Approximation Algorithms for Spreading Points

Sergio Cabello

institute of information and computing sciences, utrecht university

technical report UU-CS-2003-040

www.cs.uu.nl

Approximation Algorithms for Spreading Points *

Sergio Cabello
Institute of Information and Computing Sciences
Universiteit Utrecht
The Netherlands
sergio@cs.uu.nl

Abstract

We consider the problem of placing n points, each one inside its own, prespecified disk, with the objective of maximizing the distance between the closest pair of them. The disks can overlap and have different sizes. The problem is NP-hard and does not admit a PTAS. In the L_∞ metric, we give a 2-approximation algorithm running in $O(n\sqrt{n}\log^2 n)$ time. In the L_2 metric, we give a quadratic time algorithm that gives an $\frac{8}{3}$-approximation in general, and a ~ 2.2393-approximation when all the disks are congruent.

1 Introduction

The problem of distant representatives was recently introduced by Fiala et al. [12, 13]: given a collection of subsets of a metric space and a value $\delta > 0$, we want a representative of each subset such any two representatives are at least δ apart. They introduced this problem as a variation of the problem of systems of disjoint representatives in hypergraphs [3]. It generalizes the problem of systems of distinct representatives, and it has applications in areas such as scheduling or radio frequency (or channel) assignment to avoid interferences.

As shown by Fiala et al. [12, 13], and independently by Baur and Fekete [4], the problem of deciding the existence of distant representatives is NP-hard even in the plane under natural metrics. Furthermore, in most applications, rather than systems of representatives at a given distance, we would be more interested in systems of representatives whose closest pairs are as separated as possible. Therefore, the design of approximation algorithms for the latter problem seems a suitable alternative.

Here, we consider the problem of maximizing the distance of the closest pair in systems of representatives in the plane with either the L_∞ or the Euclidean L_2 metric. The subsets that we consider are (possibly intersecting) disks.

This geometric optimization problem finds applications in cartography [7], graph drawing [8], and more generally in data visualization, where the readability of the displayed data is a basic requirement, and often a difficult task. In many cases, there are some restrictions on how and where each object has to be drawn, as well as some freedom. For example, cartographers improve the readability of a map by displacing some features with respect to their real position. The displacement has to be small to preserve correctness. A similar problem arises when displaying a molecule in the plane: the exact position of each atom is

*Revised in April 2004. Partially supported by Cornelis Lely Stichting, NWO, and DIMACS.

not known, but instead, we have a region where each atom is located. In this case, we also have some freedom where to draw each atom, and a thumb rule tells that the drawing of the molecule improves as the separation between the atoms increases. In both applications, the problem can be abstracted as follows. We want to place a fixed number of points (0-dimensional cartographic features or atoms) in the plane, but with the restriction that each point has to lie inside a prespecified region. The regions may overlap, and we want the placement that maximizes the distance between the closest pair. The region where each point has to be placed is application dependent. We will assume that they are given, and that they are disks.

Formulation of the problem. Given a distance d in the plane, consider the function $D : (\mathbb{R}^2)^n \to \mathbb{R}$ that gives the distance between a closest pair of n points

$$D(p_1, \ldots, p_n) = \min_{i \neq j} d(p_i, p_j).$$

Let $\mathcal{B} = \{B_1, \ldots, B_n\}$ be a collection of (possibly intersecting) disks in $\mathbb{R}^2$ under the metric d. A *feasible* solution is a placement of points $p_1, \ldots, p_n$ with $p_i \in B_i$. We are interested in a feasible placement $p_1^*, \ldots, p_n^*$ that maximizes D

$$D(p_1^*, \ldots, p_n^*) = \max_{(p_1,\ldots,p_n) \in B_1 \times \cdots \times B_n} D(p_1, \ldots, p_n).$$

We use $D(\mathcal{B})$ to denote this optimal value.

A t-approximation, with $t \geq 1$, is a feasible placement $p_1, \ldots, p_n$, with $t \cdot D(p_1, \ldots, p_n) \geq D(\mathcal{B})$. We will use $B(p, r)$ to denote the disk of radius r centered at p. Recall that under the L_∞ metric, $B(p, r)$ is an axis-aligned square centered at p and side length $2r$. We assume that the disk B_i is centered at c_i and has radius r_i, so $B_i = B(c_i, r_i)$.

Related work. The decision problem associated to our optimization one is the original distant representatives problem: for a given value δ, is $D(\mathcal{B}) \geq \delta$? Fiala et al. [12, 13] showed that this problem is NP-hard in the Euclidean and Manhattan metrics. Furthermore, their result can be modified to show that, unless $NP = P$, there is a certain constant $T > 1$ such that no T-approximation is possible. They also notice that the one dimensional problem can be solved using the scheduling algorithm by Simons [22].

Closely related are *geometric dispersion* problems: we are given a polygonal region of the plane and we want to place n points on it such that the closest pair is as far as possible. This problem has been considered by Baur and Fekete [4] (see also [6, 11]), where both inapproximability results and approximation algorithms are presented. Their NP-hardness proof and inapproximability results can easily be adapted to show inapproximability results for our problem, showing also that no polynomial time approximation scheme is possible, unless $P = NP$.

In a more general setting, we can consider the following problem: given a collection $S_1, \ldots, S_n$ of regions in $\mathbb{R}^2$, and a function $f : S_1 \times \cdots \times S_n \to \mathbb{R}$ that describes the quality of a feasible placement $(p_1, \ldots, p_n) \in S_1 \times \cdots \times S_n$, we want to find a feasible placement $p_1^*, \ldots, p_n^*$ such that

$$f(p_1^*, \ldots, p_n^*) = \max_{(p_1,\ldots,p_n) \in S_1 \times \cdots \times S_n} f(p_1, \ldots, p_n).$$

metric	regions	approximation ratio	running time
L_∞	arbitrary disks	2	$O(n\sqrt{n}\log^2 n)$
L_2	arbitrary disks	$\frac{8}{3}$	$O(n^2)$
	congruent disks	~ 2.2393	$O(n^2)$

Table 1: Approximation algorithms for the plane in this paper.

Geometric dispersion problems are a particular instance of this type where we want to maximize the function D over k copies of the same polygonal region. In [5], given a graph on the vertices $p_1, \ldots, p_n$, placements that maximize the number of straight-line edges in a given set of orientations are considered.

Our results. A summary of our approximation algorithms is given in Table 1. The main idea in our approach is to consider an "approximate-placement" problem in the L_∞ metric: given a value δ that satisfies $2\delta \leq D(\mathcal{B})$, we can provide a feasible placement $p_1, \ldots, p_n$ such that $D(p_1, \ldots, p_n) \geq \delta$. The proof can be seen as a suitable packing argument. This placement can be computed in $O(n\sqrt{n}\log n)$ time using the data structure by Mortensen [19] and the technique by Efrat et al. [9] for computing a matching in geometric settings. See Section 2 for details.

We then combine the "approximate-placement" algorithm with the geometric features of our problem to get a 2-approximation in the L_∞ metric. This can be achieved by paying an extra logarithmic factor; see Section 3.

The same techniques can be used in the L_2 metric, but the approximation ratio becomes $8/3$ and the running time increases to $O(n^2)$. However, when we restrict ourselves to congruent disks, a trivial adaptation of the techniques gives an approximation ratio of ~ 2.2393. This is explained in Section 4. We conclude in Section 5

2 A placement algorithm in L_∞

Consider an instance $\mathcal{B} = \{B_1, \ldots, B_n\}$ of the problem in the L_∞ metric, and let $\delta^* = D(\mathcal{B})$ be the maximum value that a feasible placement can attain. We will consider the "approximate-placement" problem that follows: given a value δ, we provide a feasible placement $p_1, \ldots, p_n$ such that, if $\delta \leq \frac{1}{2}\delta^*$ then $D(p_1, \ldots, p_n) \geq \delta$, and otherwise there is no guarantee on the placement. We start by presenting an algorithm and discussing its approximation performance. Then we discuss a more efficient version of it.

2.1 Algorithm and its approximation ratio

Let $\Lambda = \mathbb{Z}^2$, that is, the lattice $\Lambda = \{(a, b) \,|\, a, b \in \mathbb{Z}\}$. For any $\delta \in \mathbb{R}$ and any point $p = (p_x, p_y) \in \mathbb{R}^2$, we define $\delta p = (\delta p_x, \delta p_y)$ and $\delta\Lambda = \{\delta p \,|\, p \in \Lambda\}$. Observe that $\delta\Lambda$ is also a lattice. The reason to use this notation is that we can use $p \in \Lambda$ to refer to $\delta p \in \delta\Lambda$ for different values of δ. An *edge* of the lattice $\delta\Lambda$ is a horizontal or vertical segment joining two

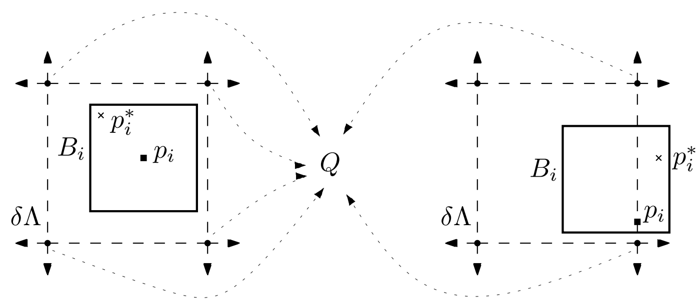

Figure 1: Special cases where the disk B_i does not contain any lattice point. Left: B_i is fully contained in a cell of $\delta\Lambda$. Right: B_i intersects an edge of $\delta\Lambda$.

points of $\delta\Lambda$ at distance δ. The edges of $\delta\Lambda$ divide the plane into squares of side length δ, which we call the *cells* of $\delta\Lambda$.

The idea is that whenever $2\delta \leq \delta^*$, the lattice points $\delta\Lambda$ almost provide a solution. However, we have to treat as a special case the disks with no lattice point inside. More precisely, let $Q \subset \delta\Lambda$ be the set of points that cannot be considered as a possible placement because there is another already placed point too near by. Initially, we have $Q = \emptyset$. If a disk B_i does not contain any point from the lattice, there are two possibilities:

- B_i is contained in a cell C of $\delta\Lambda$; see Figure 1 left. In this case, place $p_i := c_i$ in the center of B_i, and remove the vertices of the cell C from the set of possible placements for the other disks, that is, add them to Q.

- B_i intersects an edge E of $\delta\Lambda$; see Figure 1 right. In this case, choose p_i on $E \cap B_i$, and remove the vertices of the edge E from the set of possible placements for the other disks, that is, add them to Q.

We are left with disks, say $B_1, \ldots, B_k$, that have some lattice points inside. Consider for each such disk B_i the set of points $P_i := B_i \cap (\delta\Lambda \setminus Q)$ as candidates for the placement corresponding to B_i. Observe that P_i may be empty if $(B_i \cap \delta\Lambda) \subset Q$. We want to make sure that each disk B_i gets a point from P_i, and that each point gets assigned to at most one disk B_i. We deal with this by constructing a bipartite graph G_δ with $B := \{B_1, \ldots, B_k\}$ as one class of nodes and $P := P_1 \cup \cdots \cup P_k$ as the other class, and with an edge between $B_i \in B$ and $p \in P$ whenever $p \in P_i$.

It is clear that a (perfect) matching in G_δ provides a feasible placement. When a matching is not possible, the algorithm reports a feasible placement by placing each point in the center of its disk. We call this algorithm Placement, and its pseudocode is given in Algorithm 1. See Figure 2 for an example.

In any case, Placement always gives a feasible placement $p_1, \ldots, p_n$, and we can then compute the value $D(p_1, \ldots, p_n)$ by finding a closest pair in the placement. We will show that, if $2\delta \leq \delta^*$, a matching exists in G_δ and moreover Placement(δ) gives a placement whose closest pair is at distance at least δ. In particular, this implies that if $B_i \cap \delta\Lambda \neq \emptyset$ but $P_i = B_i \cap (\delta\Lambda \setminus Q) = \emptyset$, then there is no matching in G_δ because the node B_i has no edges, and so we can conclude that $2\delta > \delta^*$. We first make the following definitions.

Definition 1 *In the L_∞ metric, we say that* Placement(δ) succeeds *if the computed placement $p_1, \ldots, p_n$ satisfies $D(p_1, \ldots, p_n) \geq \delta$. Otherwise,* Placement($\delta$) fails.

Algorithm 1 PLACEMENT(δ)

$Q := \emptyset$ {$Q \equiv$ Lattice points that cannot be used further}
for all B_i s.t. $B_i \cap \delta\Lambda = \emptyset$ **do**
  if $B_i \cap E \neq \emptyset$ for some edge E of $\delta\Lambda$ **then**
    choose p_i on $B_i \cap E$;
    add the vertices of E to Q
  else {B_i is fully contained in a cell C of $\delta\Lambda$}
    $p_i := c_i$;
    add the vertices of C to Q;
$P := \emptyset$;
for all B_i s.t. $B_i \cap \delta\Lambda \neq \emptyset$ **do**
  $P_i := B_i \cap (\delta\Lambda \setminus Q)$;
  $P := P \cup P_i$;
construct $G_\delta := (\{B_i \mid B_i \cap \delta\Lambda \neq \emptyset\} \cup P, \{(B_i, p) | p \in P_i\})$;
if G_δ has a (perfect) matching **then**
  for each disk B_i, let p_i be the point that it is matched to;
else
  for each disk B_i, let $p_i := c_i$;

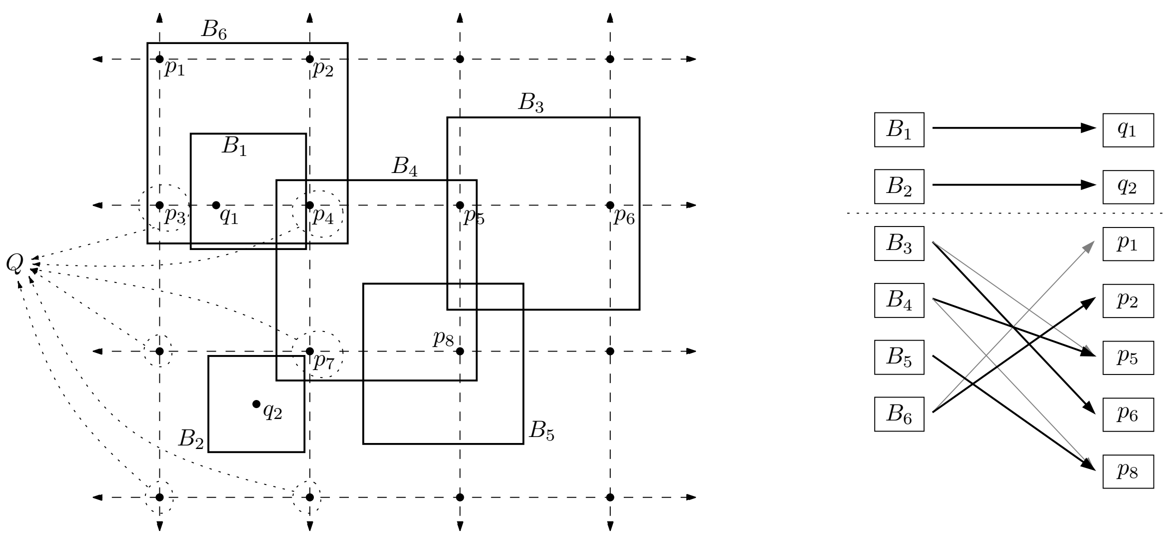

Figure 2: Example showing the main features of the placement algorithm in L_∞.

Lemma 2 *If $2\delta \leq \delta^*$, then* PLACEMENT(δ) *succeeds.*

Proof: The proof is divided in two steps. Firstly, we will show that if $2\delta \leq \delta^*$ then the graph G_δ has a matching. Secondly, we will see that if $p_1, \ldots, p_n$ is a placement computed by PLACEMENT(δ) when $2\delta \leq \delta^*$, then indeed $D(p_1, \ldots, p_n) \geq \delta$.

Consider an optimal placement $p_1^*, \ldots, p_n^*$. The points that we added to Q due to a disk B_i are in the interior of $B(p_i^*, \delta^*/2)$ because of the following analysis:

- If $B_i \cap \delta\Lambda = \emptyset$ and B_i is completely contained in a cell C of $\delta\Lambda$, then p_i^* is in C, and $C \subset B(p_i^*, \delta) \subset B(p_i^*, \delta^*/2)$; see Figure 1 left.

- If $B_i \cap \delta\Lambda = \emptyset$ and there is an edge E of $\delta\Lambda$ such that $B_i \cap E \neq \emptyset$, then $E \subset B(p_i^*, \delta) \subset B(p_i^*, \delta^*/2)$; see Figure 1 right.

The interiors of the disks (in L_∞) $B(p_i^*, \delta^*/2)$ are disjoint, and we can use them to construct a matching in G_δ as follows. If $B_i \cap \delta\Lambda \neq \emptyset$, then $B(p_i^*, \delta^*/2) \cap B_i$ contains some lattice point $p_i \in \delta\Lambda$. Because the interiors of the disks $B(p_i^*, \delta^*/2)$ are disjoint, we have $p_i \notin Q$ and $p_i \in P_i$. We cannot directly add the edge (B_i, p_i) to the matching that we are constructing because it may happen that p_i is on the boundary of $B(p_i^*, \delta^*/2) \cap B_i$, but also on the boundary of $B(p_j^*, \delta^*/2) \cap B_j$. However, in this case, $B(p_i^*, \delta^*/2) \cap B_i \cap \delta\Lambda$ contains an edge of $\delta\Lambda$ inside. If we match each B_i to the lexicographically smallest point in $B(p_i^*, \delta^*/2) \cap B_i \cap \delta\Lambda$, then, because the interiors of disks $B(p_i^*, \delta^*/2)$ are disjoint, each point is claimed by at most one disk. This proves the existence of a matching in G_δ provided that $2\delta \leq \delta^*$.

For the second part of the proof, let p_i, p_j be a pair of points computed by Placement(δ). We want to show that p_i, p_j are at distance at least δ. If both were computed by the matching in G_δ, they both are different points in $\delta\Lambda$, and so they have distance at least δ. If p_i was not placed on a point of $\delta\Lambda$ (at c_i or on an edge of $\delta\Lambda$), then the lattice points closer than δ to p_i were included in Q, and so the distance to any p_j placed during the matching of G_δ is at least δ. If both p_i, p_j were not placed on a point of $\delta\Lambda$, then B_i, B_j do not contain any point from $\delta\Lambda$, and therefore $r_i, r_j < \delta/2$. Two cases arise:

- If both B_i, B_j do not intersect an edge of $\delta\Lambda$, by the triangle inequality we have $d(p_i, p_j) \geq d(p_i^*, p_j^*) - d(p_i, p_i^*) - d(p_j, p_j^*) > \delta^* - \delta/2 - \delta/2 \geq \delta$, provided that $2\delta \leq \delta^*$.
- If one of the disks, say B_i, intersects an edge E of $\delta\Lambda$, then B_i is contained in the two cells of $\delta\Lambda$ that have E as an edge. Let C be the six cells of $\delta\Lambda$ that share a vertex with E. If B_j does not intersect any edge of $\delta\Lambda$, then $B_j \cap C = \emptyset$ because otherwise $d(p_i^*, p_j^*) < 2\delta$, and so $d(p_i, p_j) \geq \delta$. If B_j intersects an edge E' of $\delta\Lambda$, we have $E \cap E' = \emptyset$ because otherwise $d(p_i^*, p_j^*) < 2\delta$. It follows that $d(p_i, p_j) \geq \delta$.

$\circledcirc$

Notice that, in particular, if r_{min} is the radius of the smallest disk and we set $\delta = (r_{min}/\sqrt{n})$, then the nodes of type B_i in G_δ have degree n, and there is always a matching. This implies that $\delta^* = \Omega(r_{min}/\sqrt{n})$.

Observe also that whether Placement fails or succeeds is not a monotone property. That is, there may be values $\delta_1 < \delta_2 < \delta_3$ such that both Placement(δ_1) and Placement(δ_3) succeed, but Placement(δ_2) fails. This happens because for values $\delta \in (\frac{\delta^*}{2}, \delta^*]$, we do not have any guarantee on what Placement(δ) does.

The following observations will be used later.

Observation 3 *If* Placement(δ) *succeeds for $\mathcal{B}$, but* Placement(δ) *fails for a translation of $\mathcal{B}$, then $\delta \leq \delta^* < 2\delta$ and we have a 2-approximation.*

Observation 4 *If for some $\delta > \delta'$,* Placement(δ) *succeeds, but* Placement(δ') *fails, then $\delta^* < 2\delta' < 2\delta$ and we have a 2-approximation.*

The algorithm can be adapted to compute Placement($\delta + \epsilon$) for an infinitesimal $\epsilon > 0$ because only the points of $\delta\Lambda$ lying on the boundaries of $B_1, \dots, B_n$ are affected. More precisely, if a point $\delta p \in \delta\Lambda$ is in the interior of B_i, then, for a sufficiently small $\epsilon > 0$, the point $(\delta+\epsilon)p \in (\delta+\epsilon)\Lambda$ is in B_i as well. On the other hand, if a point $\delta p \in \delta\Lambda$ is on the boundary of B_i, then, for a sufficiently small $\epsilon > 0$, the point $(\delta+\epsilon)p \in (\delta+\epsilon)\Lambda$ is outside B_i if and only if δp is the point of the segment $l_p \cap B_i$ furthest from the origin, where l_p is the line passing through the origin and p. Similar arguments apply for deciding if a disk B_i is contained in a cell of $(\delta+\epsilon)\Lambda$ or it intersects some of its edges. Therefore, for an infinitesimal $\epsilon > 0$, we can decide if PLACEMENT$(\delta+\epsilon)$ succeeds or fails. This leads to the following observation.

Observation 5 *If* PLACEMENT(δ) *succeeds, but* PLACEMENT$(\delta+\epsilon)$ *fails for an infinitesimal* $\epsilon > 0$*, then* $\delta^* \leq 2\delta$ *and we have a 2-approximation.*

2.2 Efficiency of the algorithm

The algorithm PLACEMENT, as stated so far, is not strongly polynomial because the sets $P_i = B_i \cap (\delta\Lambda \setminus Q)$ can have arbitrarily many points, depending on the value δ. However, when P_i has more than n points, we can just take any n of them. This is so because a node B_i with degree at least n is never a problem for the matching: if $G_\delta \setminus B_i$ does not have a matching, then G_δ does not have it either; if $G_\delta \setminus B_i$ has a matching M, then at most $n-1$ nodes from the class P participate in M, and one of the n edges leaving B_i has to go to a node in P that is not in M, and this edge can be added to M to get a matching in G_δ.

For a disk B_i we can decide in constant time if it contains some point from the lattice $\delta\Lambda$: we round its center c_i to the closest point p of the lattice, and depending on whether p belongs to B_i or not, we decide. Each disk B_i adds at most 4 points to Q, and so $|Q| \leq 4n$. We can construct Q and remove repetitions in $O(n \log n)$ time.

If a disk B_i has radius bigger than $3\delta\sqrt{n}$, then it contains more than $5n$ lattice points, that is, $|B_i \cap \delta\Lambda| > 5n$. Because Q contains at most $4n$ points, P_i has more than n points. Therefore, we can shrink the disks with radius bigger than $3\delta\sqrt{n}$ to disks of radius exactly $3\delta\sqrt{n}$, and this does not affect to the construction of the matching. We can then assume that each disk $B_i \in \mathcal{B}$ has radius $O(\delta\sqrt{n})$. In this case, each B_i contains at most $O(n)$ points of $\delta\Lambda$, and so the set $P = \bigcup_i P_i$ has $O(n^2)$ elements.

In fact, we only need to consider a set P with $O(n\sqrt{n})$ points. The idea is to divide the disks $\mathcal{B}$ into two groups: the disks that intersect more than $\sqrt{n}$ other disks, and the ones that intersect less than $\sqrt{n}$ other disks. For the former group, we can see that they bring $O(n\sqrt{n})$ points in total to P. As for the latter group, we only need to consider $O(\sqrt{n})$ points per disk.

Lemma 6 *It is sufficient to consider a set* P *with* $O(n\sqrt{n})$ *points. Moreover, we can construct such a set* P *in* $O(n\sqrt{n}\log n)$ *time.*

Proof: As mentioned above, we can assume that all disks in $\mathcal{B}$ have radius $O(\delta\sqrt{n})$. Among those disks, let $\mathcal{B}_<$ be the set of disks that intersect less than $\sqrt{n}$ other disks in $\mathcal{B}$, and let $\mathcal{B}_>$ be the set of disks that intersect at least $\sqrt{n}$ other disks in $\mathcal{B}$. We treat $\mathcal{B}_<$ and $\mathcal{B}_>$ independently. We first show that for the disks in $\mathcal{B}_<$ we only need to consider $O(n\sqrt{n})$ points, and then we show that the disks in $\mathcal{B}_>$ add at most $O(n\sqrt{n})$ points to P.

For each disk $B_i \in \mathcal{B}_<$, it is enough if P_i consists of $\sqrt{n}$ points. This is so because then the node B_i is never a problem for the matching in G_δ. If $G_\delta \setminus B_i$ does not have a matching, then G_δ does not have it either. If $G_\delta \setminus B_i$ has a matching M, then at most $\sqrt{n}-1$ nodes of P_i participate in M because only the disks that intersect B_i can use a point in P_i, and there are at most $\sqrt{n}-1$ by definition of $\mathcal{B}_<$. Therefore, one of the $\sqrt{n}$ edges leaving B_i has to go to a node in P_i that is not in M, and this edge can be added to M to get a matching in G_δ.

We can construct the sets P_i for all the disks in $\mathcal{B}_<$ in $O(n\sqrt{n}\log n)$ time. First, construct Q and preprocess it to decide in $O(\log n)$ time if a query point is in Q or not. This takes $O(n \log n)$ time. For each disk $B_i \in \mathcal{B}_<$, pick points in $B_i \cap (\delta\Lambda \setminus Q)$ as follows. Initialize $P_i = \emptyset$. Take a point $p \in B_i \cap \delta\Lambda$ and check in $O(\log n)$ time if $p \in Q$. If $p \in Q$, then take another point p and repeat the test. If $p \notin Q$, then add p to P_i. Stop when P_i has $\sqrt{n}$ points or there are no points left in $B_i \cap \delta\Lambda$.

For a disk B_i we may spend $\Omega(n)$ time if, for example, $Q \subset (B_i \cap \delta\Lambda)$. However, each point in Q has appeared in the construction of at most $\sqrt{n}$ different sets P_i, as otherwise there is a point $q \in Q$ that intersects $\sqrt{n}$ disks in $\mathcal{B}_<$, which is impossible. Therefore, we have spent $O(n\sqrt{n}\log n)$ time overall.

As for the disks in $\mathcal{B}_>$, let $U = \bigcup_{B_i \in \mathcal{B}_<} B_i$ be the region that they cover. We will see how to compute $U \cap \delta\Lambda$ in $O(n\sqrt{n}\log n)$ time, and this will finish the proof. Consider the disk $B_i \in \mathcal{B}$ with biggest radius, say r, and grow each disk in $\mathcal{B}$ to have radius r. We keep calling them $\mathcal{B}$. Construct a subset $\tilde{\mathcal{B}}_> \subset \mathcal{B}_>$ as follows. Initially set $\tilde{\mathcal{B}}_> = \emptyset$, and for each $B_i \in \mathcal{B}_>$, add B_i to $\tilde{\mathcal{B}}_>$ if and only if B_i does not intersect any disk in the current $\tilde{\mathcal{B}}_>$.

Consider the number I of intersections between elements of $\tilde{\mathcal{B}}_>$ and $\mathcal{B}$. On the one hand, each disk in $\tilde{\mathcal{B}}_>$ intersects at least $\sqrt{n}$ elements of $\mathcal{B}$ by definition of $\mathcal{B}_>$, so we have $|\tilde{\mathcal{B}}_>|\sqrt{n} \le I$. On the other hand, because the disks in $\tilde{\mathcal{B}}_>$ are disjoint by construction and all have the same size after the growing, each disk of $\mathcal{B}$ can intersect at most four other disks of $\tilde{\mathcal{B}}_>$, and we get $I \le 4n$. We conclude that $|\tilde{\mathcal{B}}_>| \le O(\sqrt{n})$.

Each disk in $\mathcal{B}_>$ intersects some disk in $\tilde{\mathcal{B}}_>$. Therefore, because r is the radius of the largest disk in $\mathcal{B}_>$, we can cover the whole region U by putting disks of radius $3r$ centered at the disks of $\tilde{\mathcal{B}}_>$. Formally, we have that $U \subset \bigcup_{B_i \in \tilde{\mathcal{B}}_>} B(c_i, 3r) =: \tilde{U}$. There are $O(\sqrt{n})$ such disks, and each of them contains $O(n)$ points of $\delta\Lambda$ because $3r = O(\delta\sqrt{n})$. We can then compute all the lattice points $\tilde{P} = \tilde{U} \cap \delta\Lambda$ in this region and remove repetitions in $O(n\sqrt{n}\log n)$ time. In particular, we have that $|U \cap \delta\Lambda| \le |\tilde{P}| = O(n\sqrt{n})$.

To report $U \cap \delta\Lambda$, we first compute U and decide for each point in $\tilde{P}$ if it belongs to U or not. Because the disks behave like pseudo-disks, U has linear size description, and we can compute it in near-linear time [16]. We can then process U to decide in $O(\log n)$ time if it contains a query point or not. We query with the $O(n\sqrt{n})$ points in $\tilde{P}$, and add to P those that are contained in U. This accomplishes the computation of $U \cap \delta\Lambda$ in $O(n\sqrt{n}\log n)$ time.
☺

We are left with the following problem: given a set P of $O(n\sqrt{n})$ points, and a set $\mathcal{B}$ of n disks, find a maximum matching between P and $\mathcal{B}$ such that a point is matched to a disk that contains it. We also know that each B_i contains at most $O(n)$ points of P.

If we forget about the geometry of the problem, we have a bipartite graph G_δ whose smallest class has n vertices and $O(n^2)$ edges. We can construct G_δ explicitly in $O(n^2)$ time, and then compute a maximum matching in $O(\sqrt{n}n^2) = O(n^{2.5})$ time [15]; see also [21]. In fact, to achieve this running time it would be easier to forget Lemma 6, and construct each set P_i by choosing $5n$ points per disk B_i, and then removing Q from them.

However, the graph G_δ does not need to be constructed explicitly because its edges are implicitly represented by the the disk-point containment. This type of matching problem, when both sets have the same cardinality, has been considered by Efrat et al. [9, 10]. Although in our setting one of the sets may be much larger than the other one, we can make minor modifications to the algorithm in [9] and use Mortensen's data structure [19] to get the following result.

Lemma 7 *In the L_∞ metric,* Placement *can be adapted to run in $O(n\sqrt{n}\log n)$ time.*

Proof: We compute the set P of Lemma 6 in $O(n\sqrt{n}\log n)$ time, and then apply the idea by Efrat et al. [9] to compute the matching; see also [10]. The maximum matching has cardinality at most n, and then the Dinitz's matching algorithm finishes in $O(\sqrt{n})$ phases [15]; see also [21].

In each of the phases, we need a data structure for the points P that supports point deletions and witness queries with squares (disks in L_∞). If we construct the data structure anew in each phase, and P has $\Omega(n\sqrt{n})$ points, then we would need $\Omega(n\sqrt{n})$ time per phase, which is too much. Instead, we construct the data structure $\mathcal{D}(P)$ of [19] only once, and reuse it for all phases. The data structure $\mathcal{D}(P)$ can be constructed in $O(n\sqrt{n}\log n)$ time, and it supports insertions and deletions in $O(\log n)$ time per operation. Moreover, $\mathcal{D}(P)$ can be modified for answering witness queries in $O(\log n)$ time [20]: for a query rectangle R, it reports a witness point in $R \cap P$, or the empty set.

We show how a phase of the algorithm can be implemented in $O(n\log n)$ time. Consider the construction of the layered graph $\mathcal{L}$, as in Section 3 of [9]; $\mathcal{B}$ for the odd layers, and P for the even layers. We make the following modifications:

- We construct the whole layered graph $\mathcal{L}$ but without the last layer. Call it $\mathcal{L}'$. The reason is that the graph $\mathcal{L}'$ only has $O(n)$ vertices. All odd layers together have at most n vertices; an odd layer is a subset of $\mathcal{B}$, and each $B_i \in \mathcal{B}$ appears in at most one layer. In all the even layers together except for the last, the number of vertices is bounded by the matching, and so it has $O(n)$ vertices (points).

  The last layer may have a superlinear number of vertices (points), but we can avoid its complete construction: if we are constructing a layer L_{2j} and we detect that it contains more than n vertices, then L_{2j} necessarily has an exposed vertex, that is, a vertex that is not used in the current matching. In this case we just put back into $\mathcal{D}$ all the vertices of L_{2j} that we already computed.

  For constructing $\mathcal{L}'$ we need to query $O(n)$ times the data structure $\mathcal{D}$, and make $O(n)$ deletions. This takes $O(n\log n)$ time. If $P' \subset P$ is the subset of points that are in $\mathcal{L}'$, the final status of $\mathcal{D}$ is equivalent, in time bounds, to $\mathcal{D}(P \setminus P')$.

- For computing the augmenting paths, we use the reduced version $\mathcal{L}'$ that we have computed, together with the data structure $\mathcal{D}(P \setminus P')$. All the layers but the last can be accessed using $\mathcal{L}'$; when we need information of the last layer, we can get the relevant information by querying $\mathcal{D}(P \setminus P')$ for a witness and delete the witness element from it. We need at most one such query per augmenting path, and so we make at most n witness queries and deletions in $\mathcal{D}$. The required time is $O(n\log n)$.

- Instead of constructing the data structure $\mathcal{D}(P)$ anew at the beginning of each phase, we reconstruct it at the end of each phase. Observe that we have deleted $O(n)$ points from $\mathcal{D}(P)$. We can insert all of them back in $O(n\log n)$ time because the data structure is fully-dynamic. In fact, because the points that we are inserting back are exactly all the points that were deleted, a data structure supporting only deletions could also do the job: for each deletion we keep track of the operations that have been done and now we do them backwards.

We have $O(\sqrt{n})$ phases, and each phase takes $O(n \log n)$ time. Therefore, we only need $O(n\sqrt{n} \log n)$ time for all the phases after P and $\mathcal{D}(P)$ are constructed.

Computing the closest pair in a set of n points can be done in $O(n \log n)$ time, and so the time to decide if Placement(δ) succeeds or fails is dominated by the time needed to compute Placement(δ).

3 Approximation algorithms for L_∞

When we have a lower and an upper bound on the optimum value $\delta^* = D(\mathcal{B})$, we can use Lemma 7 to perform a binary search on a value δ such that Placement(δ) succeeds, but Placement($\delta + \epsilon$) fails, where $\epsilon > 0$ is any constant fixed a priori. Due to Lemma 2, this means that $\delta \leq \delta^* < 2(\delta + \epsilon)$ and so we can get arbitrarily close, in absolute error, to a 2-approximation of δ^*.

We can also apply parametric search [17] to find a value $\tilde{\delta}$ such that Placement($\tilde{\delta}$) succeeds, but Placement($\tilde{\delta} + \epsilon$) fails for an infinitesimally small $\epsilon > 0$. Such a value $\tilde{\delta}$ can be computed in $O(n^3 \log^2 n)$ time, and it is a 2-approximation because of Observation 5. Megiddo's ideas [18] of using a parallel algorithms to speed up parametric search are not very fruitful in this case because the known algorithms for computing maximum matchings [14] in parallel machines do not have an appropriate tradeoff between the number of processors and the running time.

Instead, we will use the geometric characteristics of our problem to find a 2-approximation $\tilde{\delta}$ in $O(n\sqrt{n} \log^2 n)$ time. The idea is to consider for which values δ the algorithm changes its behavior, and use it to narrow down the interval where $\tilde{\delta}$ can lie. More specifically, we will use the following facts in a top-bottom fashion:

- For a given δ, only the disks B_i with radius at most $3\delta\sqrt{n}$ are relevant. Therefore, the algorithm constructs non-isomorphic graphs G_δ if δ is below or above $\frac{r_i}{3\sqrt{n}}$.

- The disks B_i with radius $r_i < \frac{\delta^*}{4}$ are disjoint.

- If all the disks in $\mathcal{B}$ are disjoint, placing each point in the center of its disk gives a 2-approximation.

- For a value δ, assume that the disks $\mathcal{B}$ can be partitioned into two sets $\mathcal{B}_1, \mathcal{B}_2$ such that the distance between any disk in $\mathcal{B}_1$ and any disk in $\mathcal{B}_2$ is bigger than δ. If $2\delta \leq \delta^*$, then we can compute a successful placement by putting together Placement(δ) for $\mathcal{B}_1$ and and Placement(δ) for $\mathcal{B}_2$.

- If for a given δ and $\mathcal{B}$ we cannot apply the division of the previous item, and each disk $B_i \in \mathcal{B}$ has radius at most R, then $\mathcal{B}$ can be enclosed in a disk B of radius $O(|\mathcal{B}|R)$.

We show how to solve this last type of problems, and then we use it to prove our main result.

Lemma 8 *Let $\mathcal{B}$ be an instance consisting of m disks such that each disk $B_i \in \mathcal{B}$ has radius $O(r\sqrt{k})$, and assume that there is a disk B of radius $R = O(mr\sqrt{k})$ enclosing all the disks in $\mathcal{B}$. If* Placement$(\frac{r}{3\sqrt{k}})$ *succeeds, then we can compute in $O(m\sqrt{m}\log^2 mk)$ time a placement $p_1, \ldots, p_m$ with $p_i \in B_i$ that yields a 2-approximation of $D(\mathcal{B})$.*

Proof: The proof is divided into three parts. Firstly, we show that we can assume that the origin is placed at the center of the enclosing disk B. Secondly, we narrow down our search space to an interval $[\delta_1, \delta_2]$ such that Placement(δ_1) succeeds but Placement(δ_2) fails. Moreover, for any $\delta \in (\delta_1, \delta_2]$, the subset of lattice points $\tilde{P} \subset \Lambda$ such that $\delta\tilde{P}$ are inside the enclosing ball B is exactly the same. Finally, we consider all the critical values $\delta \in [\delta_1, \delta_2]$ for which the flow of control of Placement(δ) is different than for Placement$(\delta + \epsilon)$ or Placement$(\delta - \epsilon)$. The important observation is that the values δ_1, δ_2 are such that not many critical values are in the interval $[\delta_1, \delta_2]$.

Let $\mathcal{B}'$ be a translation of $\mathcal{B}$ such that the center of the enclosing disk B is at the origin. By hypothesis, Placement$(\frac{r}{3\sqrt{k}})$ for $\mathcal{B}$ succeeds. If Placement$(\frac{r}{3\sqrt{k}})$ for $\mathcal{B}'$ fails, then Placement$(\frac{r}{3\sqrt{k}})$ for $\mathcal{B}$ gives a 2-approximation due to Observation 3, and we are done. From now on, we assume that Placement$(\frac{r}{3\sqrt{k}})$ succeeds and the center of B is at the origin. This finishes the first part of the proof.

As for the second part, consider the horizontal axis h. Because the enclosing disk B has radius $R = O(mr\sqrt{k})$, the lattice $(\frac{r}{3\sqrt{k}})\Lambda$ has $O(mk)$ points in $B \cap h$. Equivalently, we have $t = \max\{z \in \mathbb{Z} \text{ s.t.} (\frac{r}{3\sqrt{k}})(z, 0) \in B\} = \lfloor \frac{3R\sqrt{k}}{r} \rfloor = O(mk)$. In particular, $\frac{R}{t+1} \leq \frac{r}{3\sqrt{k}}$.

If Placement$(\frac{R}{t+1})$ fails, then Placement$(\frac{r}{3\sqrt{k}})$ is a 2-approximation due to Observation 4. So we can assume that Placement$(\frac{R}{t+1})$ succeeds. We can also assume that Placement$(\frac{R}{1})$ fails, as otherwise $\mathcal{B}$ consists of only one disk.

We perform a binary search in $\mathbb{Z} \cap [1, t+1]$ to find a value $t' \in \mathbb{Z}$ such that Placement$(\frac{R}{t'})$ succeeds but Placement$(\frac{R}{t'-1})$ fails. We can do this with $O(\log t) = O(\log mk)$ calls to Placement, each taking $O(m\sqrt{m}\log m)$ time due to Lemma 7, and we have spent $O(m\sqrt{m}\log^2 mk)$ time in total. Let $\delta_1 := \frac{R}{t'}$ and $\delta_2 := \frac{R}{t'-1}$.

Consider the lattice points $\tilde{P} := \Lambda \bigcap [-(t'-1), t'-1]^2$. For any $\delta \in (\delta_1, \delta_2]$, the points $\delta\tilde{P}$ are in B. The intuition behind why these values δ_1, δ_2 are relevant is the following. If for a point $p \in \Lambda$ we consider δp as a function of δ, then the points p that are further from the origin move quicker. Therefore, the points $\delta_2\tilde{P}$ cannot go very far from $\delta_1\Lambda$ because the extreme cases are the points on ∂B. This finishes the second part of the proof.

Before we start the third part, let us state and prove the property of δ_1, δ_2 that we will use later; see Figure 3. If $p \in \Lambda$ is such that $\delta_1 p$ is in the interior of B, and C_p is the union of all four cells of $\delta_1\Lambda$ having $\delta_1 p$ as a vertex, then $\delta_2 p \in C_p$, and more generally, $\delta p \in C_p$ for any $\delta \in [\delta_1, \delta_2]$. Therefore, if for a point $p \in \Lambda$ there is a $\delta \in [\delta_1, \delta_2]$ such that $\delta p \in \partial B_i$, then ∂B_i must intersect C_p.

To show that indeed this property holds, consider a point $p = (p_x, p_y) \in \Lambda$ such that $\delta_1 p$ is in the interior of B. We then have $|\delta_1 p_x| < R$, and because $|p_x| < \frac{R}{\delta_1} = \frac{R}{R/t'} = t'$ and $p_x \in \mathbb{Z}$, we conclude that $|p_x| \leq t' - 1$. This implies that

$$|\delta_2 p_x - \delta_1 p_x| = \left|\delta_1 p_x (\frac{\delta_2}{\delta_1} - 1)\right| = \left|\delta_1 p_x (\frac{t'}{t'-1} - 1)\right| = \delta_1 \frac{|p_x|}{t'-1} \leq \delta_1.$$

The same arguments shows that

$$|\delta_2 p_y - \delta_1 p_y| \leq \delta_1.$$

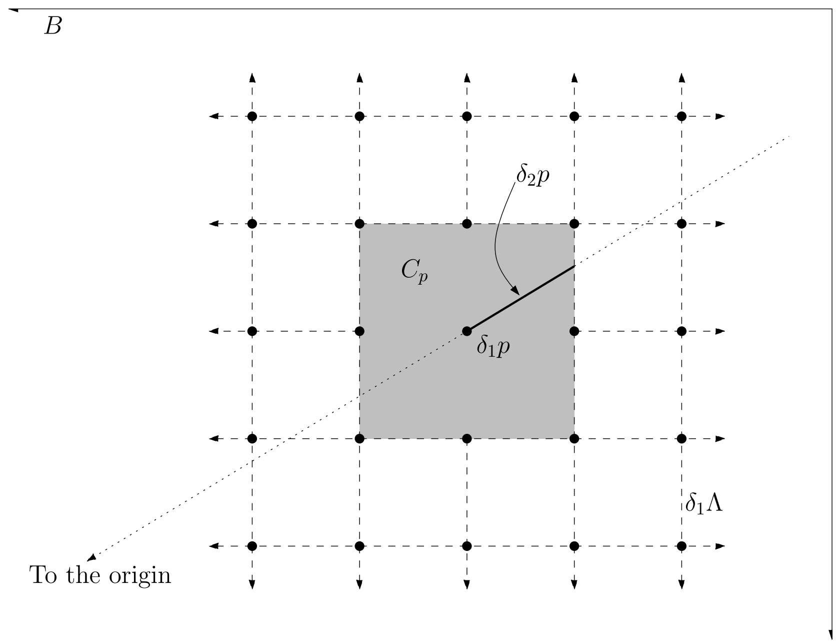

Figure 3: If for $p \in \Lambda$ we have $\delta_1 p \in B$, then $\delta_2 p$ lies in one of the cells of $\delta_1\Lambda$ adjacent to $\delta_1 p$.

Since each coordinate of $\delta_2 p$ differs by at most δ_1 of the coordinates of $\delta_1 p$, we see that indeed $\delta_2 p$ is in the cells C_p of $\delta_1\Lambda$.

We are ready for the third part of the proof. Consider the critical values $\delta \in [\delta_1, \delta_2]$ for which the flow of control of the Placement changes. They are the following:

- A point $p \in \Lambda$ such that $\delta p \in B_i$ but $(\delta + \epsilon)p \notin B_i$ or $(\delta - \epsilon)p \notin B_i$ for an infinitesimal $\epsilon > 0$. That is, $\delta p \in \partial B_i$.
- B_i intersects an edge of $\delta\Lambda$, but not of $(\delta + \epsilon)\Lambda$ $(\delta - \epsilon)\Lambda$ for an infinitesimal $\epsilon > 0$.

Because of the property of δ_1, δ_2 stated above, only the vertices V of cells of $\delta_1\Lambda$ that intersect ∂B_i can change the flow of control of Placement. In the L_∞ metric, because the disks are axis-aligned squares, the vertices V are distributed along two axis-aligned rectangles R_1 and R_2. All the vertices of V along the same side of R_1 or R_2 come in or out of B_i at the same time, that is, they intersect ∂B_i for the same value δ. Therefore, each disk B_i induces $O(1)$ such critical values Δ_i changing the flow of control of Placement, and we can compute them in $O(1)$ time.

We can compute all the critical values $\Delta = \bigcup_{i=1}^{m} \Delta_i$ and sort them in $O(m \log m)$ time. Using a binary search on Δ, we find $\delta_3, \delta_4 \in \Delta$, with $\delta_3 < \delta_4$, such that Placement(δ_3) succeeds but Placement(δ_4) fails. Because $|\Delta| = O(m)$, this can be done in $O(m\sqrt{m}\log^2 m)$ time with $O(\log m)$ calls to Placement. The flow of control of Placement(δ_4) and of Placement($\delta_3 + \epsilon$) are the same. Therefore, Placement($\delta_3 + \epsilon$) also fails, and we conclude that Placement(δ_3) yields a 2-approximation because of Observation 5. ◎

Theorem 9 *Let $\mathcal{B} = \{B_1, \ldots, B_n\}$ be a collection of disks in the plane with the L_∞ metric. We can compute in $O(n\sqrt{n}\log^2 n)$ time a placement $p_1, \ldots, p_n$ with $p_i \in B_i$ that yields a 2-approximation of $D(\mathcal{B})$.*

Proof: Let us assume that $r_1 \leq \cdots \leq r_n$, that is, B_i is smaller than B_{i+1}. Consider the values $\Delta = \{\frac{r_1}{3\sqrt{n}}, \ldots, \frac{r_n}{3\sqrt{n}}, 4r_n\}$. We know that PLACEMENT($\frac{r_1}{3\sqrt{n}}$) succeeds, and we can assume that PLACEMENT($4r_n$) fails; if it would succeed, then the disks in $\mathcal{B}$ would be disjoint, and placing each point $p_i := c_i$ would give a 2-approximation.

We use PLACEMENT to make a binary search on the values Δ and find a value r_{max} such that PLACEMENT($\frac{r_{max}}{3\sqrt{n}}$) succeeds but PLACEMENT($\frac{r_{max+1}}{3\sqrt{n}}$) fails. This takes $O(n\sqrt{n}\log^2 n)$ time, and two cases arise:

- If PLACEMENT($4r_{max}$) succeeds, then $r_{max} \neq r_n$. In the case that $4r_{max} > \frac{r_{max+1}}{3\sqrt{n}}$, we have a 2-approximation due to Observation 4. In the case that $4r_{max} \leq \frac{r_{max+1}}{3\sqrt{n}}$, consider any value $\delta \in [4r_{max}, \frac{r_{max+1}}{3\sqrt{n}}]$. On the one hand, the balls $B_{max+1}, \ldots, B_n$ are not problematic because they have degree n in G_δ. On the other hand, the balls $B_1, \ldots, B_{max}$ have to be disjoint because $\delta^* \geq 4r_{max}$, and they determine the closest pair in PLACEMENT(δ). In this case, placing the points $p_1, \ldots, p_{max}$ at the centers of their corresponding disks, computing the distance $\tilde{\delta}$ of their closest pair, and using PLACEMENT($\tilde{\delta}$) for the disks $B_{max+1}, \ldots, B_n$ provides a 2-approximation.

- If PLACEMENT($4r_{max}$) fails, then we know that for any $\delta \in [\frac{r_{max}}{3\sqrt{n}}, 4r_{max}]$ the disks B_j with $\frac{r_j}{3\sqrt{n}} \geq 4r_{max}$ have degree at least n in G_δ. We shrink them to have radius $12r_{max}\sqrt{n}$, and then they keep having degree at least n in G_δ, so they are not problematic for the matching. We also use $\mathcal{B}$ for the new instance (with shrunk disks), and we can assume that all the disks have radius $O(12r_{max}\sqrt{n}) = O(r_{max}\sqrt{n})$.

  We group the disks $\mathcal{B}$ into clusters $\mathcal{B}_1, \ldots, \mathcal{B}_t$ as follows: a *cluster* is a connected component of the intersection graph of the disks $B(c_1, r_1 + 4r_{max}), \ldots, B(c_n, r_n + 4r_{max})$. This implies that the distance between different clusters is at least $4r_{max}$, and that each cluster $\mathcal{B}_j$ can be enclosed in a disk of radius $O(r_{max}|\mathcal{B}_j|\sqrt{n})$.

  For each subinstance $\mathcal{B}_j$, we use Lemma 8, where $m = |\mathcal{B}_j|$ and $k = n$, and compute in $O(|\mathcal{B}_j|\sqrt{|\mathcal{B}_j|}\log^2(|\mathcal{B}_j|n))$ time a placement yielding a 2-approximation of $D(\mathcal{B}_j)$. Joining all the placements we get a 2-approximation of $D(\mathcal{B})$, and because $n = \sum_{i=1}^{t}|\mathcal{B}_i|$, we have used

$$\sum_{j=1}^{t} O\Big(|\mathcal{B}_j|\sqrt{|\mathcal{B}_j|}\log^2(|\mathcal{B}_j|n)\Big) = O(n\sqrt{n}\log^2 n)$$

  time for this last step.

$\circledS$

4 Approximation algorithms in the L_2 metric

We will now study how the L_2 metric changes the bounds and results of the algorithms studied for the L_∞ metric. First, we consider arbitrary disks. Then, we concentrate on congruent disks, for which we can improve the approximation ratio.

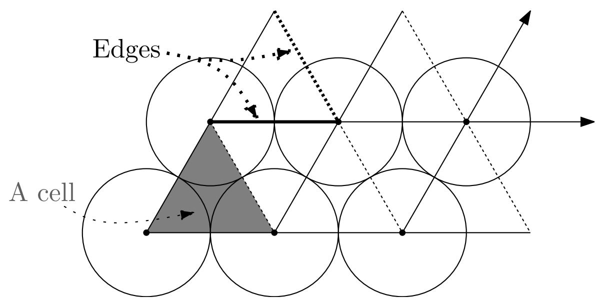

Figure 4: Hexagonal packing induced by the lattice $\delta\Lambda = \{\delta(a + \frac{b}{2}, \frac{b\sqrt{3}}{2}) \,|\, a, b \in \mathbb{Z}\}$. A cell and a couple of edges of $\delta\Lambda$ are also indicated.

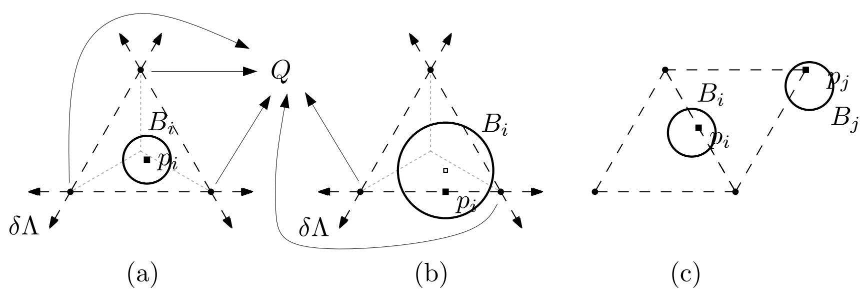

Figure 5: Cases and properties of Placement for the L_2 metric. (a) Placement when B_i is fully contained in a cell. (b) Placement when B_i intersects an edge: we project the center c_i onto the closest edge. (c) A case showing that the closest pair in Placement(δ) may be at distance $\frac{\delta\sqrt{3}}{2}$.

4.1 Arbitrary disks

For the L_∞ metric, we used the optimal packing of disks that is provided by an orthogonal grid. For the Euclidean L_2 metric we will consider the regular hexagonal packing of disks; see Figure 4. For this section, we let

$$\Lambda := \{(a + \frac{b}{2}, \frac{b\sqrt{3}}{2}) \,|\, a, b \in \mathbb{Z}\}.$$

Like in previous sections, we use $\delta\Lambda = \{\delta p \,|\, p \in \Lambda\}$. For disks of radius $\delta/2$, the hexagonal packing is provided by placing the disks centered at $\delta\Lambda$. The *edges* of $\delta\Lambda$ are the segments connecting each pair of points in $\delta\Lambda$ at distance δ. They decompose the plane into equilateral triangles of side length δ, which are the *cells* of $\delta\Lambda$; see Figure 4.

Consider a version of Placement using the new lattice $\delta\Lambda$ and modifying it slightly for the cases when B_i contains no lattice point:

- If B_i is contained in a cell C, place $p_i := c_i$ and add the vertices of C to Q; see Figure 5a.
- If B_i intersects some edges of $\delta\Lambda$, let E be the edge that is closest to c_i. Then, place p_i at the projection of c_i onto E, and add the vertices of E to Q; see Figure 5b.

Observe that, in this case, the distance between a point placed on an edge and a point in

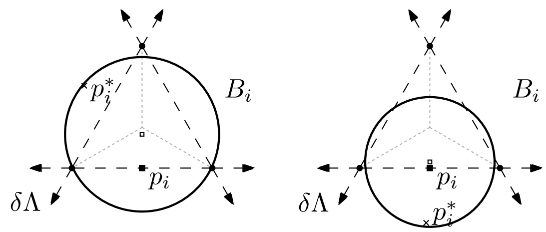

Figure 6: Part of the analysis of Placement for the L_2 metric. (a) and (b) When $B_i \cap \delta\Lambda = \emptyset$ and p_i was placed on the edge E, then the distance from p_i^* to any point of E is at most $\frac{\delta\sqrt{3}}{2}$, and therefore $E \subset B(p_i^*, \frac{\delta\sqrt{3}}{2})$.

$\delta\Lambda \setminus Q$ may be $\frac{\delta\sqrt{3}}{2}$; see Figure 5c. We modify accordingly the criteria of Definition 1 regarding when Placement succeeds, and then we state the result corresponding to Lemma 2.

Definition 10 *In the L_2 metric, we say that* Placement(δ) succeeds *if the computed placement $p_1, \ldots, p_n$ satisfies $D(p_1, \ldots, p_n) \geq \frac{\delta\sqrt{3}}{2}$. Otherwise,* Placement($\delta$) fails.

Lemma 11 *If $\frac{4\delta}{\sqrt{3}} \leq \delta^*$, then* Placement($\delta$) *succeeds.*

Proof: We follow the proof of Lemma 2. Firstly, we argue that if $\frac{4\delta}{\sqrt{3}} \leq \delta^*$, then G_δ has a matching. Secondly, we will see that if $p_1, \ldots, p_n$ is the placement computed by Placement(δ) when $\frac{4\delta}{\sqrt{3}} \leq \delta^*$, then indeed $D(p_1, \ldots, p_n) \geq \frac{\delta\sqrt{3}}{2}$, that is, Placement($\delta$) succeeds.

Consider an optimal feasible placement $p_1^*, \ldots, p_n^*$ achieving δ^*. We then know that the interiors of $B(p_1^*, \delta^*/2), \ldots, B(p_n^*, \delta^*/2)$ are disjoint. To show that G_δ has a matching, we have to argue that:

- If $B_i \cap \delta\Lambda = \emptyset$, then the points that B_i contributed to Q are in the interior of $B(p_i^*, \delta^*/2)$. We consider both cases that may happen. In case that B_i is fully contained in a cell C of $\delta\Lambda$, then $p_i^* \in C$, and so $C \subset B(p_i^*, \delta) \subset B(p_i^*, \frac{2\delta}{\sqrt{3}}) \subset B(p_i^*, \delta^*/2)$, and the vertices of C are in $B(p_i^*, \delta^*/2)$. In case that B_i intersects edges of $\delta\Lambda$ and p_i was placed on E, then E is the closest edge of $\delta\Lambda$ to c_i and $E \subset B(p_i^*, \frac{2\delta}{\sqrt{3}})$, as can be analyzed in the extreme cases depicted in Figures 6a and 6b.

- If $B_i \cap \delta\Lambda \neq \emptyset$, we have to argue that there is point $p \in B_i \cap (\delta\Lambda \setminus Q)$. If B_i has diameter smaller than $\delta^*/2$, then $B_i \subset B(p_i^*, \delta^*/2)$ and the points in $B_i \cap \delta\Lambda$ are inside $B(p_i^*, \delta^*/2)$, and so not in Q. If B_i has diameter bigger than $\delta^*/2$, then the region $B_i \cap B(p_i^*, \delta^*/2)$ contains a disk B' of diameter at least $\frac{\delta^*}{2} \geq \frac{2\delta}{\sqrt{3}}$. It is not difficult to see that then B' contains a point $p \in \delta\Lambda$ (actually this also follows from Lemma 16), and so there is a point $p \in \delta\Lambda \cap B' \subset (B(p_i^*, \delta^*/2) \cap \delta\Lambda)$ which cannot be in Q.

This finishes the first part of the proof. For the second part, consider a pair of points p_i, p_j of the placement computed by Placement(δ) when $\frac{4\delta}{\sqrt{3}} \leq \delta^*$. If they have been assigned in the matching of G_δ, then they are distinct points of $\delta\Lambda$ and so they are at distance at least

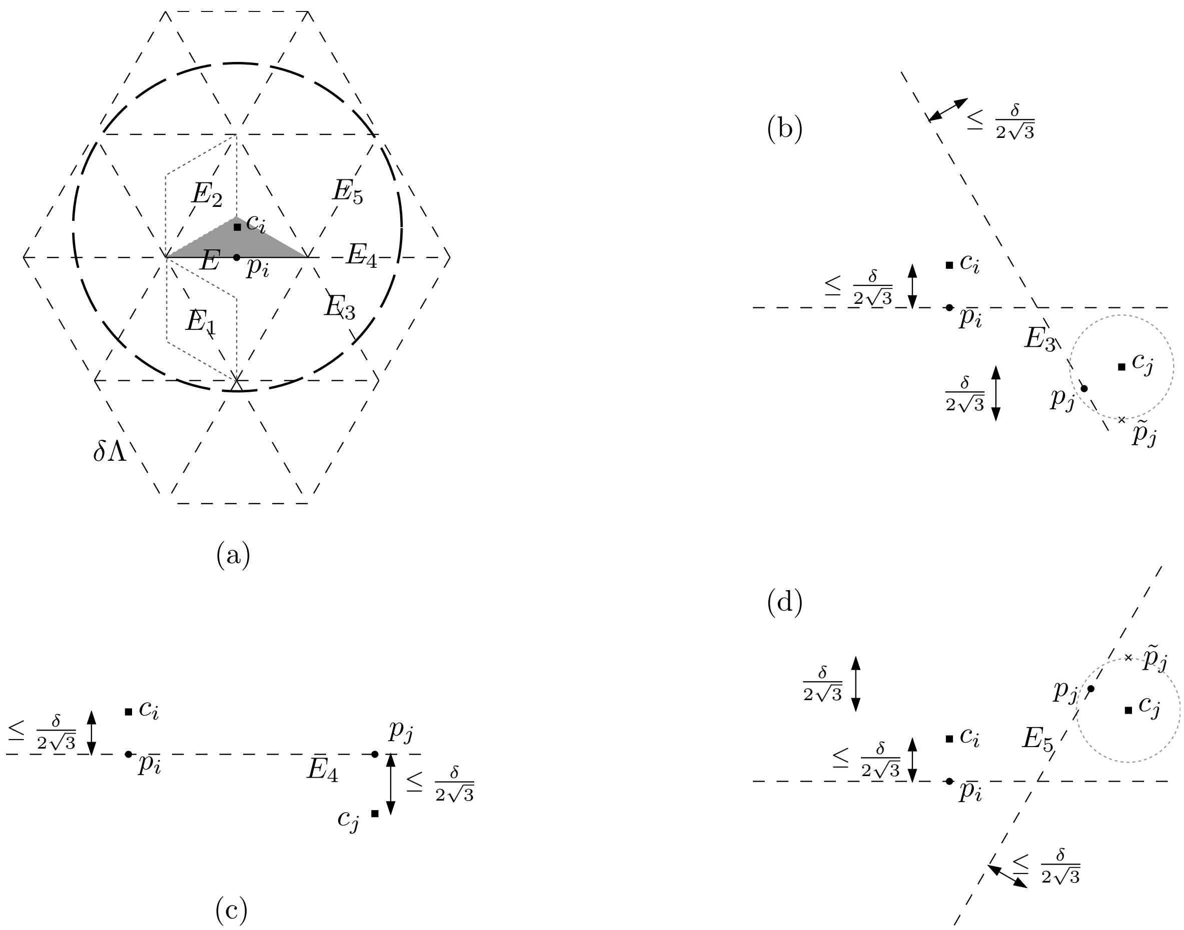

Figure 7: Analysis to show that $d(p_i, p_j) \geq \frac{\delta\sqrt{3}}{2}$ when B_i, B_j do not contain any point from $\delta\Lambda$.

δ. If p_j was assigned in the matching and B_i contains no point from $\delta\Lambda$, then the points in $\delta\Lambda_d \setminus Q$ are at distance at least $\frac{\delta\sqrt{3}}{2}$ from p_i, and so are the distance between p_i and p_j.

If both B_i, B_j do not contain any lattice point, then we know that $r_i, r_j < \frac{\delta}{\sqrt{3}}$, $d(p_i, c_i) \leq \frac{\delta}{2\sqrt{3}}$, and $d(c_i, c_j) \geq d(p_i^*, p_j^*) - d(p_i^*, c_i) - d(p_j^*, c_j) > \frac{4\delta}{\sqrt{3}} - 2\frac{\delta}{\sqrt{3}} = \frac{2\delta}{\sqrt{3}}$. We have the following cases:

- B_i and B_j do not intersect any edge of $\delta\Lambda$. Then $d(p_i, p_j) = d(c_i, c_j) > \frac{2\delta}{\sqrt{3}} > \frac{\delta\sqrt{3}}{2}$.
- B_i intersects an edge E of $\delta\Lambda$, but B_j does not. Then $d(p_i, p_j) \geq d(c_i, c_j) - d(p_i, c_i) - d(p_j, c_j) > \frac{2\delta}{\sqrt{3}} - \frac{\delta}{2\sqrt{3}} - 0 = \frac{\delta\sqrt{3}}{2}$.
- Both B_i, B_j intersect edges of $\delta\Lambda$. See Figure 7a to follow the analysis. Without loss of generality, let's assume that c_i lies in the shaded triangle, and so $p_i \in E$, where E is the closest edge to c_i. The problematic cases are when p_j is placed at the edges E_1, E_2, E_3, E_4, E_5, as the other edges are either symmetric to one of these, or further than $\frac{\delta\sqrt{3}}{2}$ from $p_i \in E$. We then have the following subcases:

E_1, E_2. Consider the possible positions of c_j that would induce $p_j \in E_1, E_2$; see Figure 7a.

The center c_j needs to lie in one of the dotted triangles that are adjacent to E_1 and E_2. But the distance between any point of the dotted triangles and any point of the grey triangle is at most $\frac{2\delta}{\sqrt{3}}$, and so in this case we would have $d(c_i, c_j) \leq \frac{2\delta}{\sqrt{3}}$, which is not possible.

E_3. Consider the possible positions of c_j that would induce $p_j \in E_3$; see Figure 7b. For a fixed value $d(c_i, c_j)$, the distance between p_i and p_j is minimized when c_i and c_j are on the same side of the line through p_i and p_j, like in the figure. Consider the point $\tilde{p}_j$ vertically below c_j and at distance $\frac{\delta}{2\sqrt{3}}$ from c_j. Then, we have that $d(p_i, \tilde{p}_j) \geq d(c_i, c_j) > \frac{2\delta}{\sqrt{3}}$. Because $d(p_j, \tilde{p}_j) \leq \frac{\delta}{2\sqrt{3}}$, we get $d(p_i, p_j) \geq d(p_i, \tilde{p}_j) - d(p_j, \tilde{p}_j) > \frac{2\delta}{\sqrt{3}} - \frac{\delta}{2\sqrt{3}} = \frac{\delta\sqrt{3}}{2}$.

E_4. Consider the possible positions of c_j that would induce $p_j \in E_4$; see Figure 7c. For a fixed value $d(c_i, c_j)$, the distance between p_i and p_j is minimized when c_i and c_j are on opposite sides of the line through p_i and p_j, and $d(p_i, c_i) = d(p_j, c_j) = \frac{\delta}{2\sqrt{3}}$. But, in this case, we can use Pythagoras' theorem to get $d(p_i, p_j) = \sqrt{d(c_i, c_j)^2 - \big(d(p_i, c_i) + d(p_j, c_j)\big)^2} > \sqrt{\left(\frac{2\delta}{\sqrt{3}}\right)^2 - \left(\frac{2\delta}{2\sqrt{3}}\right)^2} = \delta$.

E_5. Consider the possible positions of c_j that would induce $p_j \in E_5$; see Figure 7d. For a fixed value $d(c_i, c_j)$, The distance between p_i and p_j is minimized when c_i and c_j are on opposite sides of the line through p_i and p_j, like in the figure. Consider the point $\tilde{p}_j$ vertically above c_j and at distance $\frac{\delta}{2\sqrt{3}}$ from c_j. Then, we have that $d(p_i, \tilde{p}_j) \geq d(c_i, c_j) > \frac{2\delta}{\sqrt{3}}$. Because $d(p_j, \tilde{p}_j) \leq \frac{\delta}{2\sqrt{3}}$, we get $d(p_i, p_j) \geq d(p_i, \tilde{p}_j) - d(p_j, \tilde{p}_j) > \frac{2\delta}{\sqrt{3}} - \frac{\delta}{2\sqrt{3}} = \frac{\delta\sqrt{3}}{2}$.

In all cases, we have $d(p_i, p_j) \geq \frac{\delta\sqrt{3}}{2}$ and this finishes the proof of the lemma. ☺

Like before, we have the following observations.

Observation 12 *If* Placement(δ) *succeeds for* $\mathcal{B}$*, but* Placement(δ) *fails for a translation of* $\mathcal{B}$*, then* $\delta^* \leq \frac{4\delta}{\sqrt{3}}$ *holds and* Placement(δ) *gives an* $\frac{8}{3}$*-approximation.*

If for some $\delta > \delta'$, Placement(δ) *succeeds, but* Placement(δ') *fails, then* $\delta^* \leq \frac{4\delta'}{\sqrt{3}} < \frac{4\delta}{\sqrt{3}}$ *and* Placement(δ) *gives an* $\frac{8}{3}$*-approximation.*

If Placement(δ) *succeeds, but* Placement($\delta + \epsilon$) *fails for an infinitesimal* $\epsilon > 0$*, then* $\delta^* \leq \frac{4\delta}{\sqrt{3}}$ *and* Placement(δ) *gives an* $\frac{8}{3}$*-approximation.*

Lemma 6 also applies to the L_2 metric because all the properties of the L_∞ metric that we used in its proof also apply to the L_2 metric.

In the proof of Lemma 7 we used a dynamic data structure $\mathcal{D}$ for point sets that supports witness queries: given a disk B_i, report a point contained in B_i. In the L_2 case, we can handle this using a dynamic data structure $\mathcal{D}'$ for nearest neighbor queries: given a point p, report a closest point to p. When we want a witness for B_i, we query with c_i for a closest neighbor p_{c_i}. If p_{c_i} lies in B_i, then we report it as witness, and otherwise there is no point inside B_i.

Using the data structure $\mathcal{D}'$ by Agarwal and Matoušek [2] for the point set P, we can construct the data structure in $O(|P|^{1+\epsilon})$ time, it answers nearest neighbor queries in $O(\log^3 |P|)$ time, and supports updates in $O(|P|^\epsilon)$ amortized time, where $\epsilon > 0$ is an arbitrarily small

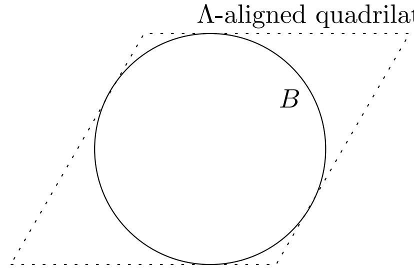

Figure 8: For computing δ_1, δ_2 in the proof of Lemma 14, we use, instead of B, the smallest Λ-aligned quadrilateral that encloses B.

positive value affecting the constants hidden in the O-notation. In the special case that all the disks are *congruent*, it is better to use the data structure developed by Efrat et al. [9]; it uses $O(|P| \log |P|)$ preprocessing time, it answers a witness query and supports a deletion in $O(\log |P|)$ amortized time. Using these data structures and with the proof of Lemma 7, we get the following result for the L_2 metric.

Lemma 13 *The Algorithm* Placement *can be adapted to run in $O(n^{1.5+\epsilon})$ time. When all the disks are congruent, it can be adapted to run in $O(n\sqrt{n}\log n)$ time.*

The running times would actually remain valid for any L_p metric, either using the data structure for nearest neighbors by Agarwal et al. [1] for the general case, or the semi-dynamic data structure of Efrat et al. [9] for congruent disks. However, we would have to use suitable lattices and we would achieve different approximation ratios.

The proof of Lemma 8 is not valid for the L_2 metric because we used the fact that the disks in the L_∞ metric are squares. Instead, we have the following result.

Lemma 14 *Let $\mathcal{B}$ be an instance with m disks such that each disk $B_i \in \mathcal{B}$ has radius $O(r\sqrt{k})$, and that there is a disk B of radius $R = O(mr\sqrt{k})$ enclosing all the disks in $\mathcal{B}$. If* Placement$(\frac{r}{3\sqrt{k}})$ *succeeds, then we can compute a placement $p_1, \ldots, p_m$ with $p_i \in B_i$ that yields an $\frac{8}{3}$-approximation of $D(\mathcal{B})$ in $O(mk)$ time plus $O(\log mk)$ calls to* Placement.

Proof: Consider the proof of Lemma 8. The first part of it is perfectly valid for the L_2 metric as well.

For the second part of the proof, when computing the values δ_1, δ_2, instead of using the enclosing disk B, we use the smallest Λ-aligned quadrilateral that encloses the disk B; see Figure 8. Like in Lemma 8, we compute δ_1, δ_2 by making a binary search on the values $\frac{R}{z}$ with $z \in \mathbb{Z} \cap [1, \frac{3R\sqrt{k}}{r}]$. We do not need to compute them explicitly because they are ordered by the inverse of integer numbers. Because $\frac{3R\sqrt{k}}{r} = O(mk)$, we can do this with $O(\log mk)$ calls to Placement.

Like in Lemma 8, the values δ_1, δ_2 have the property that if for a point $p \in \Lambda$ there is a $\delta \in [\delta_1, \delta_2]$ such that $\delta p \in \partial B_i$, then ∂B_i must intersect C_p. An easy way to see this is to apply a linear transformation that maps $(1, 0)$ to $(1, 0)$ and $(\frac{1}{2}, \frac{\sqrt{3}}{2})$ to $(0, 1)$. Under this transformation, the lattice $\delta\Lambda$ becomes $\delta\mathbb{Z}^2$, the disk becomes an ellipse, the enclosing Λ-aligned quadrilateral becomes a square enclosing the ellipse, and the proof of the equivalent property follows from the discussion in the proof of Lemma 8.

As for the third part of the proof in Lemma 8, where we bound the number of critical values Δ_i that a disk B_i induces, we used that in the L_∞ case the disks are squares. This does not apply to the L_2 disks, but instead we have the following analysis.

Because the perimeter of B_i is $O(r_i) = O(r\sqrt{k})$ and we have $\delta_1 = \Omega(r/\sqrt{k})$, the boundary of B_i intersects $O\big(\frac{r\sqrt{k}}{r/\sqrt{k}}\big) = O(k)$ cells of $\delta_1\Lambda$. Together with the property of δ_1, δ_2 stated above, this means that B_i induces $O(k)$ critical values changing the flow of control of Placement. That is, the set $\Delta_i = \{\delta \in [\delta_1, \delta_2] \mid \exists p \in \Lambda \text{ s.t. } \delta p \in \partial B_i\}$ has $O(k)$ values. Each value in Δ_i can be computed in constant time, and therefore $\Delta = \bigcup_{i=1}^{m} \Delta_i$ can be computed in $O(mk)$ time.

Making a binary search on Δ, we find $\delta_3, \delta_4 \in \Delta$, with $\delta_3 < \delta_4$, such that Placement(δ_3) succeeds but Placement(δ_4) fails. If at each step of the binary search we compute the median M of the elements where we are searching, and then use Placement(M), we find δ_3, δ_4 with $O(\log mk)$ calls to Placement plus $O(mk)$ time for computing all medians because at each step we reduce by half the number of elements where to search.

The flow of control of Placement(δ_4) and of Placement($\delta_3 + \epsilon$) are the same. Therefore, Placement($\delta_3 + \epsilon$) also fails, and we conclude that Placement(δ_3) yields an $\frac{8}{3}$-approximation because of Observation 12. ⊛

Theorem 15 *Let $\mathcal{B} = \{B_1, \ldots, B_n\}$ be a collection of disks in the plane with the L_2 metric. We can compute in $O(n^2)$ time a placement $p_1, \ldots, p_n$ with $p_i \in B_i$ that yields an $\frac{8}{3}$-approximation of $D(\mathcal{B})$.*

Proof: Everything but the time bounds remains valid in the proof of Theorem 9. The proof of Theorem 9 is applicable. For solving the subinstances $\mathcal{B}_j$ we used Lemma 8, and now we need to use Lemma 14. Together with Lemma 13, it means that for solving the subinstance $\mathcal{B}_j$ we have $m = |\mathcal{B}_j|$ and $k = n$, and so we need to use

$$O(|\mathcal{B}_j|n + |\mathcal{B}_j|^{1.5+\epsilon} \log |\mathcal{B}_j|n)$$

time. Summing over all t subinstances, and because $n = \sum_{j=1}^{t} |\mathcal{B}_j|$, we have spent

$$\sum_{j=1}^{t} O\big(|\mathcal{B}_j|n + |\mathcal{B}_j|^{1.5+\epsilon} \log n\big) = O(n^2)$$

time overall. ⊛

4.2 Congruent disks

When the disks $B_1, \ldots, B_n$ are all congruent, say, of diameter one, we can improve the approximation ratio in Theorem 15. For general disks, the problematic cases are those balls that do not contain any lattice point. But when all the disks are congruent, it appears that we can rule out those cases. For studying the performance of Placement with congruent disks, we need the following geometric result.

Lemma 16 *Let B be a disk of diameter one, and let B' be a disk of diameter $1 \leq \delta^* \leq 2$ whose center is in B. Consider $\delta = \frac{-\sqrt{3}+\sqrt{3}\delta^*+\sqrt{3+2\delta^*-\delta^{*2}}}{4}$. Then, the lattice $\delta\Lambda$ has some* *point in $B \cap B'$. Furthermore, this is the biggest value δ having this property. If B' has diameter $\delta^* \leq 1$, then the lattice $(\delta^*/2)\Lambda$ has some point in $B \cap B'$.*

Proof: Firstly, we consider the case $1 \leq \delta^* \leq 2$ and give a construction showing that $\delta = \frac{-\sqrt{3}+\sqrt{3}\delta^*+\sqrt{3+2\delta^*-\delta^{*2}}}{4}$ is indeed the biggest value for which the property holds. Then, we show that $\delta\Lambda$ always has some point in $B \cap B'$ by comparing the different scenarios with the previous construction. Finally, we consider the case $\delta^* \leq 1$.

Assume without loss of generality that the line through the centers of B and B' is vertical. The worst case happens when the center of B' is on the boundary of B. Consider the equilateral triangle T depicted on the left in Figure 9. If the center of B is placed at $(0, 0)$, then the lowest point of T is placed at $(1/2 - \delta^*/2, 0)$, and the line L forming an angle of $\pi/3$ with a horizontal line has equation $L \equiv y = 1/2 - \delta^*/2 + \sqrt{3}x$. The intersection of this line with the boundary of B, defined by $y^2 + x^2 = 1/4$, gives the solutions $x = \pm\frac{-\sqrt{3}+\sqrt{3}\delta^*+\sqrt{3+2\delta^*-\delta^{*2}}}{8}$. Because of symmetry about the vertical line through the centers of B and B', and because the angle between this line and L, the depicted triangle is equilateral and has side length $\frac{-\sqrt{3}+\sqrt{3}\delta^*+\sqrt{3+2\delta^*-\delta^{*2}}}{4}$. This shows that the chosen value of δ is the biggest with the desired property.

We have to show now that the lattice $\delta\Lambda$ has the desired property. It is enough to see that when two vertices of a cell of $\delta\Lambda$ are on the boundary of $B \cap B'$, then the third one is also in $B \cap B'$. In the center of Figure 9 we have the case when the two vertices are on the boundary of B. Consider the edge connecting these two points, and its orthogonal bisector. The bisector passes through the center of B, and its intersection with the boundary of B' contained in B is further from it than the intersection of the boundary of B' with the vertical line through the center. Therefore, the third vertex is inside $B \cap B'$.

In the right of Figure 9 we have the case where the two vertices are on the boundary of B'. If we consider the case when the lowest edge of the cell is horizontal, we can see that the triangle has the third vertex inside. This is because the biggest equilateral triangle with that shape that is inside $B \cap B'$ has side $\delta^*/2$, and this is always bigger than $\delta = \frac{-\sqrt{3}+\sqrt{3}\delta^*+\sqrt{3+2\delta^*-\delta^{*2}}}{4}$ when $1 \leq \delta^* \leq 2$. Then the same argument as in the previous case works.

If one vertex is on the boundary of B and one on the boundary of B', we can rotate the triangle around the first of them until we bring the third vertex on the boundary of B contained in B'. Now we would have two vertices on the boundary of B. If the third vertex was outside $B \cap B'$ before the rotation, then we would have moved the second vertex outside $B \cap B'$, which would contradict the first case. Therefore, the third vertex has to be inside $B \cap B'$.

Regarding the case $\delta^* \leq 1$, we replace the disk B by another disk $\tilde{B}$ of diameter δ^* contained in B and that contains the center of B'. We scale the scenario by $1/\delta^*$ so that both $\tilde{B}$ and B' have diameter 1. If we apply the result we have shown above, we know that $(1/2)\Lambda$ contains some point in $\tilde{B} \cap B'$, and scaling back we get the desired result. ♥

On the one hand, for $1 \leq \delta^* \leq 2$ and $\delta \leq \frac{-\sqrt{3}+\sqrt{3}\delta^*+\sqrt{3+2\delta^*-\delta^{*2}}}{4}$, we have $Q = \emptyset$ when computing Placement(δ), and the graph G_δ has a matching because of Lemma 16 and the proof of Lemma 11. In this case, if $p_1, \ldots, p_n$ is the placement computed by Placement(δ), we have $D(p_1, \ldots, p_n) \geq \delta$ because all the points $p_i \in \delta\Lambda$. Therefore, for $1 \leq \delta^* \leq 2$, we can

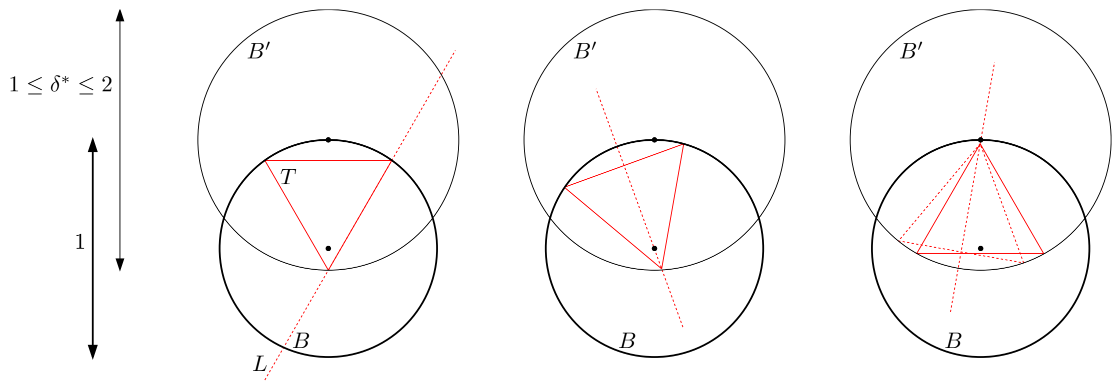

Figure 9: Illustration of the proof of Lemma 16.

get an approximation ratio of

$$\frac{\delta^*}{\delta} \geq \frac{4\delta^*}{-\sqrt{3}+\sqrt{3}\delta^*+\sqrt{3+2\delta^*-{\delta^*}^2}}.$$

For any $\delta^* \leq 1$, the second part of Lemma 16 implies that Centers gives a 2-approximation.

On the other hand, we have the trivial approximation algorithm Centers consisting of placing each point $p_i := c_i$, which gives a $\frac{\delta^*}{\delta^*-1}$-approximation when $\delta^* > 1$. In particular, Centers gives a 2-approximation when $\delta^* \geq 2$.

The idea is that the performances of Placement and Centers are reversed for different values δ^* in the interval $[1, 2]$. For example, when $\delta^* = 2$, the algorithm Placement gives a $\frac{4}{\sqrt{3}}$-approximation, while Centers gives a 2-approximation because the disks need to have disjoint interiors to achieve $\delta^* = 2$. But for $\delta^* = 1$, the performances are reversed: Placement gives a 2-approximation, while Centers does not give any constant factor approximation.

The approximation ratios of both algorithms are plotted in Figure 10. Applying both algorithms and taking the best of both solutions, we get an approximation ratio that is the minimum of both approximation ratios, which attains a maximum of

$$\alpha := 1 + \frac{13}{\sqrt{65+26\sqrt{3}}} \sim 2.2393.$$

Theorem 17 *Let $\mathcal{B} = \{B_1, \ldots, B_n\}$ be a collection of* congruent *disks in the plane with the L_2 metric. We can compute in $O(n^2)$ time a placement $p_1, \ldots, p_n$ with $p_i \in B_i$ that yields a ~ 2.2393-approximation of $D(\mathcal{B})$.*

Proof: The x-coordinate of the intersection of the two curves plotted in Figure 10 is given by

$$\frac{4\delta^*}{-\sqrt{3}+\sqrt{3}\delta^*+\sqrt{3+2\delta^*-{\delta^*}^2}} = \frac{\delta^*}{\delta^*-1}.$$

This solves to $\delta^* := \frac{1}{13}\left(13+\sqrt{13(5+2\sqrt{3}}\right)$, and therefore the approximation ratio is given by $\frac{\delta^*}{\delta^*-1} = \alpha \sim 2.2393$. ⊛

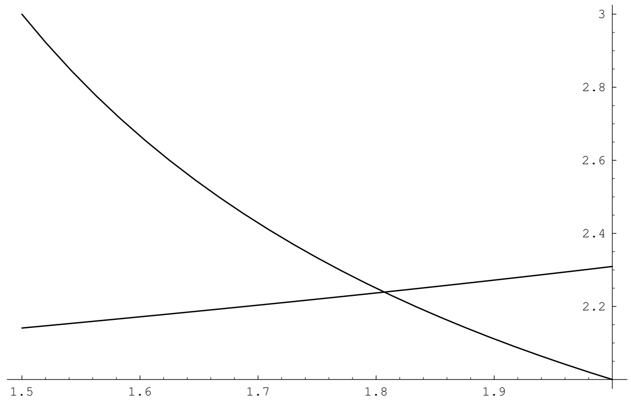

Figure 10: Approximation ratios for both approximation algorithms as a function of the optimum δ^*.

5 Concluding remarks

We have presented near quadratic time algorithms for the problem of spreading points. Our approximation ratios rely on packing arguments for the balls in the corresponding metric. However, in the running time of our results, we did not use the regularity of the point sets that we considered. An appropriate use of this property may lead to better running times, perhaps designing data structures for this particular setting.

In the proof of Lemma 14, the bottleneck of the computation is that we construct D explicitly. Instead, we could apply randomized binary search. For this to work out, we need, for given values δ, δ' and a disk B_i, to take a random point in the set $\tilde{P}_i = \{p \in \Lambda \,|\, \delta p \in B_i \text{ and } \delta' p \notin B_i, \text{ or vice versa}\}$. For the L_2 metric, we constructed $\bigcup_{i=1}^n \tilde{P}_i$ explicitly in quadratic time, and we do not see how to take a random sample in sub-quadratic time.

The approximate decision problem can be seen as using lattice packings to place disks inside the Minkowski sum $\bigcup_{i=1}^n B_i \oplus B(0, d/2)$. In the L_2 metric, we have used the lattice inducing the hexagonal packing, but we could use a different lattice. For the L_2 metric, the rectangular packing gives a worse approximation ratio, and it seems natural to conjecture that the hexagonal packing provides the best among regular lattices. On the other hand, deciding if better approximation ratios can be achieved using packings that are not induced by regular lattices seems a more challenging problem.

Acknowledgements

The author wishes to thank Pankaj Agarwal, Marc van Kreveld, Mark Overmars, and Günter Rote for several fruitful comments that improved the content of the paper. Thanks again to Marc for correcting the draft version and pointing out Lemma 6, which improved the previous running time. This research was initiated when Sergio was visiting DIMACS and Duke University, which provided good atmosphere for it.

References

[1] P. K. Agarwal, A. Efrat, and M. Sharir. Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. *SIAM J. Comput.*, 29:912–953, 1999.

[2] P. K. Agarwal and J. Matoušek. Ray shooting and parametric search. *SIAM J. Comput.*, 22(4):794–806, 1993.

[3] R. Aharoni and P. Haxell. Hall's theorem for hypergraphs. *Journal of Graph Theory*, 35:83–88, 2000.

[4] C. Baur and S. Fekete. Approximation of geometric dispersion problems. *Algorithmica*, 30:451–470, 2001. A preliminary version appeared in *APPROX'98*.

[5] S. Cabello and M. van Kreveld. Approximation algorithms for aligning points. *Algorithmica*, 37:211–232, 2003. A preliminary version appeared in *SoCG'03*.

[6] B. Chandra and M. M. Halldórsson. Approximation algorithms for dispersion problems. *J. Algorithms*, 38:438–465, 2001.

[7] B. Dent. *Cartography: Thematic Map Design*. McGraw-Hill, 5th edition, 1999.

[8] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing*. Prentice Hall, Upper Saddle River, NJ, 1999.

[9] A. Efrat, A. Itai, and M. J. Katz. Geometry helps in bottleneck matching and related problems. *Algorithmica*, 31(1):1–28, 2001.

[10] A. Efrat, M. J. Katz, F. Nielsen, and M. Sharir. Dynamic data structures for fat objects and their applications. *Comput. Geom. Theory Appl.*, 15:215–227, 2000. A preliminary version appeared in *WADS'97, LNCS 1272*.

[11] S. Fekete and H. Meijer. Maximum dispersion and geometric maximum weight cliques. To appear in *Algorithmica* 38(3), 2004.

[12] J. Fiala, J. Kratochvíl, and A. Proskurowski. Geometric systems of disjoint representatives. In *Graph Drawing, 10th GD'02, Irvine, California*, number 2528 in Lecture Notes in Computer Science, pages 110–117. Springer Verlag, 2002.

[13] J. Fiala, J. Kratochvíl, and A. Proskurowski. Systems of sets and their representatives. Technical Report 2002-573, KAM-DIMATIA, 2002. Available at `http://dimatia.mff.cuni.cz/`.

[14] A. V. Goldberg, S. A. Plotkin, and P. M. Vaidya. Sublinear-time parallel algorithms for matching and related problems. *Journal of Algorithms*, 14:180–213, 1993. A preliminary version appeared in *FOCS'88*.

[15] J. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973.

[16] K. Kedem, R. Livne, J. Pach, and M. Sharir. On the union of Jordan regions and collision-free translational motion amidst polygonal obstacles. *Discrete Comput. Geom.*, 1:59–71, 1986.

[17] N. Megiddo. Combinatorial optimization with rational objective functions. *Math. Oper. Res.*, 4:414–424, 1979.

[18] N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *J. ACM*, 30(4):852–865, 1983.

[19] C. W. Mortensen. Fully-dynamic two dimensional orthogonal range and line segment intersection reporting in logarithmic time. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 618–627. Society for Industrial and Applied Mathematics, 2003.

[20] C. W. Mortensen. Personal communication, 2003.

[21] A. Schrijver. A course in combinatorial optimization. Lecture Notes. Available at `http://homepages.cwi.nl/~lex/files/dict.ps`, 2003.

[22] B. Simons. A fast algorithm for single processor scheduling. In *Proc. 19th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 246–252, 1978.