FINDING THE LARGEST DISK CONTAINING A QUERY POINT IN LOGARITHMIC TIME WITH LINEAR STORAGE[*]

Tal Kaminker[†] *and Micha Sharir*[‡]

Abstract. Let $\mathcal{D}$ be a set of n disks in the plane. We present a data structure of size $O(n)$ that can compute, for any query point q, the largest disk in $\mathcal{D}$ that contains q, in $O(\log n)$ time. The structure can be constructed in $O(n \log^3 n)$ time. The optimal storage and query time of the structure improve several recent results on this and related problems [1, 2, 4].

1 Introduction

Let $\mathcal{D}$ be a set of n disks in the plane. We present a data structure of size $O(n)$ that can compute, for any query point $q \in \mathbb{R}^2$, the largest disk in $\mathcal{D}$ that contains q, in $O(\log n)$ time. The structure can be constructed in $O(n \log^3 n)$ time.

For simplicity, we assume general position of the disks in $\mathcal{D}$, meaning, in particular, that all disks are of different sizes, and all the y-coordinates of the disk centers are distinct; we also assume distinctness of the coordinates in the directions $\pm\pi/6$. Finally, we assume that no query point lies on the boundary of any disk in $\mathcal{D}$. Degenerate situations where these assumptions do not hold can be handled by a variety of standard techniques, such as symbolic perturbations; see, e.g., [7].

Background. The problem of constructing an efficient data structure for finding the largest disk containing a query point appears to have been first considered by Augustine et al. [1] (see also the later version of their paper [2]), as a subproblem that arose in their solution of the problem of finding the largest disk containing a query point, under the condition that the disk does not contain any point of an n-element input point set P (the largest P-empty disk containing q). They presented two solutions for this problem. The first solution uses a divide-and-conquer approach that produces a data structure of size $O(n \log n)$ that can answer a query in $O(\log^2 n)$ time. Another solution uses a simple sweeping technique that produces a data structure of size $O(n^2)$ that can answer a query in $O(\log n)$ time. A subsequent paper by Kaplan and Sharir [4], considering the same problem studied in [1, 2],

[*]A preliminary version of this paper has appeared in *Proc. 30th Annu. ACM Sympos. Comput. Geom.*, 2014, 206–213. Work on this paper has been supported by Grant 2012/229 from the U.S.-Israeli Binational Science Foundation, by Grant 892/13 from the Israel Science Foundation, by the Israeli Centers of Research Excellence (I-CORE) program (Center No. 4/11), and by the Hermann Minkowski–MINERVA Center for Geometry at Tel Aviv University.

[†]*Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel*, tkaminker@gmail.com

[‡]*Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel*, michas@tau.ac.il

presents an improved solution that uses only near-linear storage, with $O(\log^2 n)$ query time. It is argued in [4] that the center of the largest P-empty disk containing a query point q must lie either on an edge of the Voronoi diagram of P, or at a Voronoi vertex (assuming that the query point lies inside the convex hull of P). Surprisingly, finding the largest P-empty disk containing q whose center lies in the relative interior of some Voronoi edge can be done in $O(\log n)$ time using a data structure of linear size [4]. In contrast, finding the largest P-empty disk containing q and centered at a Voronoi vertex (these are the Delaunay disks of P), with a structure of near-linear size, could only be done in [4] in $O(\log^2 n)$ time (using the same idea as in the first solution of Augustine et al. [1], which also requires $\Theta(n \log n)$ storage).

This leads to a special case of the problem studied in this paper: Given the $O(n)$ Delaunay disks of P, preprocess them into a data structure of linear size, so that the largest disk containing a query point can be found in $O(\log n)$ time. Kaplan and Sharir present some partial results, where the query time improves to $O(k \log n)$, for any prespecified paramenter k, but the storage grows to $O(n^{1+\frac{1}{k}})$.

To recap, the failure of the previous works to solve the problem with optimal query time and storage makes it an interesting challenge: Given n disks in the plane, construct a data structure of linear size that can find, in $O(\log n)$ time, the largest disk containing a query point.

Our algorithm meets this challenge, as specified. The only performance parameter of our algorithm that we do not know to be optimal is the preprocessing, which takes $O(n \log^3 n)$ time. In particular, this improves the result of [4], yielding an overall algorithm that preprocesses a set P of n points in the plane, in $O(n \log^3 n)$ time, into a data structure of linear size, that can find, in $O(\log n)$ time, the largest P-empty disk containing a query point.

We note that this problem is a special case of a more general range searching setting where ranges have priorities (in our case the size of the disk is its priority) and we want a data structure that can find the range of highest priority, containing a query point.

2 The algorithm

2.1 Overview

We first describe the data structure and its high-level construction. Then we present and analyze the querying procedure. Finally, we provide a detailed description of an efficient implementation of the construction of the structure.

Construction of the data structure. The algorithm divides each disk into three equal parts, called the *right*, *top*, and *bottom* parts, by the radii at orientations $\frac{\pi}{3}$, π, and $-\frac{\pi}{3}$ (see Figure 1). The algorithm will be run separately on all the right parts of the disks, on all the top parts, and on all the bottom parts. The resulting substructures will then be combined into one global structure. For simplicity, and with no loss of generality, we will only describe the algorithm for the right portions of the disks.

The algorithm constructs a planar map M_r, over the right portions, with the following **properties**:

1. Each disk contributes at most one connnected arc to M_r.

2. No two arcs in M_r cross each other[1].

3. For each $q \in \mathbb{R}^2$, let $D_{max}(q)$ denote the largest disk in $\mathcal{D}$ that contains q. Suppose that q lies in the right portion of $D_{max}(q)$. Then the first arc of M_r hit by the rightward-directed ray emanating from q is the arc that $D_{max}(q)$ contributes to M_r.

Properties (1) and (2) also hold for the two other maps. Property (3) holds too, except that the relevant ray emanates from q in direction $2\pi/3$ (for the map of the top parts) or in direction $-2\pi/3$ (for the map of the bottom parts).

Properties (1) and (2) imply that all the maps have linear complexity. Property (3) suggests that one should construct a point location structure on M_r for locating the first arc that lies straight to the right from a query point, and construct similar structures for the other two maps, with respect to the corresponding ray orientations $\pm 2\pi/3$ mentioned above. By combining the results from all three point location answers, one can easily find $D_{max}(q)$: it is the largest disk that contains q among the three retrieved disks. See Figure 2 for an illustration.

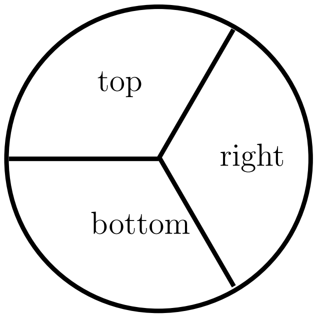

Figure 1: The partition of a disk into three equal parts; the orientation of the three dividing radii are the same for all disks.

2.2 Construction of M_r: High level description

This section presents a high-level description of the construction of the map M_r. An actual efficient implementation of this construction will be described later in Section 2.3.

As stated above, we divide each disk into three parts. Each part contributes at most one arc (a subarc of its boundary arc) to the corresponding map. As already stated, we focus only on the right part of each disk, and on the corresponding map M_r. For each disk $d \in \mathcal{D}$ we denote by T_d the right part of d, and by A_d the right arc (the arc bounding T_d).

[1]In general, an arc may terminate at a point that lies in the relative interior of another arc.

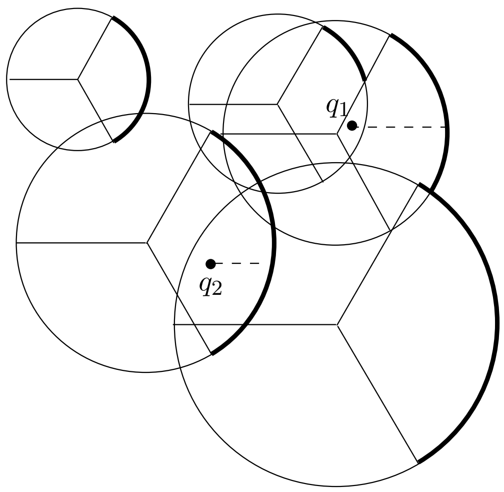

Figure 2: A set of disks and the corresponding map M_r (consisting of the thicker arcs). Note that the map returns the correct answer for q_1 but not for q_2, because q_2 does not lie in the right portion of the largest disk containing it.

For each arc A_d, we go over all disks larger than d, and each such disk d' might shrink A_d into a potentialy smaller subarc $A_d^{d'}$ (possibly eliminating it altogether). This yields, for each arc A_d, a set of subarcs $\left\{A_d^{d'} \mid d' \text{ is larger than } d\right\}$, and its intersection $A_d^* = \bigcap_{d'} A_d^{d'}$ (over the disks d' larger than d), which, as will be seen immediately, is a single (possibly empty) connected subarc, is added to M_r.
(This description can trivially be turned into an $O(n^2)$ algorithm for constructing M_r; a more efficient near-linear construction is described in Section 2.3.)

There are two rules for creating $A_d^{d'}$ from A_d and d', which depend on the number of connected components of $A_d \backslash T_{d'}$.

1. If $A_d \backslash T_{d'}$ consists of a single connected component[2] then $A_d^{d'} = A_d \backslash T_{d'}$; see Figure 3(a).

2. If $A_d \backslash T_{d'}$ consists of two connected components[3] then, if the center of disk d is higher (in the y-direction) than the center of d', then $A_d^{d'}$ is the top part of $A_d \backslash T_{d'}$; otherwise, $A_d^{d'}$ is the bottom part of $A_d \backslash T_{d'}$; see Figure 3(b).

Each of these rules creates a single (potentialy smaller) subarc of A_d, and thus, as already noted, $A_d^* = \bigcap_{d'} A_d^{d'}$ is a connected subarc of A_d. In some cases A_d^* might be empty, and then the arc A_d does not contribute anything to M_r, or A_d^* might be equal to A_d, and then the arc is not affected by other disks (for instance, this is always the case for the largest disk in $\mathcal{D}$). This implies the first property postulated for M_r. It is easy to see that, after the execution of these rules on an arc A_d and a disk d' which is larger than d, the resulting

[2]The trivial cases where $A_d \cap T_{d'} = \emptyset$ or $A_d \subset T_{d'}$ are also considered under this rule.
[3]It is easy to check that $A_d \setminus T_{d'}$ cannot have more than two connected components, because T_d and $T_{d'}$ are homothetic — see later in the paper.

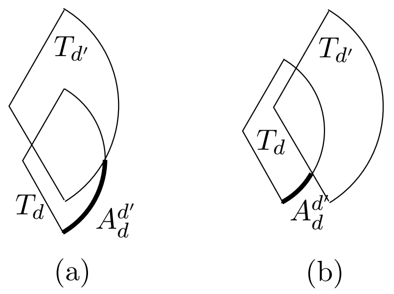

Figure 3: The two cases of the rule for constructing $A_d^{d'}$. (a) $A_d \setminus T_{d'}$ consists of one connected component. (b) $A_d \setminus T_{d'}$ consists of two connected components. In both cases $A_d^{d'}$ is the thick arc (the lowest portion of A_d).

subarc $A_d^{d'}$ does not cross the arc $A_{d'}$, although it might have one or both endpoints lying on $A_{d'}$. Since $A_d^* \subseteq A_d^{d'}$, this implies the second propery of M_r. To complete the analysis, we next establish property (3), the main property of the map.

Lemma 2.1. *For each $q \in \mathbb{R}^2$, let $d = D_{max}(q)$ be the largest disk of $\mathcal{D}$ that contains q. If q belongs to the right portion T_d of d, then the first arc of M_r to the right of q is A_d^*.*

Proof. First we note that it suffices to prove the lemma for the simpler case where there are only two disks, that is, $\mathcal{D} = \{d, d'\}$. Indeed, if the lemma were false, the ray emanating from q would not have hit A_d^* first. That is, it would have either hit first another arc $A_{d'}^*$, or reach infinity without hitting any arc, and then the point where the ray hits A_d (which always exists) must have been removed by another disk d''. It is easily seen that in either case, the same situation would arise in the presence of just d and the other disk d' or d''.

Suppose first that d' is smaller than d. In this case, by construction, A_d is not affected by d', and $A_{d'}^d$ does not enter the region T_d, thus rendering the lemma trivial. Suppose then that d' is larger than d, so $q \notin d'$. Note that in this case we have $A_d^* = A_d^{d'}$. We may assume, without loss of generality, that the center of d is lower (in the y-direction) than the center of d'; the case when the center of d is higher is handled in a fully symmetric manner. Let $r_{d'}$ and r_d denote the respective radii of d' and d.

We need to show that either $A_d^{d'}$ is the only arc that is to the right of q, or else it comes before (that is, to the left of) $A_{d'}$ (clearly, since d' is larger, $A_{d'}$ appears in its entirety in the map of only the disks d, d').

First, for any point $(x_0, y_0) \in \mathbb{R}^2$ and any compact geometric object A, define the (rightward) distance in the x-direction from (x_0, y_0) to A by

$$dist_x\left((x_0, y_0),\, A\right) = \min\left\{x - x_0 \mid (x, y_0) \in A,\, x \geq x_0\right\}.$$

The analysis relies on the following simple but crucial property. Consider the region $L_{d'}$ of all points p in the plane such that $dist_x(p,\, T_{d'}) \leq r_{d'}$ (see Figure 4). Then, as follows by simple geometry, $L_{d'} \subset d'$. We will also consider the region $K_{d'}$ of all points p in the plane such that $dist_x(p,\, A_{d'}) \leq r_{d'}$ (see Figure 5). Since $A_{d'} \subseteq T_{d'}$ we have $K_{d'} \subseteq L_{d'} \subset d'$.

Consider the rightward-directed ray v emanating from q. Clearly v hits A_d (since $q \in T_d$). Suppose that v hits $A_{d'}$ before it hits A_d. This implies that $dist_x(q, A_{d'}) < dist_x(q, A_d) \le r_d < r_{d'}$. Thus $q \in K_{d'} \subset d'$, contradicting our assumption. That is, either v misses $A_{d'}$ altogether, or v hits $A_{d'}$ after it hits A_d.

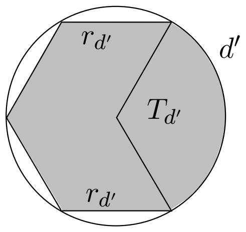

Figure 4: The (shaded) region $L_{d'}$ of all points p satisfying $dist_x(p, T_{d'}) \le r_{d'}$.

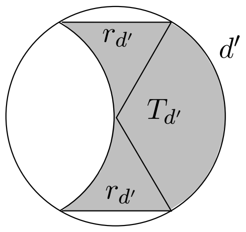

Figure 5: The (shaded) region $K_{d'}$ of all the points p satisfying $dist_x(p, A_{d'}) \le r_{d'}$.

It remains to show that the point I of intersection of v with A_d belongs to $A_d^* = A_d^{d'}$. We proceed according to the rule by which $A_d^{d'}$ was constructed.

Case 1: $A_d \backslash T_{d'}$ consists of a single connected component.

Recall that in this case $A_d^{d'} = A_d \backslash T_{d'}$. Assume to the contrary that $I \notin A_d^{d'}$, so $I \in T_{d'}$. But then $dist_x(q, T_{d'}) \le |qI| \le r_d < r_{d'}$. Thus $q \in L_{d'} \subset d'$, contrary to our assumption. Hence $I \in A_d^{d'}$ in this case.

Case 2: $A_d \backslash T_{d'}$ consists of two connected components.

Let c, c' be the top and bottom vertices of T_d, respectively. Since T_d and $T_{d'}$ are homothetic copies of each other, their boundaries intersect in two points (see [5]), both of which lie on A_d in this case[4]. This implies that either both c and c' lie inside $T_{d'}$ or both lie outside $T_{d'}$. If c and c' lie inside $T_{d'}$ then, as is easily checked, both intersection points of ∂T_d and $\partial T_{d'}$ also lie on $A_{d'}$, and the portion $A_d \setminus T_{d'}$ would have to be the middle portion of A_d (between the two intersection points), so $A_d \setminus T_{d'}$ would be connected, contrary to assumption. Using

[4]Our general position assumption allows us to assume that ∂T_d and $\partial T_{d'}$ do not overlap (in some straight segment).

the facts that both c and c' lie outside $T_{d'}$ and no more intersection points between ∂T_d and $\partial T_{d'}$ exist except the two on A_d, we conclude that the only part of ∂T_d that is inside $T_{d'}$ is the middle portion of A_d. See Figure 7.

Let l, l' denote the horizontal lines touching the top and bottom vertices of $T_{d'}$, respectively (see Figure 6). Clearly the top vertex c of T_d lies above l', and since the center of d is lower than the center of d', and $r_d < r_{d'}$, the point c has to lie below the line l. In conclusion c lies between l and l'. Observe also that c is behind (in the x-direction) $T_{d'}$, for otherwise, using the fact that both the top and the bottom straight edges of T_d are outside $T_{d'}$, it would be impossible for ∂T_d and $\partial T_{d'}$ to cross each other, contrary to assumption. Denote by a the top intersection point of A_d and $\partial T_{d'}$. Using the fact that c is outside and behind $T_{d'}$ and between l and l', and the fact that the top subarc $\overline{ac}$ of A_d does not cross $\partial T_{d'}$ (except touching it at a), we conclude that a lies on one of the straight edges of $T_{d'}$.

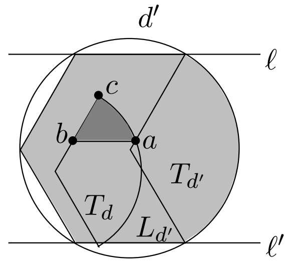

Figure 6: The triangle-like region abc bounded by ab, bc, and $\overline{ac}$. The lightly-shaded region is $L_{d'}$, as in Figure 4.

We next claim that the point a lies above the center of d. Indeed, suppose to the contrary that both intersection points of A_d with $\partial T_{d'}$ (a and the second, lower point a') are on the bottom half of A_d (i.e., on the subarc of A_d starting at the bottom vertex c' of A_d up until the middle of A_d). Since the center of d is lower than the center of d', both intersection points are on the bottom half of $\partial T_{d'}$ (i.e., each lying either on the bottom edge of $T_{d'}$ or on the bottom half of $A_{d'}$). Since the bottom half of A_d is the graph of a monotone increasing function and the bottom edge of $T_{d'}$ is the graph of a monotone decreasing function, the bottom half of A_d and the bottom edge of $T_{d'}$ cross each other at most once. This means that at least one intersection point lies on $A_{d'}$, which, as we have just argued, must be the lower point a' (because a lies on an edge of $T_{d'}$). This however is impossible, because a' lies to the left of a, and thus it lies to the left of $A_{d'}$, contrary to assumption. This contradiction establishes our claim.

To proceed, we consider the possible ways in which A_d can intersect $\partial T_{d'}$ twice (see Figure 7). As proven before, the top intersection point of A_d and $\partial T_{d'}$ lies on one of the edges of $T_{d'}$. It is impossible that both intersection points lie on the top edge of $T_{d'}$, for then the center of d would have to be higher than the center of d', as is easily checked. This leaves us with four subcases: Either (i) both intersection points lie on the bottom edge of $T_{d'}$ (see Figure 7(a)), or (ii) one intersection point lies on $A_{d'}$ and one on the top edge of $T_{d'}$ (see Figure 7(b)), or (iii) one intersection point lies on the bottom edge and one on the top edge of $T_{d'}$ (see Figure 7(c)), or (iv) one intersection point lies on $A_{d'}$ and one on the bottom edge of $T_{d'}$ (see Figure 7(d)).

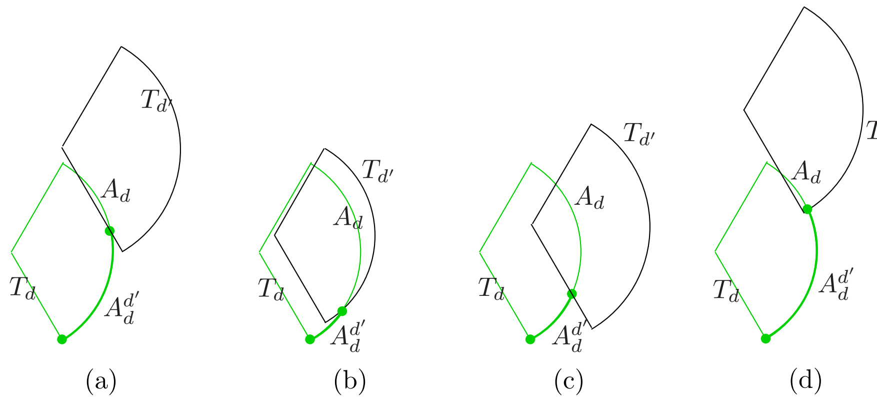

Figure 7: The four situations of case 2; the grey disk is the smaller disk d. (a) A_d crosses the bottom edge of $T_{d'}$ twice. (b) A_d crosses $A_{d'}$ and the top edge of $T_{d'}$. (c) A_d crosses both the top and bottom edges of $T_{d'}$. (d) A_d crosses $A_{d'}$ and the bottom edge of $T_{d'}$.

The proof below deals with all of these situations in the same manner. The proof-related Figures 6 and 8 are shown for situation 7(c), but the proof applies, as is, for the other cases as well. Recall that we have assumed that the center of d is below the center of d' (in the y-direction), so that $A_d^{d'}$ is the bottom part of A_d. As noted before, we only need to prove that the intersection point I of A_d with the ray v belongs to $A_d^{d'}$. Suppose to the contrary that this is not the case. Then either I is in the middle part of A_d or I is in the top part (see Figure 8) . An almost identical proof to the one in case 1 shows that if I is in the middle part of A_d then $q \in d'$, contrary to our assumption. Suppose then that I is in the top part of A_d.

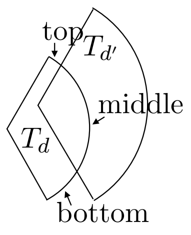

Figure 8: Decomposition of A_d in case 2; $A_d^{d'}$ is the bottom subarc.

Denote by b the point of intersection of ∂T_d and the leftward-directed ray emanating from a; see Figure 6. Since $|ba| \le r_d < r_{d'}$, we get that b lies in $L_{d'}$. By what has been argued above, b lies on the top edge e of T_d. Since e is parallel to the top left edge e^* of $L_{d'}$, starts (at its bottom) below e^*, and is shorter than e^*, it follows that the top endpoint c of e lies in $L_{d'}$. In conclusion, b and c both lie in $L_{d'}$ and outside $T_{d'}$. The top subarc $\overline{ac}$ of A_d is the graph of a monotone decreasing function, both of whose endpoints lie in $L_{d'}$. It then easily follows that the entire arc is contained in $L_{d'}$. Since b lies on the top edge of T_d, the triangle-like region abc, bounded by ab, bc, and the arc $\overline{ac}$, is fully contained in $L_{d'}$. Since q lies in this region by assumption, it follows that $q \in L_{d'} \subset d'$, contrary to our assumption. Hence I must lie in $A_d^{d'}$, as claimed.

This concludes the proof of the lemma, and thus establishes property (3) of M_r. □

We summarize the results of this section in the following theorem.

Theorem 2.2. *Given a set $\mathcal{D}$ of n disks in the plane, one can construct a data structure of linear size, consisting of (point location structures for) the map M_r and its two symmetric counterparts M_t and M_b, constructed respectively over the top parts and over the bottom parts of the disks of $\mathcal{D}$, such that, for a given query point $q \in \mathbb{R}^2$, the largest disk of $\mathcal{D}$ containing q is the largest of the disks that contain q among the three disks d_r, d_t, d_b where d_r is the disk whose arc $A^*_{d_r}$ is the first arc of M_r hit by the rightward directed ray from q, and where d_t, d_b are similarly defined for M_t, M_b, respectively, and for the respective rays in directions $2\pi/3$, $-2\pi/3$. The query time is $O(\log n)$.*

Remark. The reason for dividing each disk into three parts is to ensure that $L_d \subset d$ (and thus also $K_d \subset d$) for each disk d. Both properties fail if T_d is larger than a third of a disk.

2.3 Efficient Construction of M_r

As already noted, the operational definition of the map M_r, as given in Section 2.2, leads to a straightforward and simple $O(n^2)$ algorithm for constructing the map. We now describe a more efficient procedure for constructing M_r, which runs in $O(n \log^3 n)$ time.

Fix a disk $d \in \mathcal{D}$, and let $\mathcal{D}_d$ denote the set of all disks in $\mathcal{D}$ larger than d. Let $\mathcal{T}_d$ denote the collection of the right portions $T_{d'}$ of all the disks $d' \in \mathcal{D}_d$, and let U_d denote their union. Since the elements of $\mathcal{T}_d$ are homothetic copies of one another, their union has linear complexity; see [5, 6].

Before we proceed, we first establish the following lemma. For an arc A_d of some disk d, and a point $p \in A_d$ in the top (resp., bottom) half of A_d, we define the *conjugate* point $\bar{p}$ of p (with respect to A_d) to be the second intersection point of A_d and the vertical line through p; see Figure 9(a).

Lemma 2.3. *Let d, $d' \in \mathcal{D}$ such that d' is larger than d and the center of d' lies above (resp., below) the center of d (in the y-direction). Let $p \in A_d \setminus T_{d'}$ be a point in the top (resp., bottom) half of A_d. Then the conjugate point $\bar{p}$ of p (with respect to A_d) is also in $A_d \setminus T_{d'}$.*

Proof. See Figure 9(b) for an illustration. We may assume, without loss of generality, that the center of d' lies above the center of d; the complementary case can be handled in a fully symmetric manner. In this case p lies in the top half of A_d and $\bar{p}$ lies in the bottom half. Suppose to the contrary that $\bar{p}$ lies in $T_{d'}$. Let l be the vertical line through p and $\bar{p}$. Let u and v denote the two intersection points of l with $\partial T_{d'}$, with u lying above v; these points must exist, for otherwise $\bar{p}$ would trivially lie in $A_d \setminus T_{d'}$. Observe that either u lies on the top edge of $T_{d'}$ and v on the bottom edge, or both points lie on $A_{d'}$. In either case, the midpoint w of uv has the same y-coordinate as the center of d', and therefore must lie above the midpoint p_0 of $p\bar{p}$, whose y-coordinate is equal to that of the center of d. Since p lies outside $T_{d'}$ and $\bar{p}$ lies inside, it follows that u lies between p and $\bar{p}$ and v lies below $\bar{p}$. But then w must lie below p_0, as is easily checked, a contradiction that completes the proof. □

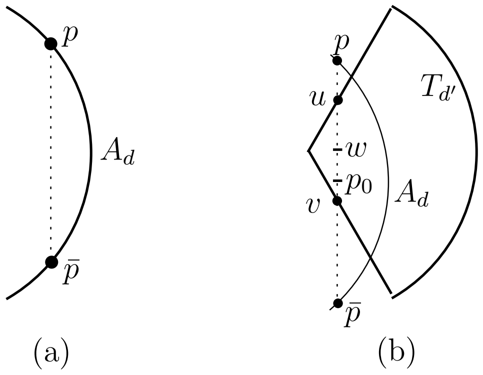

Figure 9: (a) Point $p \in A_d$ and its conjugate point $\bar{p}$. (b) Illustrating the setup in Lemma 2.3.

Let $\mathcal{D}_d^{(a)}$ (resp., $\mathcal{D}_d^{(b)}$) denote the collection of all the disks d' such that d' is larger than d and the center of d' lies above (resp., below) the center of d (in the y-direction); thus $\mathcal{D}_d = \mathcal{D}_d^{(a)} \cup \mathcal{D}_d^{(b)}$. Let $\mathcal{T}_d^{(a)}$ (resp., $\mathcal{T}_d^{(b)}$) denote the collection of the regions $T_{d'}$, for $d' \in \mathcal{D}_d^{(a)}$ (resp., $d' \in \mathcal{D}_d^{(b)}$), and let $U_d^{(a)}$ (resp., $U_d^{(b)}$) denote the union of $\mathcal{T}_d^{(a)}$ (resp., $\mathcal{T}_d^{(b)}$); in particular, $U_d = U_d^{(a)} \cup U_d^{(b)}$. Lemma 2.3 implies the following corollary.

Corollary 2.4. *Let $p \in A_d \setminus U_d^{(a)}$ (resp., $p \in A_d \setminus U_d^{(b)}$) be a point in the top (resp., bottom) half of A_d. Then the conjugate point $\bar{p}$ of p (with respect to A_d) is also in $A_d \setminus U_d^{(a)}$ (resp., $A_d \setminus U_d^{(b)}$).*

Set $A_d^{(a)} := \bigcap_{d' \in \mathcal{D}_d^{(a)}} A_d^{d'}$, and $A_d^{(b)} := \bigcap_{d' \in \mathcal{D}_d^{(b)}} A_d^{d'}$. That is, $A_d^{(a)}$ (resp., $A_d^{(b)}$) is the subarc of A_d resulting from applying the rule for creating A_d^* only to the disks in $\mathcal{D}_d^{(a)}$ (resp., in $\mathcal{D}_d^{(b)}$). By definition, $A_d^* = A_d^{(a)} \cap A_d^{(b)}$. The algorithm uses the following lemma to construct $A_d^{(a)}$ and $A_d^{(b)}$.

Lemma 2.5. *$A_d^{(a)}$ (resp., $A_d^{(b)}$) is the lowest (resp., highest) connected component of $A_d \setminus U_d^{(a)}$ (resp., $A_d \setminus U_d^{(b)}$).*

Proof. Without loss of generality, we prove the lemma only for $A_d^{(a)}$, as the handling of $A_d^{(b)}$ is fully symmetric. For disks $d' \in \mathcal{D}_d^{(a)}$, both rules for constructing $A_d^{d'}$ boil down to the single rule that $A_d^{d'}$ is the lowest connected component of $A_d \setminus T_{d'}$. By applying this rule to all of the disks $d' \in \mathcal{D}_d^{(a)}$, we get that $A_d^{(a)}$ is disjoint from $U_d^{(a)}$, and since all the subarcs

$A_d^{d'}$, for $d' \in \mathcal{D}_d^{(a)}$, are connected, we conclude that $A_d^{(a)}$ is connected and contained in a single connected component of $A_d \setminus U_d^{(a)}$. In particular, if $A_d \setminus U_d^{(a)} = \emptyset$ then $A_d^{(a)}$ is also empty. For each $d' \in \mathcal{D}_d^{(a)}$, each of the endpoints of the subarc $A_d^{d'}$ (if $A_d^{d'} \neq \emptyset$) is either a vertex of A_d or one of the intersection points of A_d and $\partial T_{d'}$. Thus, each endpoint of the subarc $A_d^{(a)}$, which is a finite intersection of such arcs $A_d^{d'}$, is also either a vertex of A_d or one of the intersection points of A_d with some $T_{d'}$. This implies that, if not empty, $A_d^{(a)}$ is a maximal connected component of[5] $A_d \setminus U_d^{(a)}$.

Let L be the lowest connected component of $A_d \setminus U_d^{(a)}$. Clearly, if $L \neq A_d^{(a)}$ then either $A_d^{(a)} = \emptyset$ and $L \neq \emptyset$, or $A_d^{(a)} \neq \emptyset$ but L is a lower connected component of $A_d \setminus U_d^{(a)}$ than $A_d^{(a)}$. Assume first to the contrary that $A_d^{(a)} \neq \emptyset$ but L is a lower connected component of $A_d \setminus U_d^{(a)}$ than $A_d^{(a)}$. Pick points $p \in A_d^{(a)}$ and $p' \in L$, so p' is lower than p. By construction, $p \in A_d^{d'}$ for every disk $d' \in \mathcal{D}_d^{(a)}$, and $p' \in A_d \setminus T_{d'}$. By definition of $A_d^{d'}$, p' must also belong to this arc. Since this holds for all $d' \in \mathcal{D}_d^{(a)}$, $p' \in A_d^{(a)}$, contrary to our assumption. Thus $L = A_d^{(a)}$.

Suppose then, again to the contrary, that $A_d^{(a)} = \emptyset$ and $L \neq \emptyset$ and let p be some arbitrary point in L. Notice that in order for that to happen, p must be in the top connected component of $A_d \setminus T_{d'}$ for some $d' \in \mathcal{D}_d^{(a)}$, i.e., $A_d \setminus T_{d'}$ consists of two connected components and p is in the top one. Indeed, if this does not happen, p, which lies outside all the sets $T_{d'}$, for $d' \in \mathcal{D}_d^{(a)}$, must lie in all the arcs $A_d^{d'}$ and thus also in $A_d^{(a)}$, which is impossible. As shown in the analysis of case 2 in the proof of Lemma 2.1, the top intersection point of A_d and $\partial T_{d'}$ is above the center of d. Hence p must lie in the top half of A_d, which means that L is contained in the top half of A_d. Using Corollary 2.4 we conclude that the conjugate point $\bar{p}$ of p, which lies in the bottom half of A_d, is in $A_d \setminus U_d^{(a)}$ as well, contradicting the fact L is the lowest component of $A_d \setminus U_d^{(a)}$.

In conclusion, we have shown that $L = A_d^{(a)}$ in all cases. This, and a symmetric argument for $A_d^{(b)}$, complete the proof of the lemma. □

Remark. Implicitly, the lemma also asserts that $A_d^{(a)}$ (resp., $A_d^{(b)}$) is empty if and only if $A_d \setminus U_d^{(a)}$ (resp., $A_d \setminus U_d^{(b)}$) is empty.

Since $A_d^* = A_d^{(a)} \cap A_d^{(b)}$, it follows that A_d^* is the intersection of the lowest component of $A_d \setminus U_d^{(a)}$ and the highest component of $A_d \setminus U_d^{(b)}$.

Decomposability. We note that the observations just made are more general in nature, and yield the following *decomposability* property of the construction of A_d^*. Suppose that $\mathcal{D}_d = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \cdots \cup \mathcal{D}_s$. For each $j = 1, \ldots, s$, let $\mathcal{D}_j^{(a)}$ (resp., $\mathcal{D}_j^{(b)}$) denote the set of those disks in $\mathcal{D}_j$ whose centers lie above (resp., below) the center of d, and let $A_{d;j}^{(a)}$ (resp., $A_{d;j}^{(b)}$) denote the lowest (resp., highest) connected component of $A_d \setminus \bigcup \mathcal{D}_j^{(a)}$ (resp., of $A_d \setminus \bigcup \mathcal{D}_j^{(b)}$).

[5] Clearly, if $A_d \setminus U_d^{(a)} = \emptyset$ then $A_d^{(a)} = \emptyset$ as well; the converse implication will shortly be established.

Then $A^*_d = \bigcap_{j=1}^{s} \left(A^{(a)}_{d;j} \cap A^{(b)}_{d;j} \right)$.

A divide-and-conquer algorithm. The preceding analysis suggests the following divide-and-conquer procedure for constructing M_r. Let $\mathcal{D}^+$ (resp., $\mathcal{D}^-$) denote the collection of the $n/2$ larger (resp., smaller) disks of $\mathcal{D}$, and let M_r^+ (resp., M_r^-) denote the map constructed (recursively) on the right portions of the disks in $\mathcal{D}^+$ (resp., $\mathcal{D}^-$).

Note that all the arcs in M_r^+ are arcs of the final map M_r (they are not affected by the addition of the smaller disks), but the arcs of M_r^- might require some trimming to turn them into the correct arcs in M_r, because of the effect of the larger disks in $\mathcal{D}^+$ on them. Let A_d^- be an arc in M_r^-, contributed by some disk $d \in \mathcal{D}^-$. To incorporate the effect that the disks in $\mathcal{D}^+$ have on A_d^-, we compute the lowest (resp., highest) connected arc $A_d^{+(a)}$ (resp., $A_d^{+(b)}$) of $A_d \setminus \bigcup_{d' \in \mathcal{D}^+ \cap \mathcal{D}_d^{(a)}} T_{d'}$ (resp., $A_d \setminus \bigcup_{d' \in \mathcal{D}^+ \cap \mathcal{D}_d^{(b)}} T_{d'}$), and add to the final map M_r the arc $A^*_d := A_d^{+(a)} \cap A_d^{+(b)} \cap A_d^-$. The last identity follows directly from the definition:

$$A^*_d = \bigcap_{d' \in \mathcal{D}_d} A_d^{d'} = \left(\bigcap_{d' \in \mathcal{D}^+ \cap \mathcal{D}_d^{(a)}} A_d^{d'} \right) \cap \left(\bigcap_{d' \in \mathcal{D}^+ \cap \mathcal{D}_d^{(b)}} A_d^{d'} \right) \cap \left(\bigcap_{d' \in \mathcal{D}^- \cap \mathcal{D}_d} A_d^{d'} \right) = A_d^{+(a)} \cap A_d^{+(b)} \cap A_d^- .$$

To complete the description of this divide-and-conquer process, we present an efficient implementation of the construction of the arcs $A_d^{+(a)}$ and $A_d^{+(b)}$ for the right portions of all the disks $d \in \mathcal{D}^-$. In what follows we concentrate only on the efficient construction of the arcs $A_d^{+(a)}$; computing the corresponding arcs $A_d^{+(b)}$ is done in a fully symmetric manner. The arcs A^*_d are then obtained by the preceding rule, and the construction of M_r is completed.

A useful subprocedure. Consider first the following subproblem, which arises as a major step in the construction. We have a set $\mathcal{E}^- = \{d_1^-, \ldots, d_k^-\}$ of k "small" disks, and a set $\mathcal{A}^- = \{A_1, \ldots, A_k\}$ of noncrossing arcs, so that A_j is a subarc of the arc bounding the right portion $T_{d_j^-}$ of d_j^-, for $j = 1, \ldots, k$. We also have a set $\mathcal{E}^+ = \{d_1^+, \ldots, d_s^+\}$ of s "large" disks, so that all the disks of $\mathcal{E}^+$ are larger than all the disks of $\mathcal{E}^-$, and the center of each disk of $\mathcal{E}^+$ lies above the center of every disk of $\mathcal{E}^-$ (in the y-direction). Let U^+ denote the union of all the right portions T_d, for $d \in \mathcal{E}^+$. Our goal, according to Lemma 2.5 and the decomposability property noted following it, is to compute, for each $j = 1, \ldots, k$, the lowest subarc α_j of $A_j \setminus U^+$.

We first note the following operational definition of α_j: Let a be the lower endpoint of A_j. If $a \notin U^+$ then a is also the lower endpoint of α_j, and the upper endpoint of α_j is the lowest intersection point b of A_j with ∂U^+ (or the upper endpoint of A_j if no such intersection exists). Otherwise, if $a \in U^+$, the intersection point b just defined (if it exists) is the lower endpoint of α_j, and the upper endpoint is the second lowest intersection point of A_j with ∂U^+ (or, again, the upper endpoint of A_j if no second intersection exists); if b does not exist then $\alpha_j = \emptyset$. See Figure 10 for an illustration.

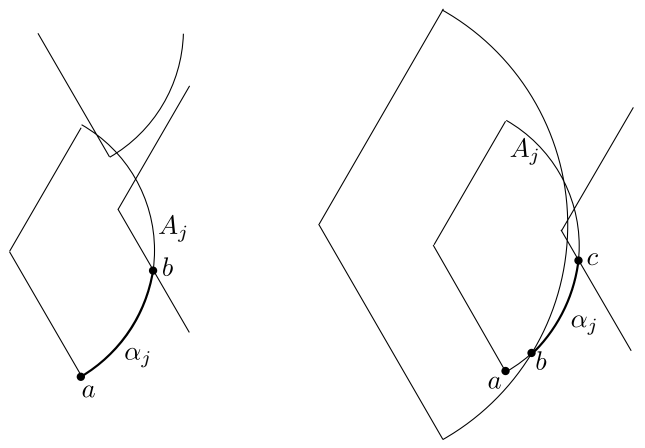

Figure 10: (a) α_j is the arc $\overline{ab}$ when $a \notin U^+$. (b) α_j is the arc $\overline{bc}$ when $a \in U^+$.

We compute the union U^+, and construct on it a standard point location data structure (see, e.g., [3]), which allows us to determine, in logarithmic time, whether a query point lies inside the union. As already noted several times earlier, U^+, as the union of s homothetic regions, has linear complexity (that is, $O(s)$). The cost of constructing U^+ is discussed below, towards the end of the presentation of the algorithm. Let $\mathcal{B}^+$ denote the set of all the $O(s)$ circular arcs and straight segments that form the boundary of U^+.

We then run a horizontal line sweeping algorithm, in increasing y-direction, on the collections $\mathcal{A}^-$ and $\mathcal{B}^+$, whose goal is to compute, for each arc of $\mathcal{A}^-$, its lowest or two lowest intersection points with the elements of $\mathcal{B}^+$. In more detail, for each arc $A_j \in \mathcal{A}^-$, we locate the lower endpoint a_j of A_j in U^+. If $a_j \notin U^+$, it suffices to compute the first lowest intersection point, and otherwise we need to compute the two lowest intersections.

The implementation of the sweep is straightforward; we omit here the bulk of its description, and only give the following few comments about its execution. First, since the arcs of $\mathcal{A}^-$ are pairwise noncrossing, and so are the arcs and segments of $\mathcal{B}^+$, the sweep will only encounter intersections between the elements of $\mathcal{A}^-$ and those of $\mathcal{B}^+$. For each arc A_j of $\mathcal{A}^-$, the sweep extracts from the event priority queue (the "y-structure") the intersection points of A_j with the elements of $\mathcal{B}^+$ in increasing y-order. As soon as it extracts the first or second such intersection, as appropriate for A_j, it discards A_j, removing the arc itself from the "x-structure" (the balanced search tree that represents the arcs crossed by the sweepline), and removing any future events involving A_j from the queue. In this manner, the algorithm processes only $O(k + s)$ events, at a total cost of $O((k + s)\log(k + s))$ time.

Back to the divide-and-conquer construction. Returning to the original divide-and-conquer procedure, we would like to apply the sweeping technique to the arcs of M_r^- (playing the role of $\mathcal{A}^-$) and to the union of the right portions T_d of the disks $d \in \mathcal{D}^+$ (playing the role of U^+). However, since we do not have control over the relative positions of the centers of the two corresponding families of disks, we need a secondary process to control the locations of the centers.

Specifically, we sort the disks of $\mathcal{D}^+$ in increasing y-order of their centers, and store them in this order at the leaves of a balanced binary tree Q. For each node $v \in Q$, we consider the subset $\mathcal{D}_v$ of the disks stored at the leaves of the subtree rooted at v. Note that $\sum_v |\mathcal{D}_v| = O(n \log n)$. Given a disk $d \in \mathcal{D}^-$, we can obtain, by searching in Q with the center of d, the subset $\mathcal{D}^+ \cap \mathcal{D}_d^{(a)}$ of disks of $\mathcal{D}^+$ whose centers lie above that of d, as the (disjoint) union of $O(\log n)$ subsets $\mathcal{D}_v$, and the decomposability propery implies that it suffices to solve the problem for each of them separately, and form the intersection of the $O(\log n)$ resulting subarcs of A_d, to obtain the desired subarc $A_d^{+(a)}$, as defined above. As a result of the searches in Q with the centers of the disks of $\mathcal{D}^-$, each node $v \in Q$ stores a set S_v of all the disks $d \in \mathcal{D}^-$ that use v as part of their decomposition (i.e., the decomposition of $\mathcal{D}^+ \cap \mathcal{D}_d^{(a)}$). The overall size of the sets S_v, as well as the time to construct them, is $O(n \log n)$.

Now, for each node $v \in Q$ we have a pair $(S_v, \mathcal{D}_v)$ of subsets $S_v \subseteq \mathcal{D}^-$ and $\mathcal{D}_v \subseteq \mathcal{D}^+$. We recover from M_r^- the arcs that correspond to the disks in S_v, and run the sweeping algorithm described above, where the recovered arcs play the role of $\mathcal{A}^-$, and where $\mathcal{D}_v$ plays the role of $\mathcal{E}^+$. After all the sweeps are performed, we have computed, for each disk $d \in \mathcal{D}^-$, $O(\log n)$ arcs $A_d^{(1)}, A_d^{(2)}, \ldots$. As argued above, the intersection of all these arcs is the desired $A_d^{+(a)}$. We then run this procedure again, essentially reversing the direction of the y-axis, to obtain the subarcs $A_d^{+(b)}$, for $d \in \mathcal{D}^-$, and add $A_d^* = A_d^{+(a)} \cap A_d^{+(b)} \cap A_d^-$ to the output map M_r.

The total time of all the sweeps is bounded by

$$O\left(\sum_{v\in Q}(|\mathcal{D}_v| + |S_v|)\log(|\mathcal{D}_v| + |S_v|)\right) = O\left(\sum_{v\in Q}(|\mathcal{D}_v| + |S_v|)\log n\right) = O(n \log^2 n).$$

To complete the presentation, we next discuss the cost of constructing the union $U_v^+ = \bigcup \{T_d \mid d \in \mathcal{D}_v\}$, for all the nodes v of Q. We construct these unions in a bottom-up manner, from the leaves of Q towards its root. The construction of the leaves is trivial, because at each leaf the union involves a single region T_d. Let v be a non-leaf node of Q with children w,z. Then $U_v^+ = U_w^+ \cup U_z^+$. if v has s leaves in its subtree, then each of U_w^+, U_z^+, U_v^+ has $O(s)$ complexity. This means that U_v^+ can be constructed by a straightforward line sweeping procedure over the overlay of U_w^+ and U_z^+, in time $O(s \log s)$. Adding up the costs, over all nodes v of Q, all the unions U_w^+ can be constructed in a total of $O(n \log^2 n)$ time.

In conclusion, we have presented a procedure that, given two maps M_r^- and M_r^+, constructed respectively over the $n/2$ smaller disks of $\mathcal{D}$ and over the $n/2$ larger disks, merges them into the final map M_r, in $O(n \log^2 n)$ time. Denoting by $T(n)$ the maximum time for the algorithm to run on a set of n disks, we obtain the recurrence relation: $T(n) = 2T(\frac{n}{2}) + O(n \log^2 n)$, and thus $T(n) = O(n \log^3 n)$.

Remark. As described, the preprocessing algorithm uses $O(n \log n)$ space (for maintaining all the sets S_v, $\mathcal{D}_v$). To impove the storage requirement to linear, we construct Q, and the corresponding sets S_v, $\mathcal{D}_v$, in an incremental bottom-up manner, maintaining at each step the sets S_v, $\mathcal{D}_v$ only within a single level of Q. The information concerning the sets $\mathcal{D}_v$ and the unions of their disks is easy to transfer from each level to the next one up. To obtain the sets S_v within a level the simplest way is to search in Q with the centers of disks of $\mathcal{D}^-$ from scratch, only until the desired level. This implementation takes only linear space.

Putting everything together, and summarizing the statement of Theorem 2.2, we obtain the following summary result of this paper.

Theorem 2.6. *Let $\mathcal{D}$ be a set of n disks in the plane. One can preprocess $\mathcal{D}$ into a data structure of linear size, in time $O(n \log^3 n)$, so that, for any query point q we can report the largest disk of $\mathcal{D}$ that contains q or determine that there is no such disk, in $O(\log n)$ time.*

Remark. Although this is somewhat marginal, it would be interesting to reduce the cost of the preprocessing.

Acknowledgements. The authors wish to thank Haim Kaplan for helpful discussions and feedback on the problem studied in this paper. We also thank indirectly Joe Mitchell and Günter Rote, whose discussions of this problem with Haim were a prime motivation for us to study this problem.

References

[1] J. Augustine, S. Das, A. Maheshwari, S. C. Nandy, S. Roy, and S. Sarvattomananda, Recognizing the largest empty circle and axis-parallel rectangle in a desired location, in arXiv:1004.0558, 2010.

[2] J. Augustine, S. Das, A. Maheshwari, S. C. Nandy, S. Roy, and S. Sarvattomananda, Querying for the largest empty geometric object in a desired location, in arXiv:1004.0558v2, 2010.

[3] M. de Berg, O. Cheong, M. van Kreveld and M. Overmars, *Computational Geometry: Algorithms and Applications*, 3rd Edition, Springer Verlag, Berlin–Heidelberg, 2008.

[4] H. Kaplan and M. Sharir, Finding the maximal empty disk containing a query point, *Int. J. Comput. Geom. Appl.*, 23 (2013), 335–355. Also in *Proc. 28th Annu. ACM Sympos. Comput. Geom.*, 2012, pp. 287–292

[5] K. Kedem, R. Livne, J. Pach, and M. Sharir, On the union of Jordan regions and collision-free translational morion amdist polygonal obstacles, *Discrete Comput. Geom.* 1 (1986), 59–71.

[6] P. K. Agarwal, J. Pach and M. Sharir, State of the union (of geometric objects), in *Proc. Joint Summer Research Conf. on Discrete and Computational Geometry: 20 Years Later*, Contemp. Math. 452, AMS, 2008, pp. 9–48.

[7] C. K. Yap, Geometric consistency theorem for a symbolic perturbation scheme, *J. Computer and System Sciences* 40(1) (1990), 2–18.