An Efficient Algorithm for 2D Euclidean 2-Center with Outliers[⋆]

Pankaj K. Agarwal and Jeff M. Phillips

Department of Computer Science, Duke University, Durham, NC 27708

Abstract. For a set P of n points in $\mathbb{R}^2$, the Euclidean 2-center problem computes a pair of congruent disks of the minimal radius that cover P. We extend this to the $(2,k)$-center problem where we compute the minimal radius pair of congruent disks to cover $n-k$ points of P. We present a randomized algorithm with $O(nk^7 \log^3 n)$ expected running time for the $(2,k)$-center problem. We also study the (p,k)-center problem in $\mathbb{R}^2$ under the ℓ_∞-metric. We give solutions for $p = 4$ in $O(k^{O(1)} n \log n)$ time and for $p = 5$ in $O(k^{O(1)} n \log^5 n)$ time.

1 Introduction

Let P be a set of n points in $\mathbb{R}^2$. For a pair of integers $0 \le k \le n$ and $p \ge 1$, a family of p congruent disks is called a *(p,k)-center* if the disks cover at least $n-k$ points of P; $(p,0)$-center is the standard p-center. The Euclidean (p,k)-center problems asks for computing a (p,k)-center of P of the smallest radius. In this paper we study the $(2,k)$-center problem. We also study the (p,k)-center problem under the ℓ_∞-metric for small values of p and k. Here we wish to cover all but k points of P by p congruent axis-aligned squares of the smallest side length. Our goal is to develop algorithms whose running time is $n(k \log n)^{O(1)}$.

Related work. There has been extensive work on the p-center problem in algorithms and operations research communities [4, 14, 18, 9]. If p is part of the input, the problem is NP-hard [22] even for the Euclidean case in $\mathbb{R}^2$. The Euclidean 1-center problem is known to be LP-type [20], and therefore can be solved in linear time for any fixed dimension. The Euclidean 2-center problem is not LP-type. Agarwal and Sharir [3] proposed an $O(n^2 \log^3 n)$ time algorithm for the 2-center problem. The running time was improved to $O(n \log^{O(1)} n)$ by Sharir [24]. The exponent of the $\log n$ factor was subsequently improved in [15, 6]. The best known deterministic algorithm takes $O(n \log^2 n \log^2 \log n)$ time in the worst case, and the best known randomized algorithm takes $O(n \log^2 n)$ expected time.

There is little work on the (p,k)-center problem. Using a framework described by Matoušek [19], the $(1,k)$-center problem can be solved in $O(n \log k + k^3 n^\varepsilon)$

⋆ This work is supported by NSF under grants CNS-05-40347, CFF-06-35000, and DEB-04-25465, by ARO grants W911NF-04-1-0278 and W911NF-07-1-0376, by an NIH grant 1P50-GM-08183-01, by a DOE grant OEGP200A070505, and by a grant from the U.S. Israel Binational Science Foundation.

time for any $\varepsilon > 0$. In general, Matoušek shows how to solve this problem with k outliers in $O(nk^d)$ time where d is the inherent number of constraints in the solution. The bound for the $(1,k)$-center problem is improved by Chan [7] to $O(n\beta(n)\log n + k^2n^\varepsilon)$ expected time, where $\beta(\cdot)$ is a slow-growing inverse-Ackermann-like function and $\varepsilon > 0$.

The p-center problem under ℓ_∞-metric is dramatically simpler. Sharir and Welzl [25] show how to compute the ℓ_∞ p-center in near-linear time for $p \leq 5$. In fact, they show that the rectilinear 2- and 3-center problems are LP-type problems and can be solved in $O(n)$ time. Also, they show the 1-dimensional version of the problem is an LP-type problem for any p, with combinatorial dimension $O(p)$. Thus applying Matoušek's framework [19], the ℓ_∞ (p,k)-center in $\mathbb{R}^2$ for $p \leq 3$, can be found in $O(k^{O(1)}n)$ time and in $O(k^{O(p)}n)$, for any p, if the points lie in $\mathbb{R}^1$.

Our results. Our main result is a randomized algorithm for the Euclidean $(2,k)$-center problem in $\mathbb{R}^2$ whose expected running time is $O(nk^7\log^3 n)$. We follow the general framework of Sharir and subsequent improvements by Eppstein. To handle outliers we first prove, in Section 2, a few structural properties of levels in an arrangement of unit disks, which are of independent interest.

As in [24, 15], our solution breaks the $(2,k)$-center problem into two cases depending on the distance between the centers of the optimal disks; (i) the centers are further apart than the optimal radius, and (ii) they are closer than their radius. The first subproblem, which we refer to as the *well-separated case* and describe in Section 3, takes $O(k^6 n\log^3 n)$ time in the worst case and uses parametric search [21]. The second subproblem, which we refer to as the *nearly concentric case* and describe in Section 4, takes $O(k^7 n\log^3 n)$ expected time. Thus we solve the $(2,k)$-center problem in $O(k^7 n\log^3 n)$ expected time. We can solve the nearly concentric case and hence the $(2,k)$-center problem in $O(k^7 n^{1+\delta})$ deterministic time, for any $\delta > 0$. We present near-linear algorithms for the ℓ_∞ (p,k)-center in $\mathbb{R}^2$ for $p = 4, 5$. The ℓ_∞ $(4,k)$-center problem takes $O(k^{O(1)}n\log n)$ time, and the ℓ_∞ $(5,k)$-center problem takes $O(k^{O(1)}n\log^5 n)$ time. See the full version [2] for the description of these results. We have not made an attempt to minimize the exponent of k. We believe that it can be improved by a more careful analysis.

2 Arrangement of Unit Disks

Let $\mathcal{D} = \{D_1, \ldots, D_n\}$ be a set of n unit disks in $\mathbb{R}^2$. Let $\mathcal{A}(\mathcal{D})$ be the arrangement of $\mathcal{D}$.[1] $\mathcal{A}(\mathcal{D})$ consists of $O(n^2)$ vertices, edges, and faces. For a subset $\mathcal{R} \subseteq \mathcal{D}$, let $\mathcal{I}(\mathcal{R}) = \bigcap_{D\in\mathcal{R}} D$ denote the intersection of disks in $\mathcal{R}$. Each disk in $\mathcal{R}$ contributes at most one edge in $\mathcal{I}(\mathcal{R})$. We refer to $\mathcal{I}(\mathcal{R})$ as a *unit-disk polygon* and a connected portion of $\partial\mathcal{I}(\mathcal{R})$ as a *unit-disk curve*. We introduce the notion

[1] The *arrangement* of $\mathcal{D}$ is the planar decomposition induced by $\mathcal{D}$; its vertices are the intersection points of boundaries of two disks, its edges are the maximal portions of disk boundaries that do not contain a vertex, and its faces are the maximal connected regions of the plane that do not intersect the boundary of any disk.

of level in $\mathcal{A}(\mathcal{D})$, prove a few structural properties of levels, and describe an algorithm that will be useful for our overall algorithm.

Levels and their structural properties. For a point $x \in \mathbb{R}^2$, the *level* of x with respect to $\mathcal{D}$, denoted by $\lambda(x, \mathcal{D})$, is the number of disks in $\mathcal{D}$ that *do not* contain x. (Our definition of level is different from the more common definition in which it is defined as the number of disks whose interiors contain x.) All points lying on an edge or face ϕ of $\mathcal{A}(\mathcal{D})$ have the same level, which we denote by $\lambda(\phi)$. For $k \leq n$, let $\mathcal{A}_k(\mathcal{D})$ (resp. $\mathcal{A}_{\leq k}(\mathcal{D})$) denote the set of points in $\mathbb{R}^2$ whose level is k (resp. at most k); see Fig. 1. By definition, $\mathcal{A}_0(\mathcal{D}) = \mathcal{A}_{\leq 0}(\mathcal{D}) = \mathcal{I}(\mathcal{D})$.

The boundary of $\mathcal{A}_{\leq k}(\mathcal{D})$ is composed of the edges of $\mathcal{A}(\mathcal{D})$. Let $v \in \partial D_1 \cap \partial D_2$, for $D_1, D_2 \in \mathcal{D}$, be a vertex of $\partial\mathcal{A}_{\leq k}(\mathcal{D})$. We call v *convex* (resp. *concave*) if $\mathcal{A}_{\leq k}(\mathcal{D})$ lies in $D_1 \cap D_2$ (resp. $D_1 \cup D_2$) in a sufficiently small neighborhood of v. $\partial\mathcal{A}_{\leq 0}(\mathcal{D})$ is composed of convex vertices; see Fig. 1(a). We define the complexity of $\mathcal{A}_{\leq k}(\mathcal{D})$ to be the number of edges of $\mathcal{A}(\mathcal{D})$ whose levels are at most k. Since the complexity of $\mathcal{A}_{\leq 0}(\mathcal{D})$ is n, the following lemma follows from the result by Clarkson and Shor [11] (see also Sharir [23] and Chan [8]).

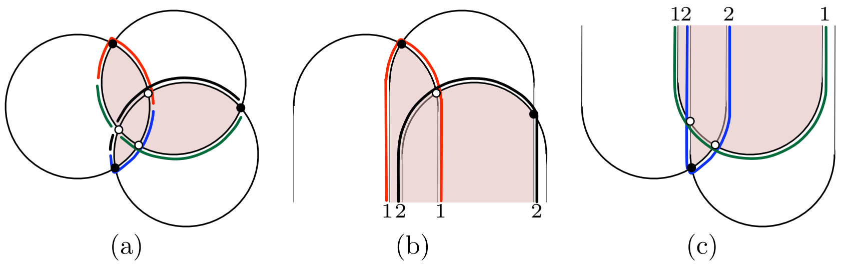

Fig. 1. (a) $\mathcal{A}(\mathcal{D})$, shaded region is $\mathcal{A}_{\leq 1}(\mathcal{D})$, filled (resp. hollow) vertices are convex (resp. concave) vertices of $\mathcal{A}_{\leq 1}(\mathcal{D})$, covering of $\mathcal{A}_{\leq 1}(\mathcal{D})$ edges by six unit-disk curves. (b) $\mathcal{A}(\Gamma^+)$, shaded region is $\mathcal{A}_{\leq 1}(\Gamma^+)$, and the covering of $\mathcal{A}_{\leq 1}(\Gamma^+)$ edges by two concave chains. (c) $\mathcal{A}(\Gamma^-)$, shaded region is $\mathcal{A}_{\leq 1}(\Gamma^-)$, and the covering of $\mathcal{A}_{\leq 1}(\Gamma^-)$ edges by two convex chains.

Lemma 1. [11] *For $k \geq 0$, the complexity of $\mathcal{A}_{\leq k}(\mathcal{D})$ is $O(nk)$.*

Remark 1. The argument by Clarkson and Shor can also be used to prove that $\mathcal{A}_{\leq k}(\mathcal{D})$ has $O(k^2)$ connected components and that it has $O(k^2)$ local minima in $(+y)$-direction. See also [19, 10]. These bounds are tight in the worst case; see Fig. 2.

It is well known that the edges in the $\leq k$-level of a line arrangement can be covered by $k+1$ concave chains [17], as used in [13, 7]. We prove a similar result for $\mathcal{A}_{\leq k}(\mathcal{D})$; it can be covered by $O(k)$ unit-disk curves.

For a disk D_i, let γ_i^+ (resp. γ_i^-) denote the set of points that lie in or below (resp. above) D_i; $\partial\gamma_i^+$ consists of the upper semicircle of ∂D_i plus two vertical downward rays emanating from the left and right endpoints of the semicircle — we refer to these rays as left and right rays. The curve $\partial\gamma_i^-$ has a similar structure. See Fig. 1(b). Set $\Gamma^+ = \{\gamma_i^+ \mid 1 \le i \le n\}$ and $\Gamma^- = \{\gamma_i^- \mid 1 \le i \le n\}$. Each pair of curves $\partial\gamma_i^+, \partial\gamma_j^+$ intersect in at most one point. (If we assume that the left and right rays are not vertical but have very large positive and negative slopes, respectively, then each pair of boundary curves intersects in exactly one point.) We define the level of a point with respect to Γ^+, Γ^-, or $\Gamma^+ \cup \Gamma^-$ in the same way as with respect to $\mathcal{D}$. A point lies in a disk D_i if and only if it lies in both γ_i^+ and γ_i^-, so we obtain the following inequalities:

$$\max\{\lambda(x,\Gamma^+), \lambda(x,\Gamma^-)\} \le \lambda(x,\mathcal{D}). \tag{1}$$

$$\lambda(x,\mathcal{D}) \le \lambda(x,\Gamma^+ \cup \Gamma^-) \le 2\lambda(x,\mathcal{D}). \tag{2}$$

We cover the edges of $\mathcal{A}_{\le k}(\Gamma^+)$ by $k+1$ concave chains as follows. The level of the $(k+1)$st rightmost left ray is at most k at $y = -\infty$. Let ρ_i be such a ray, belonging to γ_i^+. We trace $\partial\gamma_i^+$, beginning from the point at $y = -\infty$ on ρ_i, as long as $\partial\gamma_i^+$ remains in $\mathcal{A}_{\le k}(\Gamma^+)$. We stop when we have reached a vertex $v \in \mathcal{A}_{\le k}(\Gamma^+)$ at which it leaves $\mathcal{A}_{\le k}(\Gamma^+)$; v is a convex vertex on $\mathcal{A}_{\le k}(\Gamma^+)$. Suppose $v = \partial\gamma_i^+ \cap \partial\gamma_j^+$. Then $\partial\mathcal{A}_{\le k}(\Gamma^+)$ follows $\partial\gamma_j^+$ immediately to the right of v, so we switch to $\partial\gamma_j^+$ and repeat the same process. It can be checked that we finally reach $y = -\infty$ on a right ray. Since we switch the curve on a convex vertex, the chain Λ_i^+ we trace is a concave chain composed of a left ray, followed by a unit-disk curve ξ_i^+, and then followed by a right ray. Let $\Lambda_0^+, \Lambda_1^+, \ldots, \Lambda_k^+$ be the $k+1$ chains traversed by this procedure. These chains cover all edges of $\mathcal{A}_{\le k}(\Gamma^+)$, and each edge lies exactly on one chain. Similarly we cover the edges of $\mathcal{A}_{\le k}(\Gamma^-)$ by $k+1$ convex curves $\Lambda_0^-, \Lambda_1^-, \ldots, \Lambda_k^-$. Let $\Xi = \{\xi_0^+, \ldots, \xi_k^+, \xi_0^-, \ldots, \xi_k^-\}$ be the family of unit-disk curves induced by these convex and concave chains. By (1), Ξ covers all edges of $\mathcal{A}_{\le k}(\mathcal{D})$. Since a unit circle intersects a unit-disk curve in at most two points, we conclude the following.

Lemma 2. *The edges of $\mathcal{A}_{\le k}(\mathcal{D})$ can be covered by at most $2k+2$ unit-disk curves, and a unit circle intersects $O(k)$ edges of $\mathcal{A}_{\le k}(\mathcal{D})$.*

The curves in Ξ may contain edges of $\mathcal{A}(\mathcal{D})$ whose levels are greater that k. If we wish to find a family of unit-disk curves whose union is the set of edges in $\mathcal{A}_{\le k}(\mathcal{D})$, we proceed as follows. We add the x-extremal points of each disk as vertices of $\mathcal{A}(\mathcal{D})$, so each edge is now x-monotone and lies in a lower or an upper semicircle. By (1), only $O(k)$ such vertices lie in $\mathcal{A}_{\le k}(\mathcal{D})$. We call a vertex of $\mathcal{A}_{\le k}(\mathcal{D})$ *extremal* if it is an x-extremal point on a disk or an intersection point of a lower and an upper semicircle. Lemma 2 implies that there are $O(k^2)$ extremal vertices. For each extremal vertex v we do the following. If there is an edge e of $\mathcal{A}_{\le k}(\mathcal{D})$ lying to the right of v, we follow the arc containing e until we reach an extremal vertex or we leave $\mathcal{A}_{\le k}(\mathcal{D})$. In the former case we stop. In the latter case we are at a convex vertex v' of $\partial\mathcal{A}_{\leq k}(\mathcal{D})$, and we switch to the other arc incident on v' and continue. These curves have been drawn in Fig. 1(a). This procedure returns an x-monotone unit-disk curve that lies in $\mathcal{A}_{\leq k}(\mathcal{D})$. It can be shown that this procedure covers all edges of $\mathcal{A}_{\leq k}(\mathcal{D})$. We thus obtain the following:

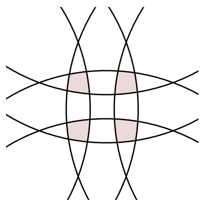
Fig. 2. Lower bound. $\mathcal{A}_{\leq 2}(\mathcal{D})$ (shaded region) has 4 connected components.

Lemma 3. *Let $\mathcal{D}$ be a set of n unit disks in $\mathbb{R}^2$. Given $\mathcal{A}_{\leq k}(\mathcal{D})$, we can compute, in time $O(nk)$, a family of $O(k^2)$ x-monotone unit-disk curves whose union is the set of edges of $\mathcal{A}_{\leq k}(\mathcal{D})$.*

Remark 2. Since $\mathcal{A}_{\leq k}(\mathcal{D})$ can consist of $\Omega(k^2)$ connected components, the $O(k^2)$ bound is tight in the worst case; see Fig. 2.

Emptiness detection of $\mathcal{A}_{\leq k}(\mathcal{D})$. We need a dynamic data structure for storing a set $\mathcal{D}$ of unit disks that supports the following two operations:

– (O1) Insert a disk into $\mathcal{D}$ or delete a disk from $\mathcal{D}$;
– (O2) For a given k, determine whether $\mathcal{A}_{\leq k}(\mathcal{D}) \neq \emptyset$.

As described by Sharir [24], $\mathcal{I}(\mathcal{D})$ can be maintained under insertion/deletion in $O(\log^2 n)$ time per update. Matoušek [19] has described a data structure for solving LP-type problems with violations. Finding the lowest point in $\mathcal{I}(\mathcal{D})$ can be formulated as an LP-type problem. Therefore using the dynamic data structure with Matoušek's algorithm, we can obtain the following result.

Lemma 4. *There exists a dynamic data structure for storing a set of n unit disks so that* (O1) *can be performed in $O(\log^2 n)$ time, and* (O2) *takes $O(k^3 \log^2 n)$ time.*

3 Well-Separated Disks

In this section we describe an algorithm for the case in which the two disks D_1, D_2 of the optimal solution are well separated. That is, let c_1 and c_2 be the centers of D_1 and D_2, and let r^* be their radius. Then $||c_1c_2|| \geq r^*$; see Fig. 3(a). Without loss of generality, let us assume that c_1 lies to the left of c_2. Let D_i^- be the semidisk lying to the left of the line passing through c_1 in direction normal to c_1c_2. A line ℓ is called a separator line if $D_1 \cap D_2 = \emptyset$ and ℓ separates D_1^- from D_2, or $D_1 \cap D_2 \neq \emptyset$ and ℓ separates D_1^- from the intersection points $\partial D_1 \cap \partial D_2$. We first show that we can quickly compute a set of $O(k^2)$ lines that contains a separator line. Next, we describe a decision algorithm, and then we describe the algorithm for computing D_1 and D_2 provided they are well separated.

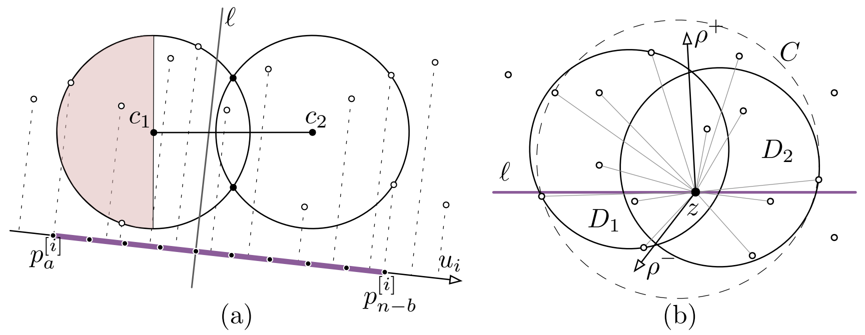

Fig. 3. (a) Let ℓ is a separator line for disks D_1 and D_2. (b) Two unit disks D_1 and D_2 or radius r^* with centers closer than a distance r^*.

Computing separator lines. We fix a sufficiently large constant h and choose a set $U = \{u_1, \ldots, u_h\} \subseteq \mathbb{S}^1$ of directions, where $u_i = (\cos(2\pi i/h), \sin(2\pi i/h))$.

For a point $p \in \mathbb{R}^2$ and a direction u_i, let $p^{[i]}$ be the projection of p in the direction normal to u_i. Let $P^{[i]} = \langle p_1^{[i]}, \ldots, p_n^{[i]} \rangle$ be the sorted sequence of projections of points in the direction normal to u_i. For each pair a, b such that $a+b \leq k$, we choose the interval $\delta_{a,b}^{[i]} = [p_a^{[i]}, p_{n-b}^{[i]}]$ and we place $O(1)$ equidistant points in this interval. See Fig. 3(a). Let $L_{a,b}^{[i]}$ be the set of (oriented) lines in the direction normal to u_i and passing though these points. Set

$$L = \bigcup_{\substack{1 \leq i \leq h \\ a+b \leq k}} L_{a,b}^{[i]}.$$

We claim that L contains at least one separator line. Intuitively, let $u_i \in U$ be the direction closest to $\overrightarrow{c_1c_2}$. Suppose p_a and p_{n-b} are the first and the last points of P in the direction u_i that lie inside $D_1 \cup D_2$. Since $|P \setminus (D_1 \cup D_2)| \leq k$, $a + b \leq k$. If $D_1 \cap D_2 = \emptyset$, then let q be the extreme points of D_1 in direction $\overrightarrow{c_1c_2}$. Otherwise, let q be the first intersection point of $\partial D_1 \cap \partial D_2$ in direction u_i. Following the same argument as Sharir [24], one can argue that

$$\langle c_1 - q, u_i \rangle \geq \alpha \langle p_{n-b} - p_a, u_i \rangle,$$

where $\alpha \leq 1$ is a constant. Hence if at least 2α points are chosen in the interval $\delta_{a,b}^{[i]}$, then one of the lines in $L_{a,b}^{[i]}$ is a separator line. Omitting all the details, which are similar to the one in [24], we conclude the following.

Lemma 5. *We can compute in $O(k^2 n \log n)$ time a set L of $O(k^2)$ lines that contains a separator line.*

Let D_1, D_2 be a $(2, k)$-center of P, let $\ell \in L$ be a line, and let $P^- \subseteq P$ be the set of points that lie in the left halfspace bounded by ℓ. We call D_1, D_2 a

$(2,k)$-center *consistent with* ℓ if $P^- \cap (D_1 \cup D_2) \subseteq D_1$, the center of D_1 lies to the left of ℓ, and ∂D_1 contains at least one point of P^-. We describe a decision algorithm that determines whether there is a $(2,k)$-center of unit radius that is consistent with ℓ. Next, we describe an algorithm for computing a $(2,k)$-center consistent with ℓ, which will lead to computing an optimal $(2,k)$-center of P, provided there is a well-separated optimal $(2,k)$-center of P.

Decision algorithm. Let $\ell \in L$ be a line. We describe an algorithm for determining whether there is a unit radius $(2,k)$-center of P that is consistent with ℓ. Let P^- (resp. P^+) be the subset of points in P that lie in the left (resp. right) halfspace bounded by ℓ; set $n^- = |P^-|$, $n^+ = |P^+|$. Suppose D_1, D_2 is a unit-radius $(2,k)$-center of P consistent with ℓ, and let c_1, c_2 be their centers. Then $P^- \cap (D_1 \cup D_2) \subseteq D_1$ and $|P^- \cap D_1| \geq n^- - k$. For a subset $Q \subset P$, let $\mathcal{D}(Q) = \{D(q) \mid q \in Q\}$ where $D(q)$ is the unit disk centered at q. Let $\mathcal{D}^- = \mathcal{D}(P^-)$ and $\mathcal{D}^+ = \mathcal{D}(P^+)$. For a point $x \in \mathbb{R}^2$, let $\mathcal{D}^+_x = \{D \in \mathcal{D}^+ \mid x \in D\}$. Since ∂D_1 contains a point of P^- and at most k points of P^- do not lie in D_1, c_1 lies on an edge of $\mathcal{A}_{\leq k}(\mathcal{D}^-)$.

We first compute $\mathcal{A}_{\leq k}(\mathcal{D}^-)$ in $O(nk \log n)$ time. For each disk $D \in \mathcal{D}^+$, we compute the intersection points of ∂D with the edges of $\mathcal{A}_{\leq k}(\mathcal{D}^-)$. By Lemma 2, there are $O(nk)$ such intersection points, and these intersection points split each edge into *edgelets*. The total number of edgelets is also $O(nk)$. Using Lemma 2, we can compute all edgelets in time $O(nk \log n)$. All points on an edgelet γ lie in the same subset of disks of $\mathcal{D}^+$, which we denote by $\mathcal{D}^+_\gamma$. Let $P^+_\gamma \subseteq P^+$ be the set of centers of disks in $\mathcal{D}^+_\gamma$, and let $\kappa_\gamma = \lambda(\gamma, \mathcal{D}^-)$. A unit disk centered at a point on γ contains P^+_γ and all but κ_γ points of P^-. If at least $k' = k - \kappa_\gamma$ points of $P^+ \setminus P^+_\gamma$ can be covered by a unit disk, which is equivalent to $\mathcal{A}_{\leq k'}(\mathcal{D}^+ \setminus \mathcal{D}_\gamma)$ being nonempty, then all but k points of P can be covered by two unit disks.

When we move from one edgelet γ of $\mathcal{A}_{\leq k}(\mathcal{D}^-)$ to an adjacent one γ' with σ as their common endpoint, then $\mathcal{D}^+_\gamma = \mathcal{D}^+_{\gamma'}$ (if σ is a vertex of $\mathcal{A}_{\leq k}(\mathcal{D}^-)$), $\mathcal{D}^+_{\gamma'} = \mathcal{D}^+_\gamma \cup \{D\}$ (if $\sigma \in \partial D$ and $\gamma' \subset \{D\}$), or $\mathcal{D}^+_{\gamma'} = \mathcal{D}^+_\gamma \setminus \{D\}$ (if $\sigma \in \partial D$ and $\gamma \subset \mathcal{D}$). We therefore traverse the graph induced by the edgelets of $\mathcal{A}_{\leq k}(\mathcal{D})$ and maintain $\mathcal{D}^+_\gamma$ in the dynamic data structure described in Section 2 as we visit the edgelets γ of $\mathcal{A}_{\leq k}(\mathcal{D}^-)$. At each step we process an edgelet γ, insert or delete a disk into $\mathcal{D}^+_\gamma$, and test whether $\mathcal{A}_{\leq j}(\mathcal{D}^+_\gamma) = \emptyset$ where $j = k - \lambda(\gamma, \mathcal{D}^-)$. If the answer is yes at any step, we stop. We spend $O(k^3 \log^2 n)$ time at each step, by Lemma 4. Since the number of edgelets is $O(nk)$, we obtain the following.

Lemma 6. *Let P be a set of n points in $\mathbb{R}^2$, ℓ a line in L, and $0 \leq k \leq n$ an integer. We can determine in $O(nk^4 \log^2 n)$ time whether there is a unit-radius $(2,k)$-center of P that is consistent with ℓ.*

Optimization algorithm. Let ℓ be a line in L. Let r^* be the smallest radius of a $(2,k)$-center of P that is consistent with ℓ. Our goal is to compute a $(2,k)$-center of P of radius r^* that is consistent with ℓ. We use the parametric search technique [21] — we simulate the decision algorithm generically at r^* and use the decision algorithm to resolve each comparison, which will be of the form: given $r_0 \in \mathbb{R}^+$, is $r_0 \leq r^*$? We simulate a parallel version of the decision procedure to reduce the number of times the decision algorithm is invoked to resolve a comparison. Note that we need to parallelize only those steps of the simulation that depend on r^*, i.e., that require comparing a value with r^*. Instead of simulating the entire decision algorithm, as in [15], we stop the simulation after computing the edgelets and return the smallest $(2, k)$-center found so far, i.e., the smallest radius for which the decision algorithm returned "yes." Since we stop the simulation earlier, we do not guarantee that we find the a $(2, k)$-center of P of radius r^* that is consistent with ℓ. However, as argued by Eppstein [15], this is sufficient for our purpose.

Let P^-, P^+ be the same as in the decision algorithm. Let $\mathcal{D}^-$, $\mathcal{D}^+$ etc. be the same as above except that each disk is of radius r^* (recall that we do not know the value of r^*). We simulate the algorithm to compute the edgelets of $\mathcal{A}_{\leq k}(\mathcal{D}^-)$ as follows. First, we compute the $\leq k^{th}$ order farthest point Voronoi diagram of P^- in time $O(n \log n + nk^2)$ [5]. Let e be an edge of the diagram with points p and q of P^- as its neighbors, i.e., e is a portion of the bisector of p and q. Then for each point $x \in e$, the disk of radius $||xp||$ centered at x contains at least $n^- - k$ points of P^-. We associate an interval $\delta_e = \{||xp|| \mid x \in e\}$. By definition, e corresponds to a vertex of $\mathcal{A}_{\leq k}(\mathcal{D}^-)$ if and only if $r^* \in \delta_e$; namely, if $||xp|| = r^*$, for some $x \in e$, then x is a vertex of $\mathcal{A}_{\leq k}(\mathcal{D}^-)$, incident upon the edges that are portions of $\partial D(p)$ and $\partial D(q)$. Let X be the sorted sequence of the endpoints of the intervals. By doing a binary search on X and using the decision procedure at each step, we can find two consecutive endpoints between which r^* lies. We can now compute all edges e of the Voronoi diagram such that $r^* \in \delta_e$. We thus compute all vertices of $\mathcal{A}_{\leq k}(\mathcal{D}^-)$. Since we do not know r^*, we do not have actual coordinates of the vertices. We represent each vertex as a pair of points. Similarly, each edge is represented as a point $p \in P^-$, indicating that e lies in $\partial D(p)$. Once we have all the edges of $\mathcal{A}_{\leq k}(P^-)$, we can construct the graph induced by them and compute $O(k^2)$ x-monotone unit-disk curves whose union is the set of edges in $\mathcal{A}_{\leq k}(P^-)$, using Lemma 3. Since this step does not depend on the value of r^*, we need not parallelize it. Let $\varXi = \{\xi_i, \ldots, \xi_u\}$, $u = O(k^2)$, be the set of these curves.

Next, for each disk $D \in \mathcal{D}^+$ and for each $\xi_i \in \varXi$, we compute the edges of ξ_i that ∂D intersects, using a binary search. We perform these $O(nk^2)$ binary searches in parallel and use the decision algorithm at each step. Incorporating Cole's technique [12] in the binary search we need to invoke the decision procedure only $O(\log n)$ times. For an edge $e \in \mathcal{A}_{\leq k}(\mathcal{D})$, let $\mathcal{D}_e^+ \in \mathcal{D}$ be the set of disks whose boundaries intersect e. We sort the disks in $\mathcal{D}_e^+$ by the order in which their boundaries intersect e. By doing this in parallel for all edges and using a parallel sorting algorithm for each edge, we can perform this step by invoking the decision algorithm $O(\log n)$ times. The total time spent is $O(nk^4 \log^3 n)$.

Putting pieces together. We repeat the optimization algorithm for all lines in L and return the smallest $(2, k)$-center that is consistent with a line in L. The argument of Eppstein [15] implies that if an optimal $(2, k)$-center of P is well-separated, then the above algorithm returns an optimal $(2, k)$-center of P. Hence, we conclude the following:

Lemma 7. *Let P be a set of n points in $\mathbb{R}^2$ and $0 \leq k \leq n$ an integer. If an optimal $(2, k)$-center of P is well separated, then the $(2, k)$-center problem for P can be solved in $O(nk^6 \log^3 n)$ time.*

4 Nearly Concentric Disks

In this section we describe an algorithm for when the two disks D_1 and D_2 of the optimal solution are not well separated. More specifically, let c_1 and c_2 be the centers of D_1 and D_2 and let r^* be their radius. Then this section handles the case where $||c_1c_2|| \leq r^*$.

First, we find an *intersector point* z of D_1 and D_2 — a point that lies in $D_1 \cap D_2$. We show how z defines a set $\mathcal{P}$ of $O(n^2)$ possible partitions of P into two subsets, such that for one partition $P_{i,j}$, $P \setminus P_{i,j}$ the following holds: $(D_1 \cup D_2) \cap P = (D_1 \cap P_{i,j}) \cup (D_2 \cap (P \setminus P_{i,j}))$. Finally, we show how to search through the set $\mathcal{P}$ in $O(k^7 n^{1+\delta})$ time, deterministically, for any $\delta > 0$, or in $O(k^7 n \log^3 n)$ expected time.

Finding an intersector point. Let C be the circumcircle of $P \cap (D_1 \cup D_2)$. Eppstein [15] shows that we can select $O(1)$ points inside C such that at least one, z, lies in $D_1 \cap D_2$. We can hence prove the following.

Lemma 8. *Let P be a set of n points in $\mathbb{R}^2$. We can generate in $O(nk^3)$ time a set $\mathcal{Z}$ of $O(k^3)$ points such that for any nearly concentric $(2, k)$-center D_1, D_2, one of the points in $\mathcal{Z}$ is their intersector point.*

Proof. If the circumcircle of P is not C, then at least one point of $P \cap \partial C$ must not be in $D_1 \cup D_2$. We remove each point and recurse until we have removed k points. Matoušek [19] shows that we can keep track of which subsets have already been evaluated and bounds the size of the recursion tree to $O(k^3)$. Building the entire recursion tree takes $O(nk^3)$ time. Since $|P \setminus C| \leq k$, at least one node in the recursion tree describes $P \cup C$. Generating $O(1)$ possible intersector points for each node completes the proof.

Let z be an intersector point of D_1 and D_2, and let ρ^+, ρ^- be the two rays from z to the points of $\partial D_1 \cap \partial D_2$. Since D_1 and D_2 are nearly concentric, the angle between them is at least some constant θ. We choose a set $U \subseteq S^1$ of $h = \lceil 2\pi/\theta \rceil$ uniformly distributed directions. For at least one $u \in U$, the line ℓ in direction u and passing through z separates ρ^+ and ρ^-, see Fig. 3(b). We fix a pair z, u in $Z \times U$ and compute a $(2, k)$-center of P, as described below. We repeat this algorithm for every pair. If D_1 and D_2 are nearly concentric, then our algorithm returns an optimal $(2, k)$-center.

Fixing z and u. For a subset $X \subset P$ and for an integer $t \geq 0$, let $r^t(X)$ denote the minimum radius of a $(1,t)$-center of X. Let P^+ (resp. P^-) be the subset of P lying above (resp. below) the x-axis; set $n^+ = |P^+|$ and $n^- = |P^-|$. Sort $P^+ = \langle p_1^+, \dots, p_{n^+}^+ \rangle$ in clockwise order and $P^- = \langle p_1^-, \dots, p_{n^-}^- \rangle$ in counterclockwise order. For $0 \leq i \leq n^+$, $0 \leq j \leq n^-$, let $P_{i,j} = \{p_1^+, \dots, p_i^+, p_1^-, \dots, p_j^-\}$ and $Q_{i,j} = P \setminus P_{i,j}$. For $0 \leq t \leq k$, let

$$m_{i,j}^t = \max\{r^t(P_{i,j}), r^{k-t}(Q_{i,j})\}.$$

For $0 \leq t \leq k$, we define an $n^+ \times n^-$ matrix M^t such that $M^t(i,j) = m_{i,j}^t$.

Suppose z is an intersector point of D_1 and D_2, ℓ separates ρ^+ and ρ^-, and ρ^+ (resp. ρ^-) lies between p_a^+, p_{a+1}^+ (resp. p_b^-, p_{b+1}^-). Then $P \cap (D_1 \cup D_2) = (P_{a,b} \cap D_1) \cup (Q_{a,b} \cup D_2)$; see Fig 3(b). If $|P_{a,b} \setminus D_1| = t$, then $r^* = m_{a,b}^t$. The problem thus reduces to computing

$$\mu(z,u) = \min_{i,j,t} m_{i,j}^t$$

where the minimum is taken over $0 \leq i \leq n^+$, $0 \leq j \leq n^-$, and $0 \leq t \leq k$. For each t, we compute $\mu^t(z,u) = \min_{i,j} m_{i,j}^t$ and choose the smallest among them.

Computing $\mu^t(z,u)$. We note two properties of the matrix M^t that will help search for $\mu^t(z,u)$:

– (P1) If $r^t(P_{i,j}) > r^{k-t}(Q_{i,j})$ then $m_{i,j}^t \leq m_{i',j'}^t$ for $i' \geq i$ and $j' \geq j$. These partitions only add points to $P_{i,j}$ and thus cannot decrease $r^t(P_{i,j})$. Similarly, if $r^{k-t}(Q_{i,j}) > r^t(P_{i,j})$, then $m_{i,j}^t < m_{i',j'}^t$ for $i' \leq i$ and $j' \leq j$.
– (P2) Given a value r, if $r^t(P_{i,j}) > r$, then $m_{i',j'}^t > r$ for $i' \geq i$ and $j' \geq j$, and if $r^t(Q_{i,j}) > r$, then $m_{i',j'}^t > r$ for $i' \leq i$ and $j' \leq j$.

Deterministic solution. We now have the machinery to use a technique of Frederickson and Johnson [16]. For simplicity, let us assume that $n^+ = n^- = 2^{\tau+1}$ where $\tau = \lceil \log_2 n \rceil + O(1)$. The algorithm works in τ phases. In the beginning of the hth phase we have a collection $\mathcal{M}_h$ of $O(2^h)$ submatrices of M^t, each of size $(2^{\tau-h+1}+1) \times (2^{\tau-h+1}+1)$. Initially $\mathcal{M}_1 = \{M^t\}$. In the hth phase we divide each matrix $N \in \mathcal{M}_h$ into four submatrices each of size $(2^{\tau-h}+1) \times (2^{\tau-h}+1)$ that overlap along one row and one column. We call the cell common to all four submatrices the center cell of N. Let $\mathcal{M}_h'$ be the resulting set of matrices. Let $\mathcal{C} = \{(i_1, j_1), \dots, (i_s, j_s)\}$ be the set of center cells of matrices in $\mathcal{M}_h$. We compute m_{i_l,j_l}^t for each $1 \leq l \leq s$. We use (P1) to remove the matrices of $\mathcal{M}_h$ that are guaranteed not to contain the value $\mu^t(z,u)$. In particular, if $m_{i_l,j_l}^t = r^t(P_{i_l,j_l})$ and there is a matrix $N \in \mathcal{M}_h'$ with the upper-left corner cell (i',j') such that $i' \geq i_l$ and $j' \geq j_l$, then we can remove N. Similarly if $m_{i_l,j_l}^t = r^{k-t}(Q_{i,j})$ and there is a matrix $N \in \mathcal{M}_h'$ with the lower-right corner cell (i',j') such that $i' \leq i_l$ and $j' \leq j_l$, we can delete N. It can be proved that after the pruning step if we have a matrix N in $\mathcal{M}_h'$ such that it spans $[a_1, a_2]$ rows and $[b_1, b_2]$ columns of M^t, then $m_{a_1,b_1}^t = r^t(P_{a_1,b_1})$ and $m_{a_2,b_2}^t = r^{k-t}(Q_{a_2,b_2})$. This implies that $O(n)$ cells remain in $\mathcal{M}_h'$ after the pruning step. We set $\mathcal{M}_h'$ to $\mathcal{M}_{h+1}$.

Finally, it is shown in [15] that the center cells in $\mathcal{C}$ can be connected by a monotone path in $\mathcal{M}^t$, which consists of $O(n)$ cells. Since $P_{i,j}$ differs from $P_{i-1,j}$ and $P_{i,j-1}$ by one point, we can compute $m^t_{i_l,j_l}$ for all $(i_l, j_l) \in \mathcal{C}$ using an algorithm of Agarwal and Matoušek [1] in total time $O(k^3 n^{1+\delta})$ for any $\delta > 0$. Agarwal and Matoušek's data structure can maintain the value of the radius of the smallest enclosing disk under insertions and deletions in $O(n^\delta)$ time per update. Each step in the path is one update, and then searching through the $O(k^3)$ nodes of the recursion tree of all possible outliers — each requires $O(1)$ updates — takes $O(k^3 n^\delta)$ time per cell. Hence, each phase of the algorithm takes $O(k^3 n^{1+\delta})$ time.

Lemma 9. *Given $z \in Z$, $u \in U$, and $0 \leq t \leq k$, $\mu^t(z, u)$ can be computed in time $O(k^3 n^{1+\delta})$, for any $\delta > 0$.*

Randomized solution. We can slightly improve the dependence on n by using the dynamic data structure in Section 2 and (P2). As before, in the hth phase, for some constant $c > 1$, we maintain a set $\mathcal{M}_h$ of at most $c2^h$ submatrices of M^t, each of side length $2^{\tau-h+1}+1$, and their center cells $\mathcal{C}$. Each submatrix is divided into four submatrices of side length $2^{\tau-h}+1$, forming a set $\mathcal{M}'_h$. To reduce the size of $\mathcal{M}'_h$, we choose a random center cell (i, j) from $\mathcal{C}$ and evaluate $r = m^t_{i,j}$ in $O(k^3 n)$ time. For each other center cell $(i', j') \in \mathcal{C}$, $m^t_{1',j'} > r$ with probability 1/2, and using (P2), we can remove a submatrix from $\mathcal{M}'_h$. Eppstein [15] shows that by repeating this process a constant number of times, we expect to reduce the size of $\mathcal{M}'_h$ to $c2^{h+1}$.

On each iteration we use the dynamic data structure described in Section 2. For $O(n)$ insertions and deletions, it can compare each center cell from $\mathcal{C}$ to r in $O(k^3 n \log^2 n)$ time. Thus, finding $\mu^t(z, u)$ takes expected $O(nk^3 \log^3 n)$ time.

Lemma 10. *Given $z \in Z$, $u \in U$, and $0 \leq t \leq k$, $\mu^t(z, u)$ can be computed in expected time $O(k^3 \log^3 n)$.*

Putting pieces together. By repeating either above algorithm for all $0 \leq t \leq k$ and for all pair $(z, u) \in Z \times U$, we can compute a $(2, k)$-center of P that is optimal if D_1 and D_2 are nearly concentric. Combining this with Lemma 7, we obtain the main result of the paper.

Theorem 1. *Given a set P of n points in $\mathbb{R}^2$ and an integer $k \geq 0$, an optimal $(2, k)$-center of P can be computed in $O(k^7 n^{1+\delta})$ (deterministic) time, for any $\delta > 0$ or in $O(k^7 n \log^3 n)$ expected time.*

Acknowledgements. We thank Sariel Har-Peled for posing the problem and for several helpful discussions.

References

1. P. K. Agarwal and J. Matoušek, Dynamic half-space range reporting and its applications, *Algorithmica*, 13 (1995), 325–345.

2. P. K. Agarwal and J. M. Phillips, An efficient algorithm for 2D Euclidean 2-center with outliers, arXiV:0806.4326, 2008.
3. P. K. Agarwal and M. Sharir, Planar geometric locations problems, *Algorithmica*, 11 (1994), 185–195.
4. P. K. Agarwal and M. Sharir, Efficient algorithms for geometric optimization, *ACM Computing Surveys*, 30 (1998), 412–458.
5. A. Aggarwal, L. J. Guibas, J. Saxe, and P. W. Shor, A linear-time algorithm for computing the voronoi diagram of a convex polygon, *Discrete Comput. Geom.*, 4 (1989), 591–604.
6. T. Chan, More planar two-center algorithms, *Comput. Geom.: Theory Apps.*, 13 (1999), 189–198.
7. T. Chan, Low-dimensional linear programming with violations, *SIAM J. Comput.*, 34 (2005), 879–893.
8. T. Chan, On the bichromatic k-set problem, *Proc. 19th Annu. ACM-SIAM Sympos. Discrete Algs.*, 2007, pp. 561–570.
9. M. Charikar, S. Khuller, D. M. Mount, and G. Narasimhan, Algorithms for faciity location problems with outliers, *12th Annu. ACM-SIAM Sympos. on Discrete Algs.*, 2001, pp. 642–651.
10. K. L. Clarkson, A bound on local minima of arrangements that implies the upper bound theorem, *Discrete Comput. Geom.*, 10 (1993), 427–433.
11. K. L. Clarkson and P. W. Shor, Applications of random sampling in geometry, II, *Discrete Comput. Geom.*, 4 (1989), 387–421.
12. R. Cole, Slowing down sorting networks to obtain faster sorting algorithms, *Journal of ACM*, 34 (1987), 200–208.
13. T. K. Dey, Improved bounds for planar k-sets and related problems, *Discrete Comput. Geom.*, 19 (1998), 373–382.
14. Z. Drezner and H. Hamacher, *Facility Location: Applications and Theory*, Springer, 2002.
15. D. Eppstein, Faster construction of planar two-centers, *Proc. 8th Annu. ACM-SIAM Sympos. on Discrete Algs.*, 1997, pp. 131–138.
16. G. N. Frederickson and D. B. Johnson, The complexity of selection and ranking in $x+y$ and matrices with sorted columns, *J. Comput. Syst. Sci.*, 24 (1982), 197–208.
17. D. Gusfield, Bounds for the parametric minimum spanning tree problem, *Humboldt Conf. on Graph Theory, Combinatorics Comput.*, Utilitas Mathematica, 1979, pp. 173–183.
18. D. S. Hochbaum, ed., *Approximation Algorithms for NP-hard Problems*, PWS Publishing Company, 1995.
19. J. Matoušek, On geometric optimization with few violated constraints, *Discrete Comput. Geom.*, 14 (1995), 365–384.
20. J. Matoušek, E. Welzl, and M. Sharir, A subexponential bound for linear programming and related problems, *Algorithmica*, 16 (1996), 498–516.
21. N. Megiddo, Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems, *SIAM J. Comput.*, 12 (1983), 759–776.
22. N. Megiddo and K. J. Supowit, On the complexity of some common geometric location problems, *SIAM J. Comput.*, 12 (1983), 759–776.
23. M. Sharir, On k-sets in arrangement of curves and surfaces, *Discrete Comput. Geom.*, 6 (1991), 593–613.
24. M. Sharir, A near-linear time algorithm for the planar 2-center problem, *Discrete Comput. Geom.*, 18 (1997), 125–134.
25. M. Sharir and E. Welzl, Rectilinear and polygonal p-piercing and p-center problems, *Proc. 12th Annu. Sympos. Comput. Geom.*, 1996, pp. 122–132.