SERIE B — INFORMATIK

Labeling Points with Circles*

Tycho Strijk° Alexander Wolff•

B 99–08
April 1999

Abstract

We present a new algorithm for labeling points with circles of equal size. Our algorithm maximizes the label size. It improves the approximation factor of the only known algorithm for this problem by more than 50 % to about 1/20. At the same time, our algorithm keeps the $O(n \log n)$ time bound of its predecessor. In addition, we show that the decision problem is NP-hard and that it is NP-hard to approximate the maximum label size beyond a certain constant factor.

* This work was done in summer 1998 during a visit of Tycho Strijk to the Institut für Informatik, Fachbereich Mathematik und Informatik, Freie Universität Berlin, Takustraße 9, 14195 Berlin-Dahlem, Germany. It was supported by the Dutch Organization for Scientific Research (N.W.O.), the ESPRIT IV LTR Project No. 21957 (CGAL), and by the Deutsche Forschungsgemeinschaft (DFG) under grant Wa 1066/3-1.

° Department of Computer Science, Utrecht University, tycho@cs.uu.nl

• Institut für Informatik, Freie Universität Berlin, awolff@inf.fu-berlin.de

Introduction

An important task in the process of information visualization is to annotate objects on display. Objects like points, lines, and polygons in maps, diagrams, and graphs must be labeled to convey information. The interest in algorithms that automate this task has increased with the amount of information to be visualized. Cartographers, graph drawers, and computational geometers have suggested solutions. The ACM Computational Geometry Impact Task Force report [C$^+$96] lists label placement as an important research area. In computational geometry, especially the problem of point labeling has become the focus of considerable attention in recent years. This may be due to the fact that there are obvious models and objective functions for this problem. Usually labels are restricted to axis-parallel rectangles which (a) have to touch the point they label, and (b) must not intersect any other label. Condition (a) has often been restricted further in that one of a label's corners must coincide with the point to be labeled. To deal with this problem, which is NP-hard in general, approximation algorithms have been suggested maximizing either the label size, the number of points with labels, or both criteria simultaneously [FW91, AvKS98, vKSW98, DMM$^+$97].

In this article we restrict ourselves to a different label shape, namely circles of uniform size, while keeping conditions (a) and (b). We *label* a point by attaching a circle to it such that the circle's boundary contains the point. Points are not allowed to lie in the interior of a circle. Our objective is to find the largest real d_{opt} that still allows us to label *all* given points with non-overlapping circles of diameter d_{opt}. We consider our labels to be open circles, thus they may touch other points or labels.

In considering a set of three points in general position, it is clear that approximating the maximum size of circular labels cannot be reduced to the same problem for square labels. While the solution in the former case is bounded (linearly in the diameter of the point set), it is unbounded in the latter.

We show that even deciding whether a set of points can be labeled with unit circles is NP-hard, see Section 5. The same proof implies that there is a constant $\delta < 1$ such that it is NP-hard to label points with circles of diameter greater $\delta \cdot d_{\mathrm{opt}}$. Nevertheless, the maximization problem has already been approximated. Doddi et al. suggested a simple algorithm which places circles with a diameter of at least $1/(4(2+\sqrt{3})) \approx 1/14.93$ times the optimum and takes $O(n \log n)$ time [DMM$^+$97]. However, a careful revision of their proof shows that their algorithm is actually worse by a factor of 2; i.e. the label diameter is only $1/(8(2+\sqrt{3})) \approx 1/29.86$ times the optimum. In this paper, we present an algorithm with an approximation factor of $(15-\sqrt{33})/(48(2+\sqrt{3})) \approx 1/19.35$. While the analysis that yields this factor becomes more involved, the algorithm remains simple.

Both algorithms first determine the smallest diameter D_3 of any three-point subset of the input points. This can be done in $O(n \log n)$ time [EE94, DLSS95]. D_3 is needed to compute the diameter of the labels, which in both cases is a constant fraction of D_3. The observation that no point set of more than two points can be labeled with circles of diameter greater than $2(2+\sqrt{3})D_3$ then yields the corresponding approximation factors.

Like the algorithm of Doddi et al., when labeling a point our algorithm only needs to know the location of the point's closest neighbor in the set of input points. However, while Doddi et al. maximize the distance between the labels of a pair of closest neighbors, we minimize it in order to use space more efficiently. This implies that they only need to know the *direction* of the closest neighbor while we also need its *distance*. Another difference is that while they label the points in arbitrary order, we exploit this order and process the points in pairs of increasing distance. In order to build and access the data structure that supplies us with this order we need no more than $O(n \log n)$ time in total. Thus our algorithm runs in $O(n \log n)$ time. It requires linear storage.

This paper is structured as follows. In Section 1, we sketch the algorithm of Doddi et alii. In Section 2 we formalize our main ideas. Then, in Section 3 we present our algorithm, analyze it in Section 4, and finally present our NP-hardness proof in Section 5.

1 Previous Work

For a (finite) set S of points in the plane, Doddi et al. define the *diameter* $\mathrm{diam}(S)$ the usual way as the maximum distance of any two points in S. They define the *k-diameter* $D_k(S)$ to be the minimum diameter over all k-element subsets of S. Then they make the following two observations. We use D_3 as shorthand for $D_3(S)$.

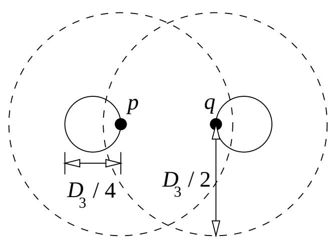

Figure 1: label placement according to the algorithm of Doddi et al.

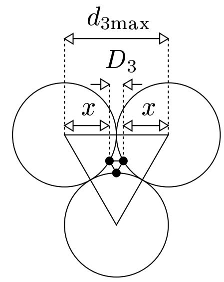

Figure 2: optimal label placement for three points

1. The open circle centered at a point $p \in S$ with radius $D_3/2$ contains at most one other point $q \in S - \{p\}$. Due to symmetry, an open circle of the same diameter centered at q only contains p and q. This allows p and q to be labeled with labels of diameter $d' = D_3/4$ as in Figure 1. Given the distance of p and q to other points in S, it is obvious that labels of other points cannot overlap those of p and q.

2. The maximum label diameter $d_{\mathrm{opt}}(S)$ of any set S of more than two points cannot exceed the maximum label diameter $d_{3\mathrm{max}}$ of three points at pairwise distance $D_3(S)$, see Figure 2. Doddi et al. compute $d_{3\mathrm{max}}$ to be $(2+\sqrt{3})D_3$, but this is incorrect: we have $d_{3\mathrm{max}} = 2x + D_3$, where $x = (d_{3\mathrm{max}}/2) \cdot \cos \pi/6$ since the triangles are equilateral. Thus we obtain $d_{3\mathrm{max}} = 2(2+\sqrt{3})D_3 \approx 7.46\ D_3$ as an upper bound for d_{opt}.

Combining these observations yields the approximation factor $d'/d_{\mathrm{opt}} \geq 1/(8(2+\sqrt{3})) \approx 1/29.86$ of their algorithm.

2 Preliminaries

We formalize the idea of the free space around a point as follows.

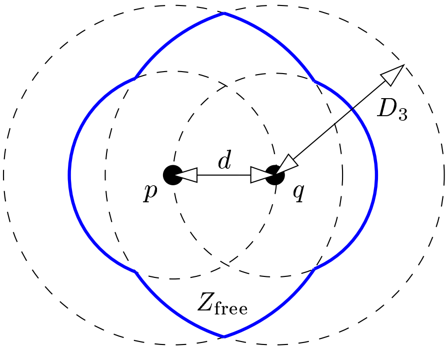

Figure 3: the point-free zone Z_{free} of p and q

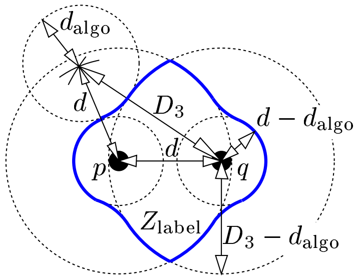

Figure 4: the label zone Z_{label} of p and q

Definition 1 *Let $C_{m,r}$ be an open circle with radius r centered at m. We say that two points are a* pair of closest neighbors *if each is a closest neighbor of the other. Given two points $p, q \in S$, we denote the* point-free zone *of p and q by*

$$Z_{\text{free}}(p,q) = (C_{p,D_3} \cap C_{q,D_3}) \cup C_{p,d} \cup C_{q,d} \subseteq \mathbb{R}^2$$

where $d = d(p,q)$ is the Euclidean distance of p and q.

The definition is illustrated in Figure 3. We show that the point-free zone of a pair of closest neighbors $\{p,q\}$ does not contain any other point of S. This will enable us to use part of the zone for labeling p and q.

Lemma 2 *Let $\{p,q\}$ be a pair of closest neighbors in S. Then $Z_{\text{free}}(p,q) \cap S = \{p,q\}$.*

Proof. Suppose $Z_{\text{free}}(p,q)$ contains a point $t \in S \setminus \{p,q\}$. Then $t \in C_{p,D_3} \cap C_{q,D_3}$ or $t \in C_{p,d} \cup C_{q,d}$. In the first case the diameter of $\{p,q,t\}$ would be less than D_3; a contradiction to the definition of D_3. The second case would contradict $\{p,q\}$ being closest neighbors. ◯

Note that Doddi et al. implicitly also used the concept of a point-free zone, namely the union of the dashed circles $C_{p,D_3/2}$ and $C_{q,D_3/2}$ depicted in Figure 1. However, their zone is always contained in our point-free zone Z_{free}, independently of d. This helps us to place larger labels. Let d_{algo} be the diameter of the labels our algorithm is going to place.

Definition 3 *Given two points $p, q \in S$ and a real number $d_{\text{algo}} < D_3$. Then we denote the* label zone *of p and q by*

$$Z_{\text{label}}(p,q;d_{\text{algo}}) = Z_{\text{free}}(p,q) - \bigcup_{x \in \mathbb{R}^2 - Z_{\text{free}}(p,q)} C_{x,d_{\text{algo}}} = Z_{\text{free}}(p,q) \ominus C_{0,d_{\text{algo}}}$$

where $\ominus$ is the Minkovski subtraction operator and 0 is the origin.

In other words, the label zone is the erosion of the point-free zone with a disk of radius d_{algo}. The definition is illustrated in Figure 4.

Lemma 4 *If we label a pair of closest neighbors $\{p, q\}$ with circles of diameter d_{algo} that are contained in the label zone $Z_{\text{label}}(p, q; d_{\text{algo}})$, then these labels cannot overlap the label of any other point $t \in S - \{p, q\}$.*

Proof. Suppose the label of t overlaps that of p. Lemma 2 tells us that $t \notin Z_{\text{free}}(p, q)$. Then the definition of the label zone ensures that $C_{t,d_{\text{algo}}}$ does not intersect $Z_{\text{label}}(p, q; d_{\text{algo}})$. Observing that t's label is contained in $C_{t,d_{\text{algo}}}$ and p's label in $Z_{\text{label}}(p, q; d_{\text{algo}})$ contradicts our assumption. ∎

The question is, of course, how large we can choose d_{algo} so that the labels of any pair of closest neighbors fit into their label zone — independently of their distance. This question is dealt with in Section 4. Let us suppose for the moment that d_{algo} can be expressed as a fraction of D_3. The next section answers two other important questions, namely how do we place the labels, and in which order.

3 Algorithm

Our algorithm proceeds as described in Figure 5.

LABEL_POINTS_WITH_CIRCLES(S)

compute $D_3(S)$
$d_{\text{algo}} := (15 - \sqrt{33})/(48(2 + \sqrt{3}))\ D_3(S)$
while $|S| > 1$
    choose $\{p, q\} \subseteq S$ with $d(p, q)$ minimal
    if $d(p, q) \geq 2d_{\text{algo}}$ **then** exit while-loop **endif**
    label p and q as in Figure 6
    $S := S \setminus \{p, q\}$
end
for all $x \in S$ **do** label x arbitrarily **end**
return all label positions

Figure 5: our algorithm

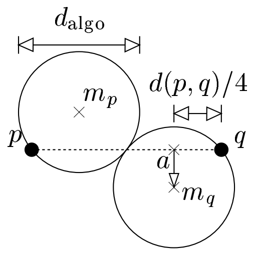

Figure 6: labeling a pair of points

In contrast to Doddi et al. who place the labels of two neighboring points as far apart from each other as possible, we label p and q such that their labels are as close as possible. This means that they will touch each other as in Figure 6 if $d(p, q) \leq 2d_{\text{algo}}$. The vectors $\vec{p}$ and $\vec{q}$ denote the coordinates of p and q in the plane. We place the center $\vec{m_q}$ of q's label at $\vec{m_q} = \vec{p}/4 + 3\vec{q}/4 + \vec{a}$. The vector $\vec{a}$ is perpendicular to $\overline{pq}$ and oriented so that it points to the left of $(\vec{p} - \vec{q})$. The length of $\vec{a}$ is $\|\vec{a}\| = \sqrt{d_{\text{algo}}^2/4 - d^2(p, q)/16}$. Correspondingly, the center of p's label is placed at $\vec{m_p} = \vec{q}/4 + 3\vec{p}/4 - \vec{a}$. We call the union of these two (open) labels the *label space* of p and q. If there are unlabeled points left after executing the while-loop, we label them arbitrarily.

Lemma 5 *Given a set S of n points and a label diameter d_{algo} such that the label space is contained by the label zone for any pair of points in S. Then the labels our algorithm places do not intersect.*

Proof. The fact that no two labels overlap follows from the order in which we process the points. It is clear that the first pair $\{p,q\}$ is a pair of closest neighbors. Due to Lemma 4, we know that we do not constrain the labeling of any other point in S when we label such a pair within its label zone. In other words, if we remove $\{p,q\}$ from S, then we can ignore p and q as well as their labels for solving the remaining problem. The next pair of points will be a pair of closest neighbors in the reduced set S. Thus Lemma 4 applies to this pair as well.

After we leave the while-loop, there might be unlabeled points left. For each such point x there are two possibilities. Either its closest neighbor in the original set is at least $2d_{\text{algo}}$ away. Or all points y with $d(x,y) < 2d_{\text{algo}}$ have been labeled before, since each had a closer neighbor z. In either case the labeling of x is not constrained by any previous label placement. Hence we can label x arbitrarily. ❍

Lemma 6 *The algorithm can be implemented such that it labels a set S of n points in $O(n \log n)$ time with linear space.*

Proof. Our algorithm labels S in three phases. In the first phase, we compute D_3 in $O(n \log n)$ time with one of the algorithms suggested in [EE94, DLSS95]. We need D_3 to compute the diameter $d_{\text{algo}} = (15 - \sqrt{33})/24\ D_3 \approx 0.39\ D_3$ of the labels we are going to place, see Section 4.

In the second phase, we set up a simple data structure, which will answer a limited closest pair query; limited in the sense that we only need to know pairs of points closer than $2d_{\text{algo}}$.

Since d_{algo} is a fraction of D_3, an axis-parallel square of edge length $2d_{\text{algo}}$ centered at a point $p \in S$ can contain at most a constant number of points in S. We call these points the *relevant neighbors* of p. If p had more than a constant number of relevant neighbors, then the square centered at p would contain three points with diameter less than D_3 — a contradiction to the definition of D_3. This observation enables us to collect the relevant neighbors for every point in S with a sweep-line algorithm that detects the k nearest neighbors according to the maximum norm for each of the n input points in $O(kn + n \log n)$ total time using linear space [For92]. After the sweep, we put the set of all pairs of relevant neighbors $\{\{p,q\} \mid p \text{ is relevant neighbor of } q\}$ into a priority queue, each pair $\{p,q\}$ according to the Euclidean distance of p and q. This takes $O(n \log n)$ time.

In the third phase, we repeatedly extract the minimum $\{p,q\}$ of the priority queue and check whether $d(p,q) < 2d_{\text{algo}}$. If this is the case, we label p and q with circles of diameter d_{algo} as in Figure 6, and delete all pairs containing either p or q from the queue. The points remaining in S after this process can be labeled in constant time per point. Phase 3 can also be done in $O(n \log n)$ steps.

Summing up the running times of the three phases gives a time bound of $O(n \log n)$. The necessary data structures can all be implemented such that they do not need more than linear space. ❍

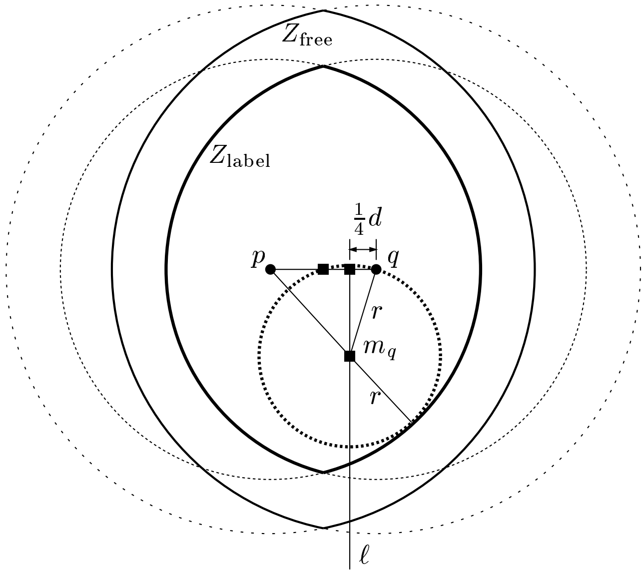

Figure 7: Point-free zone Z_{free} and label zone Z_{label} of p and q for $d \le D_3/2$.

4 Analysis

Given a pair $\{p, q\}$ of closest neighbors in S, our objective now is to compute the maximum radius r of their labels so that the label space is still contained in the label zone $Z_{\text{label}}(p, q; 2r)$ of p and q. Since this radius will only depend on the distance d of p and q, we want to find the minimum $r_{\min}$ of $r(d)$, set d_{algo} to $2r_{\min}$ and run our algorithm. Lemma 5 guarantees that no two labels will intersect then.

Since we place the labels of p and q symmetrically, it is enough to analyze the placement of q's label. We have to consider two cases depending on p and q's distance.

In case $d \le D_3/2$, the point-free zone $Z_{\text{free}}(p, q)$ is the intersection of C_{p,D_3} and C_{q,D_3}. The corresponding label zone $Z_{\text{label}}(p, q; 2r)$ is the intersection of C_{p,D_3-2r} and C_{q,D_3-2r}. As described in the previous section, the label has its center point m_q on a line ℓ, normal to the line connecting p and q. This normal line has distance $d/4$ to q. The distance of m_q to the line $\overline{pq}$ is $\sqrt{r^2 - d^2/16}$ using Pythagoras' rule.

The radius of q's label is maximized when the label touches the boundary of the label zone, i.e. the circle C_{p,D_3-2r}. We use the property that the touching point of two circles always lies on the line through their centers, see Figure 7.

This observation yields the equality

$$d(p, m_q) + r = D_3 - 2r. \tag{1}$$

We use that $d(p, m_q) = \sqrt{(\frac{3}{4}d)^2 + (r^2 - d^2/16)} = \sqrt{d^2/2 + r^2}$. The resulting equation $\sqrt{d^2/2 + r^2} = D_3 - 3r$ is quadratic and has two solutions. The solution valid for our problem is

$$r = \frac{3D_3 - \sqrt{D_3^2 + 4d^2}}{8}. \tag{2}$$

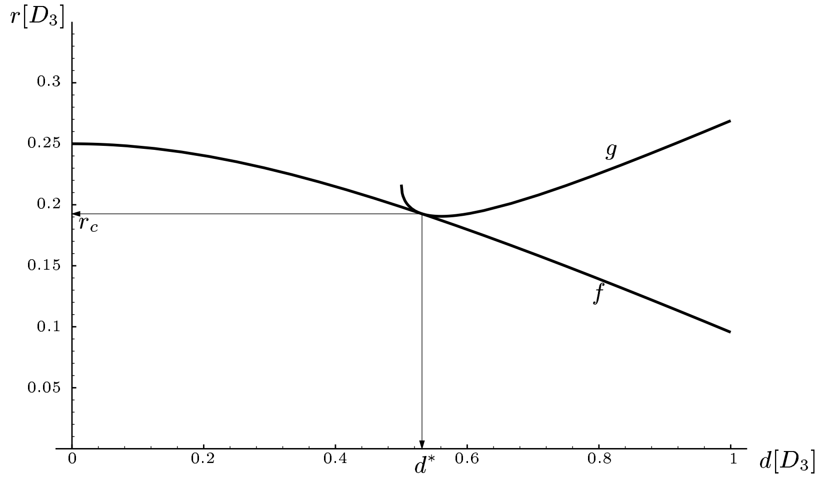

Figure 8: The graph of $r(d)$ is determined by the functions f and g. For $d < d^* \approx 0.53$, the value of r is determined by f, otherwise by g.

The graph f of this function is depicted in Figure 8 with both d and r scaled proportionally to D_3.

The case $d > D_3/2$ is more difficult. In this case, the point-free zone $Z_{\text{free}}(p,q)$ is the union of three areas, namely $C_{p,D_3} \cap C_{q,D_3}$, $C_{p,d}$, and $C_{q,d}$, see Figure 9.

Accordingly, the boundary of $Z_{\text{label}}(p,q;2r)$ consists of arc segments that are part of the circles C_{p,D_3-2r}, C_{q,D_3-2r}, $C_{p,d-2r}$, $C_{q,d-2r}$ and four circles with radius $2r$ centered at the intersections of C_{p,D_3} with $C_{q,d}$ and at the intersections of C_{q,D_3} with $C_{p,d}$. One of these last four circles is $C_{m_\times,2r}$ whose center $m_\times$ lies at an intersection of C_{p,D_3} and $C_{q,d}$, see Figure 9.

It turns out that a label with maximum radius inside the label zone always touches either C_{p,D_3-2r} or $C_{m_\times,2r}$. This depends on the value of d. There is a real $d^* = \frac{\sqrt{1+\sqrt{33}}}{2\sqrt{6}} D_3 \approx 0.53 D_3$ such that for $d \leq d^*$ the label touches C_{p,D_3-2r} and for $d \geq d^*$ it touches $C_{m_\times,2r}$.

The values of r for which the label touches C_{q,D_3-2r} have already been computed, see Equation (2). The values, for which the label touches $C_{m_\times,2r}$, are computed similarly as follows. The line connecting the center points m_q and $m_\times$ intersects the touching point of the two circles. This gives rise to the equation

$$d(m_q, m_\times) = 3r, \tag{3}$$

see Figure 9. In the following we assume that $D_3 = 1$. The point set can always be scaled so that this is true.

If we put the origin of our coordinate system at q and let the negative x-axis contain p, then

$$m_\times = \left(\frac{1-2\,d^2}{2\,d}, \frac{-\sqrt{4-d^{-2}}}{2}\right) \text{ and } m_q = \left(-\frac{1}{4}d, -\sqrt{r^2 - d^2/16}\right). \tag{4}$$

The equation $d(m_q, m_\times) = 3r$ has the following solution for our problem:

$$r = \frac{\sqrt{8 - d^{-2} + 8\,d^2 - \frac{\sqrt{1-16\,d^2+48\,d^4}}{d^2}}}{8\sqrt{2}}. \tag{5}$$

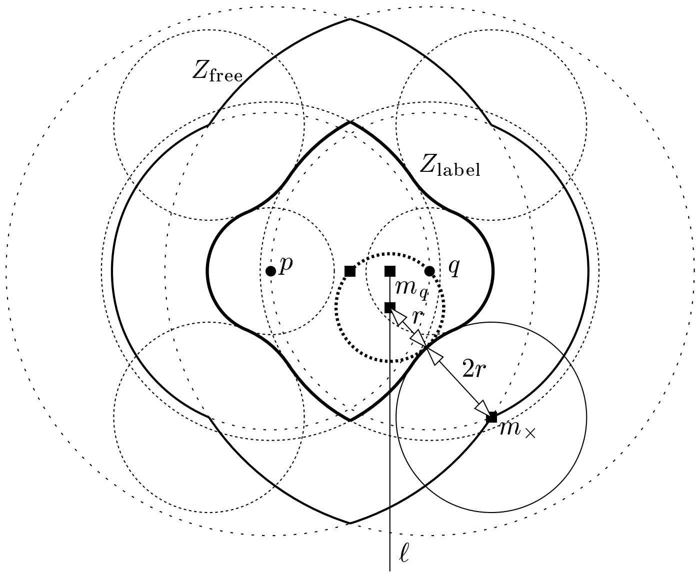

Figure 9: The point-free zone Z_{free} and the label zone Z_{label} of p and q for the case $d > D_3/2$. The bold dotted label touches the circle centered at $m_\times$.

The graph g of this function is depicted in Figure 8. The graph of $r(d)$ equals $f(d)$ for $d < d^*$ and $g(d)$ otherwise. The minimum $r_{\min}$ of $r(d)$ is reached at d^* since f decreases and g increases strongly monotonically on the corresponding interval. We obtain $r_{\min} = (15 - \sqrt{33})/48 \approx 0.19$.

Theorem 7 *Our algorithm labels a point set S with circles of diameter $d_{\text{algo}} = 2r_{\min} = (15 - \sqrt{33})/24 \approx 0.39\ D_3(S)$.*
The approximation factor γ is $(15 - \sqrt{33})/(48(2 + \sqrt{3})) \approx 1/19.35$.

Proof. Since the minimum $r_{\min}$ of $r(d)$ occurs when $d = d^*$, we know that the label space of any pair of closest neighbors will be contained by its label zone. Then Lemma 5 ensures that for a label diameter $d_{\text{algo}} = 2r_{\min}$, our algorithm will label all points with non-overlapping labels. The approximation factor is the ratio of the upper bound $2(2+\sqrt{3})D_3$ (see Figure 2) and the diameter d_{algo} of the labels we place. ◶

It is clear that any set of congruent equilateral triangles will force our algorithm to produce a labeling with circles of diameter $\gamma \cdot d_{\text{opt}}$ if the triangles are spaced appropriately. However, there are also examples with a smaller optimum labeling where the algorithm performs better. The triangular lattice formed by the centers of a densest disk packing, for example, has an optimal labeling with circles of diameter $d_{\text{opt}} = D_3$. Here our algorithm yields a ratio of $d_{\text{algo}}/d_{\text{opt}} \approx 0.39$.

5 NP-Hardness

Theorem 8 *It is NP-hard to decide whether a set of points can be labeled with unit circles.*

Proof. Our proof is by reduction from planar 3-SAT. Lichtenstein showed that this restriction of 3-SAT is NP-hard [Lic82]. Our proof follows Knuth and Raghunathan's proof of the NP-hardness of the metafont labeling problem [KR92]. We encode the variables and clauses of a boolean formula ϕ of planar 3-SAT type by a set of points such that all points can only be labeled if ϕ is satisfiable, i.e. if there is a variable assignment such that all clauses evaluate to true. The advantage of a *planar* 3-SAT formula is that there is always a way to arrange nodes corresponding to the variables on a straight line in the plane and connect these nodes by *non-intersecting* three-legged clauses, see [KR92, Figure 5].

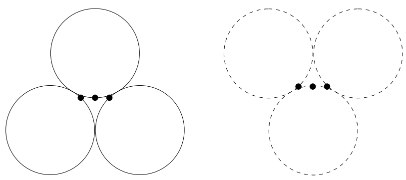

Figure 10: Label placements encoding *true* and *false*.

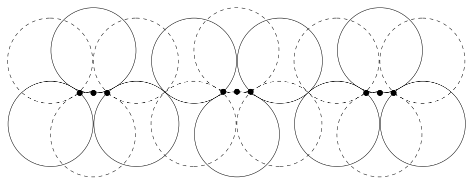

Figure 11: Rows of three-point clusters model a variable; the label positions of the leftmost cluster determine the variable's Boolean value.

The main observation leading to our proof is the following. Given three equidistant[1] points on a line, there are exactly two ways to label these points optimally, see Figure 10. This gives us a means to encode the Boolean values of a variable in the planar 3-SAT formula ϕ that we want to reduce to a set of points.

The building blocks (or "gadgets") of our reduction are the clusters for variables and three-legged "combs" for clauses. In order to be able to connect a variable to all clauses in which it occurs, we model it not by one but by several *variable clusters* on a horizontal line as in Figure 11. Note that the cluster-cluster distance of $1+\sqrt{2\sqrt{3}-3}$ (from mid-point to mid-point) is chosen such that every second cluster must be labeled the same way. Let the label position of the leftmost cluster represent a variable's value according to Figure 10.

The central idea for modeling the clauses is that we restrict the possible label positions of all points (except one) to a maximum of two. To achieve this, we use *immobilizing clusters* that can only be labeled in one specific way. They consist of three points at pairwise distance $1/(4+2\sqrt{3})$, see Figure 2. In order to distinguish these auxiliary points from the others, we use the term *active points* for all clause points with at least two possible label positions.

We model the clauses by point sets which resemble large combs, see Figure 12. Such a comb consists of a horizontal part and three legs. The horizontal part is formed by points like b in Figure 12. These points lie on a horizontal line. The immobile dotted labels of the immobilizing clusters above and below restrict the label of b to two possible positions.

The legs consist of points like a in Figures 12 and 13. These points lie on a vertical line and are also forced into one of two possible positions. At the junctions where the legs are joined to the horizontal part of a clause lack of space prevents us from using the immobilizing clusters as elsewhere. Instead, we simply attach points like x, y, and z in Figure 12 to cluster labels in the vicinity such that the labels of x, y, and z are also

[1]Since all labels have diameter 1 here, some basic geometry shows that this distance must be $(1-\sqrt{2\sqrt{3}-3})/2$

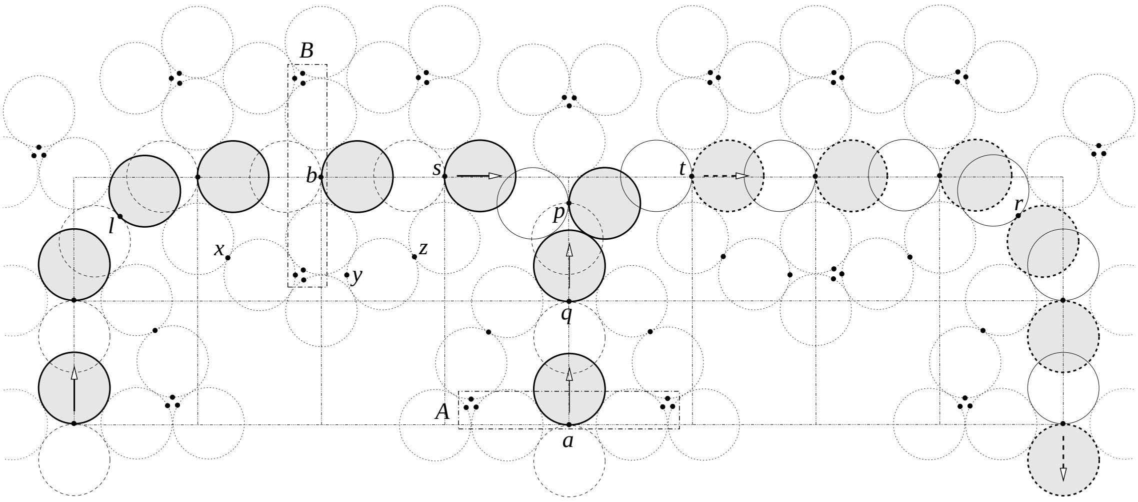

Figure 12: Clause with pressure from two variables. The current label positions are marked by shaded labels, alternatives are indicated by solid or dashed circles, and immobile labels are dotted.

immobile and at the same time restrict the label positions of points close to the junction (like l, r, s, t, and q) as desired.

Both the horizontal part and the legs of a comb can be extended as far as needed to reach the three variables belonging to the clause. This is done by repeating — at a distance of $\sqrt{3}$ — patterns of seven points like those contained in the boxes A and B in Figures 12.

Each leg is connected to a variable v. The connection depends on the literal; i.e. on whether or not v is negated in the clause. Let g be the vertical, on which all points of the leg lie. If v is negated, we connect the leg to v such that g contains a point of type d. The distance of $\sqrt{3}$ between the lowest leg point a and d is chosen to assure that the label of a can only intersect the label of d among the points modeling v. Note that the labels of a and d only intersect if the label of a is placed right below a and that of d right above d. d is the mid-point of a variable cluster that is labeled the same way as the leftmost cluster of v, see Figure 13. Hence d is labeled upwards if v is set to true and is negated in the clause. Then a and all other points on g must also be labeled upwards. To put it graphically, pressure is transmitted upwards. If v is negated and set to false, d can be labeled downwards, and no pressure is transmitted.

On the other hand, if v is not negated in the clause under consideration, we connect the clause's leg to v so that g contains a point of type d'. Such points belong to clusters labeled the opposite way as the leftmost cluster. Then pressure is transmitted if and only if v is set to false.

If all literals of a clause evaluate to false, then pressure is transmitted through all three legs into the clause. In this case there is a point (like p) that cannot be labeled, see Figure 12. In case there is at least one leg without pressure, it is obvious that all points belonging to a clause can be labeled.

Hence the question whether ϕ can be satisfied is equivalent to asking whether all points in the set, to which ϕ is reduced, can be labeled with unit circles.

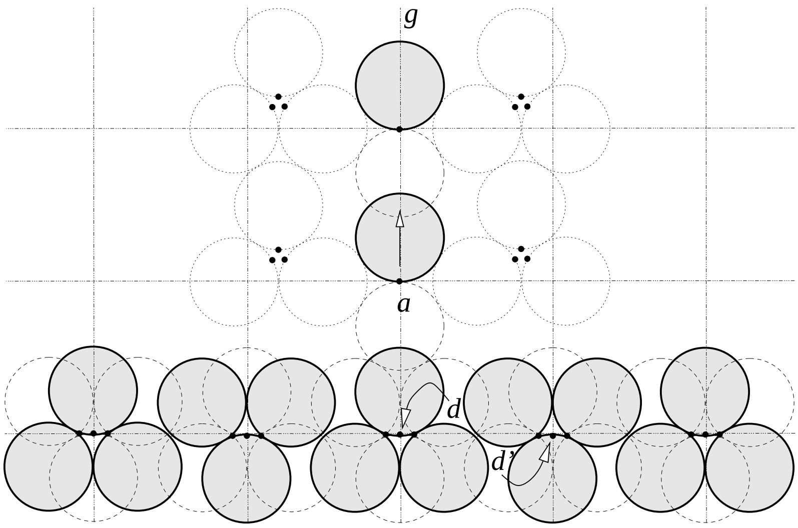

Figure 13: We connect the leg of a clause to a variable above points of type d or d'. This depends on whether the variable is negated in the clause (as in this case) or not.

To show that we can in fact connect a variable to several different clauses (below and above), we use a grid of width $\sqrt{3}$ and move all active points to grid points — except points of type l, r, and p, see Figure 12. In order to accommodate also the mid-points of the variable clusters on the grid, we slightly increase the distance of neighboring variable clusters to that of immobilizing clusters, i.e. from $1 + \sqrt{2\sqrt{3} - 3} \approx 1.68$ to $\sqrt{3} \approx 1.73$. Then the label positions in every second cluster must still be combinatorically equivalent, although labels can slightly wiggle now, see Figure 13. Given such a grid embedding of our instance, it is clear that we can connect all variables to the clauses according to ϕ.

We still have to ensure that the reduction is polynomial: if ϕ consists of m clauses and $n \leq 3m$ variables, the instance has $O(m^2)$ points. Their position can be computed in polynomial time if we round the grid-cell size and the distance between the three points of the immobilizing clusters to slightly greater rational numbers. The resulting instance is combinatorically equivalent to the one described before. ◯

It is not clear whether the problem is in $\mathcal{NP}$. We do not know whether there is always a polynomial encoding of a solution, even if the input points have rational coordinates.

Corollary 9 *There is a constant $\delta < 1$ such that it is NP-hard to label points with uniform circles of diameter greater $\delta \cdot d_{\text{opt}}$.*

Proof. The proof of Theorem 8 still holds if the diameter of all labels is slightly reduced to a $\delta < 1$. The reason for this is that though labels have a certain degree of freedom now, every new label position is combinatorically equivalent to exactly one former position. δ must be chosen close enough to 1 to prevent a label from being moved continuously from one old position to another.

If there was a polynomial-time algorithm that could label the point set of the reduction with labels of size δ, we would have $\mathcal{P} = \mathcal{NP}$. Formann and Wagner used a similar argument to show that maximizing the size of axis-parallel square labels cannot be approximated beyond 1/2 [FW91]. ❍

The bottleneck that determines the minimum value of δ seems to be the encoding of a variable, see Figure 11. When the labels (and thus δ) are scaled down gradually, there is a point when two neighboring clusters can have identical instead of alternating label positions, see Figure 10. Then the variable's Boolean value is no longer well defined, and the proof collapses.

Conclusions

This paper has proved the NP-completeness of the circle labeling problem and also the NP-hardness of approximating it beyond a constant $\delta < 1$. We have presented an algorithm to approximate the circle labeling problem. This algorithm improves the approximation factor of the existing algorithm by more than 50 per cent.

References

[AvKS98] Pankaj K. Agarwal, Marc van Kreveld, and Subhash Suri. Label placement by maximum independent set in rectangles. *Computational Geometry: Theory and Applications*, 11:209–218, 1998.

[C+96] Bernard Chazelle et al. Application challenges to computational geometry: CG impact task force report. Technical Report TR-521-96, Princeton University, April 1996. `http://www.cs.princeton.edu/~chazelle/taskforce/CGreport.ps`.

[DLSS95] Amitava Datta, Hans-Peter Lenhof, Christian Schwarz, and Michiel H. M. Smid. Static and dynamic algorithms for k-point clustering problems. *J. Algorithms*, 19:474–503, 1995.

[DMM+97] Srinivas Doddi, Madhav V. Marathe, Andy Mirzaian, Bernard M.E. Moret, and Binhai Zhu. Map labeling and its generalizations. In *Proceedings of the 8th ACM-SIAM Symposium on Discrete Algorithms*, pages 148–157, 1997.

[EE94] David Eppstein and Jeff Erickson. Iterated nearest neighbors and finding minimal polytopes. *Discrete Comput. Geom.*, 11:321–350, 1994.

[For92] Michael Formann. *Algorithms for Geometric Packing and Scaling Problems.* PhD thesis, Fachbereich Mathematik und Informatik, Freie Universität Berlin, 1992.

[FW91] Michael Formann and Frank Wagner. A packing problem with applications to lettering of maps. In *Proc. 7th Annu. ACM Sympos. Comput. Geom.*, pages 281–288, 1991.

[KR92] Donald E. Knuth and Arvind Raghunathan. The problem of compatible representatives. *SIAM J. Discr. Math.*, 5(3):422–427, 1992.

[Lic82] David Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982.

[vKSW98] Marc van Kreveld, Tycho Strijk, and Alexander Wolff. Point set labeling with sliding labels. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, pages 337–346, June 1998.