Approximation Algorithms for Spreading Points*

Sergio Cabello

Institute for Mathematics, Physics and Mechanics,
Department of Mathematics, Ljubljana, Slovenia
sergio.cabello@imfm.uni-lj.si

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, similar ideas yield 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. [11, 12]: given a collection of subsets of a metric space and a value $\delta > 0$, we want a representative of each subset such that 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 [1]. 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. [11, 12], and independently by Baur and Fekete [3], the problem of deciding the existence of distant representatives is NP-hard even in the plane under natural metrics. This problem naturally embeds within the context of packing and map labelling problems, which has a much longer history; see the discussion in [3] and references therein.

However, 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.

The geometric optimization problem under consideration finds applications in cartography [7], graph drawing [8], and more generally in data visualization,

* Extended abstract. A full version is available as [4]. This research was done as PhD student at the Institute of Information and Computing Sciences, Utrecht University, partially supported by Cornelis Lely Stichting, NWO, and DIMACS.

G. Persiano and R. Solis-Oba (Eds.): WAOA 2004, LNCS 3351, pp. 83–96, 2005.
© Springer-Verlag Berlin Heidelberg 2005

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, and the problem can be abstracted as follows. We want to place a fixed number of points (0-dimensional cartographic features) 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. [11, 12] showed that this problem is NP-hard in the Euclidean and Manhattan metrics. Furthermore, by repeating at regular intervals their construction [11–Figures 1 and 2], it follows from the slackness of the construction that, unless $NP = P$, there is a certain constant $T > 1$ such that no T-approximation is possible. See [13] for a similar argument related to the slackness. They also notice that the one dimensional problem can be solved using the scheduling algorithm by Simons [17].

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 [3] (see also [6, 10]), 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 $P = NP$.

Dispersion problems have also been considered in arbitrary metric spaces and with various optimization functions; see [6, 14] and references therein.

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

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

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. Minimum diameter covering problems try to minimize the diameter of the placement [2]. 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. 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 [16] and the technique by Efrat et al. [9] for computing a matching in geometric settings. See Sections 2 and 3 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 is done by a two-stage binary search on some special values by paying an extra logarithmic factor; see Section 4.

Section 5 *summarizes* in L_2 the results that are equivalent to those described in L_∞. In particular, the same idea of "approximate-placement" can be used in the L_2 metric, but the approximation ratio becomes 8/3 and the running time increases to $O(n^2)$. Using binary search leads to an (8/3)-approximation algorithm for the L_2 metric.

However, when we restrict ourselves to congruent disks in L_2, a trivial adaptation of the techniques gives an approximation ratio of ~ 2.2393. This is explained in Section 6.

2 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. In this section we present an algorithm and discuss its approximation performance, while in next section we discuss a more efficient version of it.

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 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 Fig. 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 Fig. 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.

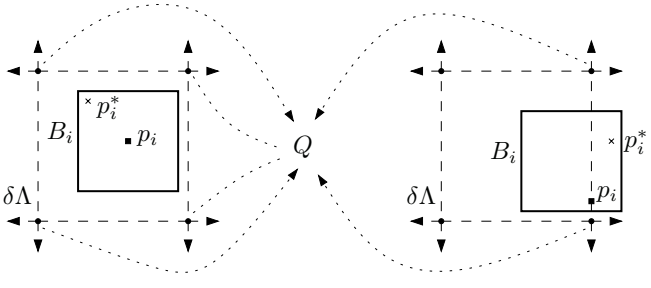

Fig. 1. Special cases where the disk B_i does not contain any lattice point. Left: B_i is fully contained in a cell of $\delta\Lambda$. Right: B_i intersects an edge of $\delta\Lambda$

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

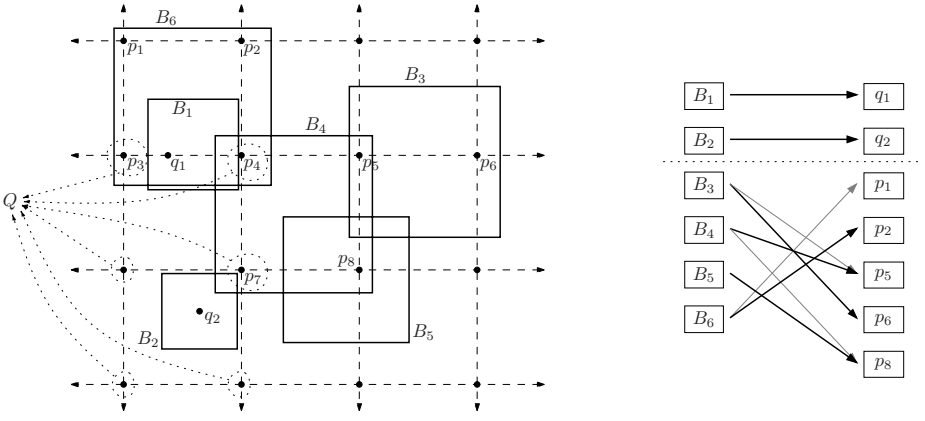

Fig. 2. Example showing the main features of the placement algorithm in L_∞

placing each point in the center of its disk. We call this algorithm PLACEMENT. 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 $D(p_1, \ldots, p_n)$ finding a closest pair in the placement. Below we show that, whenever $2\delta \leq \delta^*$, 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 $2\delta > \delta^*$.

Definition 1. *In the L_∞ metric,* 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.

Lemma 1. *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 are at 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 $d(p_i, p_j) \geq \delta$.

□

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 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 $B_1, \ldots, B_n$ are affected. Therefore, for an infinitesimal $\epsilon > 0$, we can decide if PLACEMENT($\delta+\epsilon$) succeeds or fails.

Observation 2. *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.*

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

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.*

3 Efficiency of the Placement Algorithm in L_∞

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 3. *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.*

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. However, the graph G_δ does not need to be constructed explicitly because its edges are implicitly represented by the disk-point containment. This type of matching problem, when both sets have the same cardinality, has been considered by Efrat et al. [9]. 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 the data structure designed by Mortensen [16] to get the following result.

Lemma 4. *In L_∞,* PLACEMENT *can be adapted to run in $O(n\sqrt{n}\log n)$ time.*

4 Approximation Algorithms for L_∞

We want a value $\tilde{\delta}$ such that PLACEMENT($\tilde{\delta}$) succeeds, but PLACEMENT($\tilde{\delta}+\epsilon$) fails for an infinitesimally small $\epsilon > 0$; then $\tilde{\delta}$ is a 2-approximation because of Observation 2. General techniques based on Megiddo's parametric search [15] are not very fruitful in this case because the known parallel algorithms for computing maximum matchings do not have an appropriate tradeoff between the number of processors and the running time.

Instead, we can 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. First, we state how to solve a special type of instances, and then present the main result of this section.

Lemma 5. *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. (Sketch) 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. 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 2, and we are done. 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 2. 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 4, and we have spent $O(m\sqrt{m}\log^2 mk)$ time in total. Let $\delta_1 := \frac{R}{t'}$ and $\delta_2 := \frac{R}{t'-1}$. This finishes the second part of the proof.

Before we start the third part, let us state, without proof, the property of δ_1, δ_2 that we will use later. 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.

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, and each disk B_i induces $O(1)$ such critical values Δ_i changing the flow of control of PLACEMENT.

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, we know that PLACEMENT$(\delta_3 + \epsilon)$ also fails, and conclude that PLACEMENT(δ_3) yields a 2-approximation because of Observation 2. □

Theorem 1. *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 either PLACEMENT$(\frac{r_{max+1}}{3\sqrt{n}})$ fails or $r_{max} = r_n$. This takes $O(n\sqrt{n}\log n)$ 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 2. 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 can use Lemma 5, 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 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 overall.

□

5 Analogous Results in L_2

The rest of the paper studies how the L_2 metric changes the approximation ratio and running time of the algorithms studied for the L_∞ metric. We just give the main observations, and refer to the full version for details.

5.1 Placement Algorithm in L_2

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

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 3a.
– 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 3b.

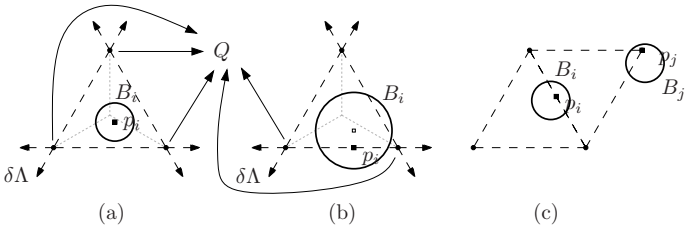

Fig. 3. Cases and properties of PLACEMENT for the L_2 metric. (a) Placement when B_i is fully contained in a cell. (b) Placement when B_i intersects an edge: we project the center c_i onto the closest edge. (c) A case showing that the closest pair in PLACEMENT(δ) may be at distance $\frac{\delta\sqrt{3}}{2}$

Observe that, in this case, the distance between a point placed on an edge and a point in $\delta\Lambda \setminus Q$ may be $\frac{\delta\sqrt{3}}{2}$; see Figure 3c. We modify accordingly the criteria of Definition 1 regarding when PLACEMENT succeeds, and then we state the result corresponding to Lemma 1.

Definition 2. *In the L_2 metric,* 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 6. *If $\frac{4\delta}{\sqrt{3}} \leq \delta^*$, then* PLACEMENT($\delta$) *succeeds.*

Observe that now, if PLACEMENT(δ) succeeds, but PLACEMENT($\delta + \epsilon$) fails for an infinitesimal $\epsilon > 0$, then we are getting an approximation ratio of $8/3$: we have $\delta \geq \frac{\delta^*\sqrt{3}}{4}$, and PLACEMENT($\delta$) gives a placement $p_1, \ldots, p_n$ that satisfies $D(p_1, \ldots, p_n) \geq \frac{\delta\sqrt{3}}{2} \geq \frac{3\delta^*}{8}$.

5.2 Approximation Algorithms in L_2

Lemma 3 also applies to the L_2 metric. However, the proof of Lemma 4 relies on some data structures that do not carry over to L_2. Instead we can show the following result, whose running time depends on whether the original disks are congruent or not.

Lemma 7. *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 proof of Lemma 5 is not valid for the L_2 metric because it relies on the fact that disks in the L_∞ metric are squares. Instead, we can solve in quadratic time the type of instances that are considered in Lemma 5. This leads to the following result, where we spend roughly $O(\sqrt{n})$ times more time than in the L_∞ case (Theorem 1).

Theorem 2. *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 $D(\mathcal{B})$.*

6 Congruent Disks in L_2

When the disks $B_1, \ldots, B_n$ are all congruent, say, of diameter one, we can improve the approximation ratio in Theorem 2. For general disks, the problematic cases are those balls that do not contain any lattice point. But when all the disks are of diameter one, we can rule out those cases.

Assume $1 \leq \delta^* \leq 2$ and take δ such that $\delta \leq \frac{-\sqrt{3}+\sqrt{3}\delta^*+\sqrt{3+2\delta^*-\delta^{*2}}}{4}$. When running PLACEMENT(δ) under this hypothesis, it is possible to show that $Q = \emptyset$ and that the graph G_δ has a matching. This requires some non-trivial geometric considerations; see the full version of the paper.

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 $Q = \emptyset$ and so all the points $p_i \in \delta\Lambda$. Therefore, for $1 \leq \delta^* \leq 2$, we can 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 $\delta^* \leq 1$, also some geometric considerations imply that PLACEMENT 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 $\delta^* \geq 2$.

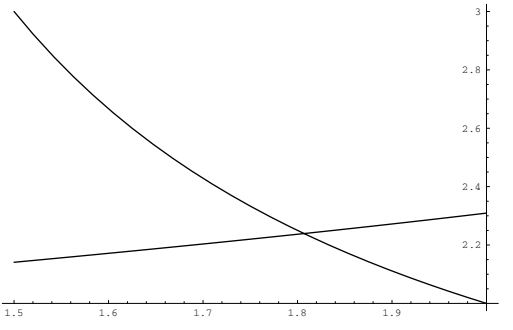

Fig. 4. Approximation ratios for both approximation algorithms as a function of the optimum δ^*

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 4. 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 3. *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 $D(\mathcal{B})$.*

Acknowledgements

The author wishes to thank Pankaj Agarwal, Refael Hassin, Marc van Kreveld, Mark Overmars, and Günter Rote for several fruitful comments that improved the content and shape of the paper. Thanks again to Marc for correcting the draft version and pointing out Lemma 3, 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. R. Aharoni and P. Haxell. Hall's theorem for hypergraphs. *Journal of Graph Theory*, 35:83–88, 2000.
2. E. Arkin and R. Hassin. Minimum diameter covering problems. *Networks*, 36:147–155, 2000.
3. C. Baur and S. Fekete. Approximation of geometric dispersion problems. *Algorithmica*, 30:451–470, 2001. A preliminary version appeared in *APPROX'98*.
4. S. Cabello. Approximation algorithms for spreading points. Technical report UU-CS-2003-040, Available at `http://www.cs.uu.nl/research/techreps/UU-CS-2003-040.html`, 2003.
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. S. Fekete and H. Meijer. Maximum dispersion and geometric maximum weight cliques. To appear in *Algorithmica* 38(3), 2004.
11. 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.

12. 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/`.
13. M. Formann and F. Wagner. A packing problem with applications to lettering of maps. In *Proc. 7th Annu. ACM Sympos. Comput. Geom.*, pages 281–288, 1991.
14. R. Hassin, S. Rubinstein, and A. Tamir. Approximation algorithms for maximum dispersion. *Operations Research Letters*, 21:133–137, 1997.
15. N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *J. ACM*, 30(4):852–865, 1983.
16. 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.
17. B. Simons. A fast algorithm for single processor scheduling. In *Proc. 19th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 246–252, 1978.