delta x$. Since $0 \leq \Delta x < 1$ we only have to consider the following two cases:

A) $\frac{1}{2} \leq \Delta x < 1$. Then $\Sigma > 2$ and $t_p + b_p \leq 1$, otherwise—due to our mapping—R_p lies completely left of $v(p)$ and cannot intersect R_q due to $\Delta x \geq \frac{1}{2}$.



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

Fig. 5. (a) Only the labels of the 17 nearest neighbors of p can intersect labels of p. (b) If there are points $v \in v(p) \setminus \overline{S_p}$ and $h \in h(p) \setminus \overline{S_p}$ then there is a point $z \in T_p \cap B_p$. (c) If rectangles R_p and R_q intersect, then $v(p) \cap h(q) \neq \emptyset$ and $v(q) \cap h(p) \neq \emptyset$. (d) t_p (b_p) is the horizontal distance between the left edge of T_p (B_p) and $v(p)$.

Similarly $t_q + b_q \leq 1$, otherwise R_q lies completely to the right of $v(q)$. Thus $t_p + t_q + b_p + b_q \leq 2$, contradicting $\Sigma > 2$.

B) $0 \leq \Delta x < \frac{1}{2}$. Then $\Sigma > 3$. Since t_p, t_q, b_p, and b_q are all at most 1, we get $t_p + b_p > 1$ and $t_q + b_q > 1$. But then R_p lies completely to the left of $v(p)$ and R_q to the right of $v(q)$—which means they do not intersect since we assumed $x_p \leq x_q$. □

Obviously the 1-rectangle 4-position algorithm solves an instance P of the 2-square labeling problem optimally if P has an optimal solution that uses only the four label positions depicted in Figure 1, as in the grid $\{(x, 2y) \mid 1 \leq x, y \leq n\}$.

However, our new 1-rectangle 4-position algorithm can also be used to find approximate solutions for another problem, namely the optimization version of the Metafont labeling problem [15]. Knuth developed the program Metafont as a tool for font design. In their paper, Knuth and Raghunathan introduced the name *problem of compatible representatives* for a large class of combinatorial problems that are NP-hard in general. This class of problems has been studied independently in the artificial intelligence community under the name *constraint satisfaction problem* (CSP). Knuth and Raghunathan investigated several special cases of the problem of compatible representatives, among them the Metafont labeling problem, where a set of n grid points is to be labeled with n 2×2 squares such that each square touches exactly one point in the center of one of its edges, see Figure 6. They showed that this 1-square 4-position point labeling problem is NP-complete. Note that the 1-rectangle 4-position problem considered in this section is *not* a more general case of the Metafont labeling problem; each of our rectangular label candidates is the union of two other candidates of the same point, while this is not the case with the square Metafont labels.

Formann and Wagner saw the connection between the Metafont labeling problem and the cartographic map-labeling problem they had attacked earlier [11]. They defined a maximization version of the Metafont labeling problem by dropping the grid constraint that comes from the Metafont application [12] and gave a factor-$\frac{1}{3}$ approximation algorithm similar to the algorithm *AS4a* [11].

Definition 3 (Metafont optimization problem). *Given a set of n points in the plane, find a set of n uniform, axis-parallel, non-intersecting, maximum-size open squares, such that each point touches* exactly *one square in the midpoint of one of its edges.*

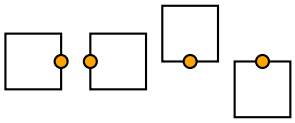


Fig. 6. The 4 label positions allowed in Metafont labeling.

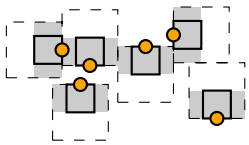


Fig. 7. Maping an optimal Metafont solution via rectangles to a Metafont solution of half the optimal size.

Obviously we can approximate the Metafont optimization problem exactly in the same way as the 2-square point-labeling problem, namely by running our 1-rectangle 4-position labeling algorithm. The mapping that transforms an optimal Metafont labeling to a feasible rectangle labeling is the same as for the 2-square problem. Here, however, it's trivial to show the feasibility of the resulting rectangle labeling, since each rectangle (shaded grey) is completely contained in a Metafont label (dashed), and no two Metafont labels intersect, see Figure 7. Finally we map each rectangle back to an inscribed square of half the optimal size (printed bold) in one of the four allowed positions of Figure 6.

This mapping automatically ensures the new uniqueness requirement of Definition 3. In a solution produced via the rectangle-labeling algorithm that is not necessarily the case, but the algorithm can be adjusted to disallow label placements where two labels completely share a long edge. Now Lemma 2 yields:

Theorem 2. *A feasible solution of the Metafont optimization problem of at least half the optimal size can be computed in $O(n \log n)$ time using linear space.*

3 Two-Circle Labeling

Definition 4 (2-circle point-labeling problem). *Given a set P of n points in the plane, find a set of $2n$ uniform, non-intersecting, maximum-size open circles such that each point touches exactly two circles.*

Zhu and Poon [20] have suggested the first approximation algorithm for this problem. Their algorithm always finds a solution of at least half the optimal size. The algorithm is very simple; it relies on the fact that D_2, the minimum Euclidean distance between any two points in P is an upper bound for the optimal label size (i.e. diameter), see Figure 9. On the other hand, given two points p and q in P, open circles $C_{p,D_2/2}$ and $C_{q,D_2/2}$ with radius $\frac{1}{2}D_2$ centered at p and q do not intersect. Thus if each point is labeled within its circle, no two labels will intersect. This allows labels of maximum diameter $\frac{1}{2}D_2$, i.e. half the upper bound for the optimal label size. The difficulty of the problem immediately comes into play when increasing the label diameter d beyond $\frac{1}{2}D_2$, since then

the intersection graph of the circles $C_{p,d}$ of all points p in P changes abruptly; the maximum degree jumps from 0 to 6.

Our approach also assigns each point a certain region such that no two regions intersect and each point can be labeled within its region. The regions we use are not circles but the cells of the *Voronoi diagram* of P, a well-known multi-purpose geometrical data structure [2]. We do not compute the Voronoi diagram explicitely—but use its dual, the *Delauney triangulation* [2]. For the description of our algorithm (see Figure 8) we need the following notation.

Definition 5. *Let* $d = \frac{D_2}{1+\cos 18^\circ} \approx 0.513\, D_2$ *and let* $p, q \in P$.

The pair (p, q) *is an edge of the Delaunay triangulation* $\mathrm{DT}(P)$ *of* P *if there is a (closed) disk* D *with* $D \cap P = \{p, q\}$. *The edge* (p, q) *is* short *if* $d(p, q) < 2d$, long *otherwise. Here* $d(p, q)$ *denotes the Euclidean distance of* p *and* q.

$\mathrm{Vor}(p) = \{x \in \mathbb{R}^2 \mid d(x, p) < d(x, q)\ \forall q \in P \setminus \{p\}\}$ *is the* Voronoi cell *of* p.

Given two lines that intersect at an angle of less than 90°, *we call the union of the two smaller (open) regions into which the plane is divided a* wedge.

Given two non-parallel short edges incident to a point p *in* P, *we say that the wedge defined by the two lines containing the edges is* free *if it does not contain any short edges incident to* p.

TWO_CIRCLE_POINT_LABELING(P)

Compute $\mathrm{DT}(P)$ and D_2 (the length of a shortest edge in $\mathrm{DT}(P)$).
Let $d = \frac{D_2}{1+\cos 18^\circ} \approx 0.513\, D_2$ be the label diameter.
Delete all long Delaunay edges (i.e. of length $\geq 2d$).
for all $p \in P$ **do**
    Compute the largest free wedge W of p.
    Place the centers of the labels of p on the bisector of W at distance $\frac{d}{2}$ from p.
end

Fig. 8. Our 2-circle point-labeling algorithm.

In order to show the correctness of our algorithm we first determine the maximum degree in the Delaunay triangulation minus the long edges and then give a lower bound for the size of the largest free wedge of a point p in P.

Fact 1. *The angle between two short edges incident to a point* p *in* P *is at least* $2 \arcsin \frac{D_2}{4d} \approx 58.4^\circ$.

Lemma 3. *Each point* p *in* P *is incident to at most six short edges.*

Proof. Consider an annular ring R with diameters D_2 and $2d$ around p (including the inner and excluding the outer circle). R contains all points in P that share a short edge with p. However, due to Fact 1, R cannot contain more than six points whose pairwise distance is at least D_2. ☍

Lemma 4. *Each point* p *in* P *has a free wedge with an angle of at least* 36°.

Proof. Due to Lemma 3 we know that p has $k \leq 6$ short edges. There are $k' \leq k$ lines that go through these edges, and there are k' wedges defined by pairs of neighboring lines (if none of them forms an angle $\geq 90°$, but then we would be done). Due to the pigeon-hole principle there must be a wedge with an angle of at least $36°$ if $k' \leq 5$. (Each wedge contributes two angles!) So we only have to consider the case $k = k' = 6$ (i.e. each line contains exactly one short edge) and all of the six wedges have an angle of less than $36°$. Then, however, one of the wedges must be delimited by two short edges on the same side of p, which is a contradiction to Fact 1. ☍

It remains to show that the labels of a point are placed within its Voronoi cell.

Lemma 5. *Let L be an open circle of diameter d that touches a point p in P. If the center of L lies on the bisector of the largest free wedge of p, then $L \subseteq \mathrm{Vor}(p)$.*

Proof. Lemma 4 guarantees that p has a free wedge W whose supporting lines s_1 and s_2 span an angle α of at least $36°$. The construction in Figure 10 shows for the extremal case $\alpha = 36°$ how the label diameter d is actually chosen, namely such that the center z of the label L of p lies at the intersection of the bisector b of W and the set F of all points with equal distance to p and a line ℓ_1. The line ℓ_1 has distance $\frac{1}{2}D_2$ from p and is perpendicular to s_1. The line ℓ_2 is defined analogously. Let x be the point that lies between p and ℓ_1 on s_1 at a distance of $\frac{D_2-d}{2}$ from p. In the right-angled triangle Δxzp we then have $\cos\frac{\alpha}{2} = \frac{D_2-d}{d}$, and thus $d = \frac{D_2}{1+\cos 18°}$. For a line h let h^+ be the open halfplane that is supported by h and contains p.

Due to our construction L is contained in the union of a disk D and a kite K, see Figure 10. The disk D is centered at p and has radius $\frac{1}{2}D_2$, thus it lies completely within $\mathrm{Vor}(p)$. The kite K is the intersection of W and the two halfplanes ℓ_1^+ and ℓ_2^+. Note that ℓ_1 and ℓ_2 touch D where they intersect the two supporting lines s_1 and s_2 of W at right angles.

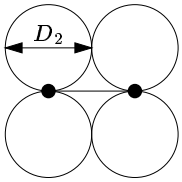


Fig. 9. D_2 is an upper bound for the optimal label size.

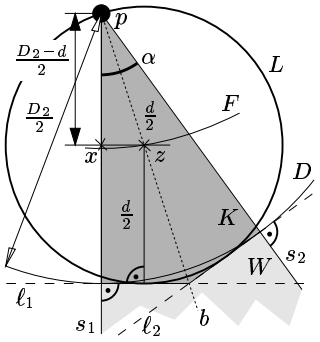


Fig. 10. We label symmetrically to a wedge W.

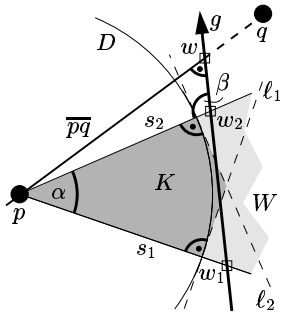


Fig. 11. No Voronoi edge intersects kite K or disk D.

Now suppose L is intersected by a Voronoi edge e of $\mathrm{Vor}(p)$. Let q be the point in P whose Voronoi cell touches that of p in e. If $d(p,q) \geq 2d$ than $d(p,e) = \frac{1}{2}d(p,q) \geq d$ and e does not intersect L. If $d(p,q) < 2d$ then q cannot lie in the wedge W, since W does not contain any short edges. So we can assume $D_2 \leq d(p,q) < 2d$ and $q \notin W$. (p,q) is a Delaunay edge: if the disk with diameter $\overline{pq}$ contained another point r of P, r would be closer than D_2 to p or q.

Let g be the line that contains e. The halfplane g^+ contains $\mathrm{Vor}(p)$ and thus D. Let w_1 and w_2 be the points where g intersects the two supporting lines s_1 and s_2 of W, see Figure 11. Let w be the intersection of g and $\overline{pq}$. w cannot lie between w_1 and w_2, otherwise q would lie in W. So we can assume that w lies outside W and closer to w_2, say. Direct g from w_1 to w_2. By definition g intersects $\overline{pq}$, the Delaunay edge between p and q, in a right angle. Then the angle β that g and s_2 form in the triangle $\Delta p w_2 w$ must be less than $90°$. Since ℓ_2 is perpendicular to s_2, this means that g intersects ℓ_2 beyond w_2—and not within W. Thus g^+ contains K, which in turn contains $L \setminus D$. This contradicts our assumption, namely that e and L intersect. ☍

Since the Voronoi cells of a point set are mutually exclusive, Lemma 5 yields the correctness of our 2-circle point-labeling algorithm. Time and space complexity follow from those of $\mathrm{DT}(P)$ and from Lemma 4. Using the same upper bound for an optimal solution as in [20, 21], we can summarize as follows.

Theorem 3. *Our algorithm labels a set of n points with $2n$ circles, two per point, of diameter at least $\frac{1}{1+\cos 18°} \approx 0.513$ times the maximum diameter in $O(n \log n)$ time using linear space.*

This algorithm uses the Delaunay triangulation to compute D_2 and to label all points with labels whose size depends on D_2. A different approach would give larger labels in general, although we were not yet able to prove a better approximation factor and keep the runtime of $O(n \log n)$. Instead of computing the Delaunay triangulation and D_2 as an upper bound for the label size, we could directly compute the Voronoi diagram, label each point optimally within its Voronoi cell, and then shrink all labels to the smallest label size we have used. If the Voronoi cell is a regular pentagon, both algorithms actually place labels of the same size.

Due to space limitations we must refer to the full paper for the proof of the following theorem. It does not only provide strong evidence for the necessity to search for approximate solutions of the 2-circle labeling problem, but it also considerably reduces the gap between our approximability result and the polynomial-time non-approximability of 2-circle labeling.

Theorem 4. *It is NP-hard to decide whether a set of points can be labeled with pairs of unit circles, and it is NP-hard to approximate the optimal label size beyond a factor of $\frac{2}{1+\sqrt{3}} \approx 0.732$.*

Acknowledgments. We wish to thank Otfried Cheong, Hong Kong University of Science and Technology, without whose generous support this research would not have been possible.

References

[1] J. Ahn and H. Freeman. AUTONAP - an expert system for automatic map name placement. In *Proc. Intl. Symp. on Spatial Data Handling*, pages 544–569, 1984.

[2] F. Aurenhammer. Voronoi diagrams: A survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3):345–405, Sept. 1991.

[3] B. Chazelle et al. Application challenges to computational geometry: CG impact task force report. Technical Report TR-521-96, Princeton University, Apr. 1996.

[4] J. Christensen, J. Marks, and S. Shieber. An empirical study of algorithms for point-feature label placement. *ACM Transactions on Graphics*, 14(3):203–232, 1995.

[5] A. Datta, H.-P. Lenhof, C. Schwarz, and M. Smid. Static and dynamic algorithms for k-point clustering problems. *J. Algorithms*, 19:474–503, 1995.

[6] S. Doddi, M. V. Marathe, A. Mirzaian, B. M. Moret, and B. Zhu. Map labeling and its generalizations. In *Proc. of the 8th ACM-SIAM Symp. on Discrete Algorithms (SODA'97)*, pages 148–157, 1997.

[7] S. Doddi, M. V. Marathe, and B. M. Moret. Point labeling with specified positions. In *Proc. 16th Annu. ACM Sympos. Comput. Geom.*, Hongkong, 2000. to appear.

[8] J. S. Doerschler and H. Freeman. An expert system for dense-map name placement. In *Proc. Auto-Carto 9*, pages 215–224, 1989.

[9] S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM J. Comput.*, 5:691–703, 1976.

[10] M. Formann. *Algorithms for Geometric Packing and Scaling Problems.* PhD thesis, Fachbereich Mathematik und Informatik, Freie Universität Berlin, 1992.

[11] M. Formann and F. Wagner. A packing problem with applications to lettering of maps. In *Proc. 7th Annu. ACM Sympos. Comput. Geom. (SoCG'91)*, pages 281–288, 1991.

[12] M. Formann and F. Wagner. An efficient solution to Knuth's METAFONT labeling problem. Manuscript available at `http://www.math-inf.uni-greifswald.de/map-labeling/papers/fw-eskml-93.ps.gz`, 1993. Freie Universität Berlin.

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

[14] K. G. Kakoulis and I. G. Tollis. On the multiple label placement problem. In *Proc. 10th Canadian Conf. Comp. Geometry (CCCG'98)*, pages 66–67, 1998.

[15] D. E. Knuth and A. Raghunathan. The problem of compatible representatives. *SIAM J. Discr. Math.*, 5(3):422–427, 1992.

[16] C. K. Poon, B. Zhu, and F. Chin. A polynomial time solution for labeling a rectilinear map. *Information Processing Letters*, 65(4):201–207, 1998.

[17] T. Strijk and A. Wolff. Labeling points with circles. Technical Report B 99-08, Institut für Informatik, Freie Universität Berlin, Apr. 1999.

[18] M. van Kreveld, T. Strijk, and A. Wolff. Point labeling with sliding labels. *Computational Geometry: Theory and Applications*, 13:21–47, 1999.

[19] F. Wagner and A. Wolff. A practical map labeling algorithm. *Computational Geometry: Theory and Applications*, 7:387–404, 1997.

[20] B. Zhu and C. K. Poon. Efficient approximation algorithms for multi-label map labeling. In *Proc. Tenth Annual Intl. Symp. on Algorithms and Computation (ISAAC'99)*, LNCS, pages 143–152, Chennai, India, 1999. Springer-Verlag.

[21] B. Zhu and Z. Qin. New approximation algorithms for map labeling with sliding labels. Dept. of Computer Science, City University of Hong Kong, 2000.

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