International Journal of Computational Geometry & Applications
Vol. 9, No. 2 (1999) 109–123

COMPUTING A DOUBLE-RAY CENTER FOR A PLANAR POINT SET*

ALEX GLOZMAN KLARA KEDEM†

and

GREGORY SHPITALNIK

Department of Mathematics and Computer Science
Ben-Gurion University of the Negev, POBox 653, Beer-Sheva 84105, Israel
†*E-mail: kedem@CS.Cornell.EDU*

Received 22 October 1996
Revised 22 October 1998
Communicated by J. S. B. Mitchell

ABSTRACT

A double-ray configuration is a configuration in the plane consisting of two rays emanating from one point. Given a set S of n points in the plane, we want to find a double-ray configuration that minimizes the Hausdorff distance from S to this configuration. We call this problem the double-ray center problem. We present an efficient algorithm for computing the double-ray center for set S of n points in the plane which runs in time $O(n^3\alpha(n)\log^2 n)$.

Keywords: Algorithm, optimization, center, planar point set, Hausdorff distance.

1. Introduction

The double-ray center problem extends a list of the following well researched center problems:

- The point center[16]: Find the center (and the radius) of the smallest disk enclosing a set of points S.
- The two center problem[17,8]: Find centers of two disks, whose maximum radius is minimal, which covers the set S.
- The line center problem[15]: Given a set of points S find the line that minimizes the distance between the points and the line.

*Work by K. Kedem has been supported by a grant from the U.S.-Israeli Binational Science Foundation.

- The two-line center problem[13,14,9,12]: Find two strips, whose maximal width is minimized, that cover the set of points in the plane.

These problems are solved in time $O(n)$ for one disk.[16] For two disks a deterministic algorithm runs in time $O(n \log^9 n)$ (Ref. [17]), improved by a randomized algorithm to $O(n \log^2 n)$ time.[8] The line center algorithm runs in time $O(n \log n)$ (Ref. [15]), and the two line center in $O(n^2 \log^4 n)$ time,[13,14,9] this time bound has been recently improved to $O(n^2 \log^2 n)$ (Ref. [12]). In a very recent work[3] a set of points is approximated to be contained in the two sides of wedge of given apex angle. For a given apex angle θ they find the wedge with thinnest sides in time $O(n^2 \log n)$ time.

The double-ray center problem is defined as follows. Let S be a set of n points in the plane. We want to find a configuration, $\mathcal{C} = (O, r_1, r_2)$, consisting of a point O in the plane and two rays, r_1 and r_2, emanating from O, such that the Hausdorff distance from S to $\mathcal{C}$ is minimized (see Figure 1). The Hausdorff distance from S

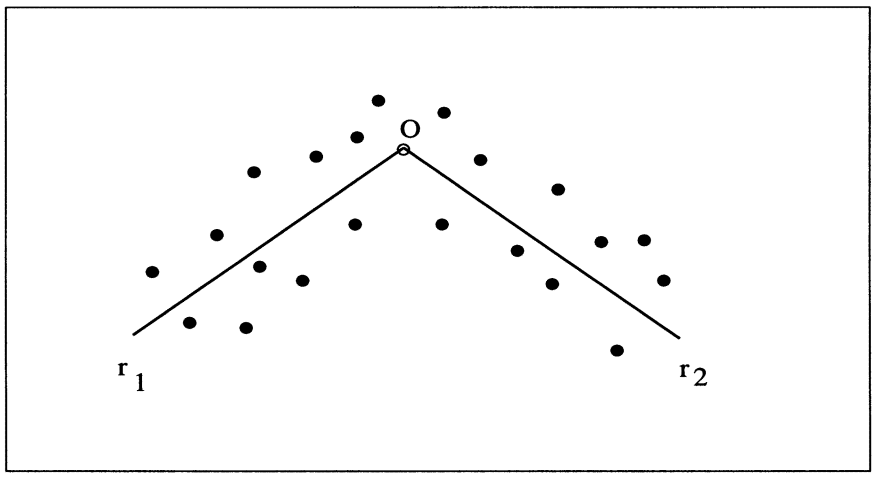

Fig. 1. The double-ray center problem

to $\mathcal{C}$ is defined by

$$h(S, \mathcal{C}) = \max_{s \in S} \min[dist(s, r_1), dist(s, r2)]$$

where $dist(s, r)$ denotes the Euclidean distance between the point s and the ray r. (The distance between a point s and a ray r starting at point O is defined to be the distance between s and the line l through r if the perpendicular line to l through s intersects r, it is the distance between O and s otherwise.)

The double-ray center problem is to find an *optimal configuration* $\mathcal{C}_{opt}$ which minimizes $h(S, \mathcal{C})$. The minimum Hausdorff distance is thus defined by

$$\rho = h(S, \mathcal{C}_{opt}) = \min_{\mathcal{C}} h(S, \mathcal{C}).$$

In this paper we present an algorithm that finds a double-ray center of S. This algorithm can be applied, *e.g.*, to approximating a point set by two rays. The thinnest

wedge problem mentioned above,[3] is applied to metrology problems. Namely approximating measured points by lines of a wedge with a given apex angle. Our algorithm can give a better approximation to such a set of points, (a) because it measures the Euclidean distance from the apex of the wedge (the circular part in our wedge), and (b) our algorithm determines the apex angle that minimizes the widths of the sides of the wedge.

Our algorithm finds the location of a point O and the orientations of the two rays, r_1 and r_2, emanating from O. We also find the minimum Hausdorff distance. We note that the double-ray center is not necessarily unique (see Figure 2). But it can be easily made unique if we define an additional criterion of optimization, for example, a condition on the angle between the rays r_1 and r_2.

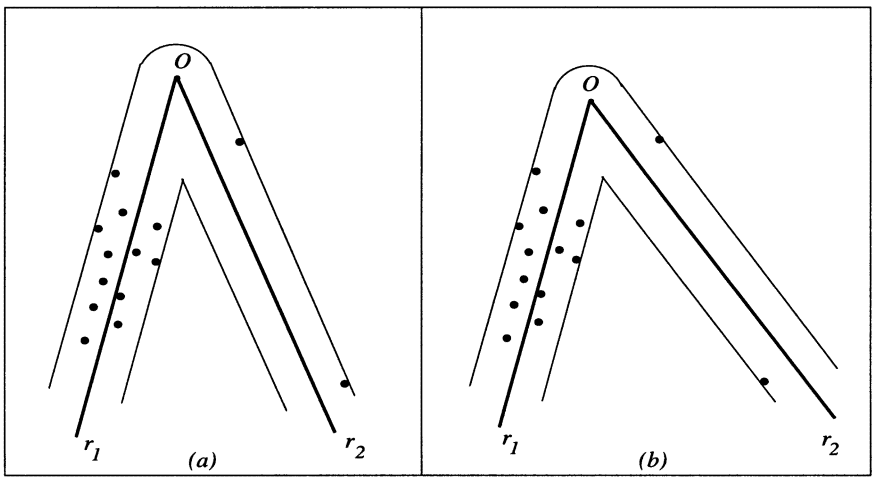

Fig. 2. The non-uniqueness of the double-ray center

We could find the optimal distance ρ by performing a binary search over the sequence $\mathcal{D}$ of all possible values of Hausdorff distance d. We show below that the cardinality of $\mathcal{D}$ is $O(n^4)$. In order to produce an efficient algorithm we cannot afford to construct the set $\mathcal{D}$ explicitly. Instead we apply an implicit searching technique for locating ρ by using the parametric search technique of Megiddo.[16] This technique requires efficient sequential and parallel decision algorithms for the problem.

The decision problem that we answer is the following: given a distance d, is there a double-ray configuration $\mathcal{C}$ such that $h(S, \mathcal{C}) \leq d$? We show that if there is a double-ray configuration $\mathcal{C}$ of distance d, then there is a *special* double-ray configuration $\mathcal{C}'$ with distance not greater than d. Thus, by checking all the possible special double-ray configurations we find the answer for the decision problem. The sequential version of the decision algorithm runs in time $O(n^3\alpha(n))$ and the parallel version runs in $O(\log n)$ time using $O(n^3)$ processors. Applying parametric search yields an $O(n^3\alpha(n)\log^2 n)$ time algorithm for the double-ray center problem.

We show in Section 2 that a given double-ray configuration, with distance d, can be transformed into a special double-ray configuration with distance not greater

than d. In Section 3 we present efficient sequential and parallel algorithms to answer the decision problem for a given d, and then apply parametric search to find an optimal double-ray configuration.

2. Special Double-Ray Configurations

Let $\mathcal{C} = (O, r_1, r_2)$ be a double-ray configuration with distance d for a set $S = \{s_1, \ldots, s_n\}$ of points in the plane. Then the set of points S is contained in the Minkowski sum of $\mathcal{C}$ and a disk U_d of radius d. Denote it by $H = \mathcal{C} \oplus U_d$. Denote $H_1 = r_1 \oplus U_d$ and $H_2 = r_2 \oplus U_d$ (see Figure 3). As will be seen below, throughout our transformations we do not need to worry about points in $H_1 \cap H_2$. We define the set A to be the points of S which are contained in $H_A = H_1 - H_2$, and similarly the set B of all the points of S which are in $H_B = H_2 - H_1$.

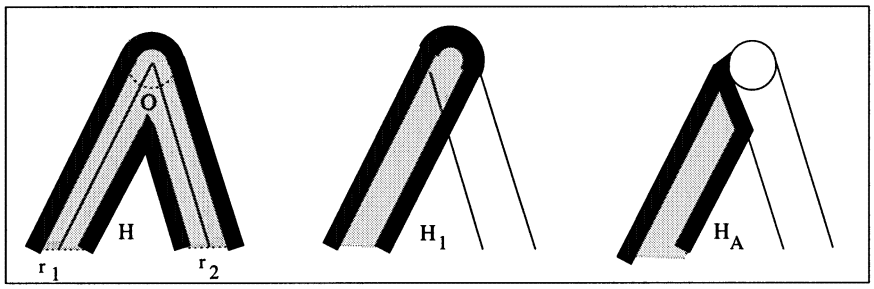

Fig. 3. Illustration of the definitions of H, H_1 and H_A (shaded)

The next theorem allows us to modify a given double-ray configuration $\mathcal{C}$ with distance d to a special double-ray configuration with distance d. We begin by presenting two lemmas for convex polygons whose proofs are straightforward. We term *diagonals* of a convex polygon segments that connect pairs of vertices of the polygon (including the sides of the polygon).

Lemma 1 *Let P be a convex polygon in the plane and let O be a point in $\Re^2 - P$. Then there is a unique line r, passing through O such that the two supporting lines of P, parallel to r, are equidistant from r, and the line r passes either through the midpoint of exactly two diagonals of P – if r is parallel to a side of P, or through the midpoint of one diagonal of P – otherwise.*

We call the line r from the lemma above a *center diagonal* line with respect to O and P, and will discuss in what follows the *center diagonal ray* which emanates from O towards the midpoint(s) of the diagonal(s) of P. We define the *distance* from a convex polygon P to a ray r to be the largest Euclidean distance between the vertices of P and the ray r.

Lemma 2 *Let P be a convex polygon and let O be a point in $\Re^2 - P$. Let r be a ray emanating from O. Then there is a direction (clockwise or anti-clockwise) in which r can be rotated about O such until r is the unique center diagonal ray through O with respect to P, and throughout this rotation the distance between r and P decreases monotonically.*

Theorem 1 *Let C be a double-ray configuration $C = (O, r_1, r_2)$, with distance $d = h(S, C)$. Then, either*

1. *S is of width not greater than $4d$, or*
2. *S can be covered by two parallel strips of width $2d$ each, say (L_1, L_2) and (L_3, L_4), such that both L_1 and L_4 pass through a point of $CH(S)$, and*
   (a) *Either L_1 or L_4 passes through an edge of $CH(S)$, or,*
   (b) *One of the strips has points of S on both of its boundaries, or*
3. *There exists a configuration $C' = (O', r_1', r_2')$, with distance d, such that there is a point of S on each of the four infinite rays of $H = C' \oplus U_d$.*

Proof. The double-ray configuration $C = (O, r_1, r_2)$ defines subsets A and B of S as described above. We start the sequence of transformations by choosing arbitrarily A or B. Assume without loss of generality that we chose A.

The *first transformation* that C undergoes is to rotate the ray r_1 about O such that it is center diagonal with respect to A. By Lemma 2, the set A is not further from the new ray r_1', and the maximum distance from the points of $CH(A)$ to the left of r_1' and r_1' is equal to the maximum distance from the points of $CH(A)$ to the right of r_1' and r_1' (see Figures 4(1) and 4(2), in the latter the distance become d'). For simplicity of notation we rename r_1' by r_1.

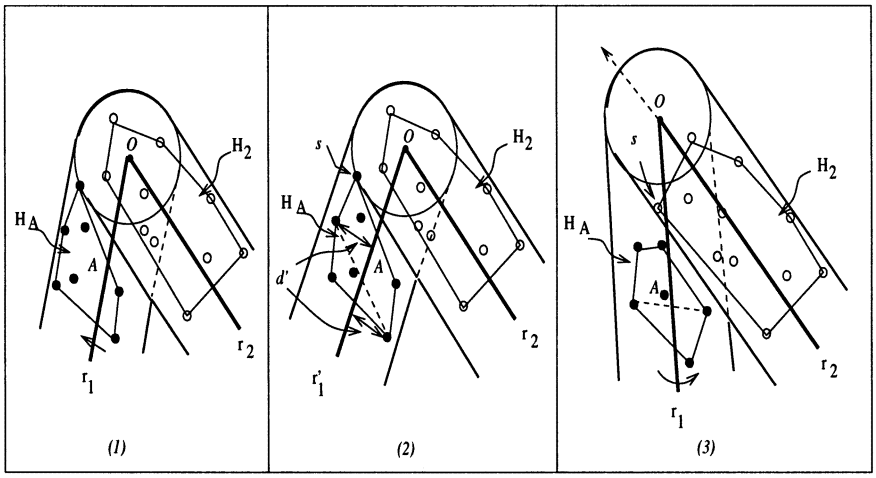

Fig. 4. Transformations.

For the *second transformation*, we pull O along the line containing r_2 and in an opposite direction (we denote it by direction $-r_2$), keeping the ray r_1 to be center diagonal to $CH(A)$, aiming for two points of A to be on the infinite sides of H_A. Notice that pulling O in this direction does not affect the points that were in H_2 initially, since they remain in the transformed H_2. However, we need to rotate r_1 continuously for it to remain center diagonal. During this transformation three events might happen.

(i) A point $s \in A$ might get covered by the moving H_2. So we update A to be $A - s$ and proceed in pulling O in direction $-r_2$ (see Figure 4(3)), and keeping r_1 center diagonal with respect to the new A.
(ii) O reaches infinity, consequently r_1 and r_2 become parallel. We switch to dealing with parallel strips below.
(iii) The distance from A to r_1 is d. The new ray r_1 is center diagonal with respect to A, hence there is a point of $CH(A)$ on each of the infinite rays of H_A. We switch to the next transformation.

Now we perform the two transformations above again, but this time swapping the roles of the rays r_1 and r_2. At the end of these transformations we either have two parallel rays, or we have two center diagonal rays with respect to A and B, such that there is a point of S on each infinite side of H.

We show now that we can transform parallel strips to special parallel strips. If S is covered by two parallel infinite strips, (L_1, L_2) and (L_3, L_4), of width $2d$ each, as defined above, we first move the strips towards each other (translate the strips along a direction perpendicular to them), without changing their widths, until L_1 and L_4 support S. If the two strips overlap at the end of this translation then we are done. Since then the width of S is not greater than $4d$.

If this is not the case then we start rotating the strips, parallel to each other, keeping their widths to be $2d$, and maintaining contact of L_1 and L_4 with $CH(S)$, until, either the strips overlap (then we are done), or either L_1 or L_4 is tangent to a side of $CH(S)$, or either L_2 or L_3 touches a point of S. □

We call the special configurations in Theorem 1, configurations of type (1), (2a), (2b) or (3) (see Figure 5).

Corollary 1 *The number of candidate distances in $\mathcal{D}$ is $O(n^4)$.*

Proof. There are $O(n^4)$ possibilities to pick quadruples of points that determine special configurations of type (3) as defined in Theorem 1. □

3. The Algorithms

The algorithm we discuss first solves the double-ray decision problem by checking the existence of a special configuration of a given distance d. If a special configuration is found then the answer to the decision problem is *yes*, otherwise the answer is *no*. Since finding special configurations of types (1) and (2) are quite trivial we bind them in Algorithm A, and describe in more detail Algorithm B, which finds special configurations of type (3).

3.1. Algorithm A

1. Find if S can be covered by a strip of width not greater than $4d$.
2. For each side e of $CH(S)$, align a supporting line L_1 with e, and find the antipodal supporting line for $CH(S)$, L_4. These lines and d determine uniquely the strips (L_1, L_2) and (L_3, L_4) of width $2d$ each. Check whether S is covered by these strips.

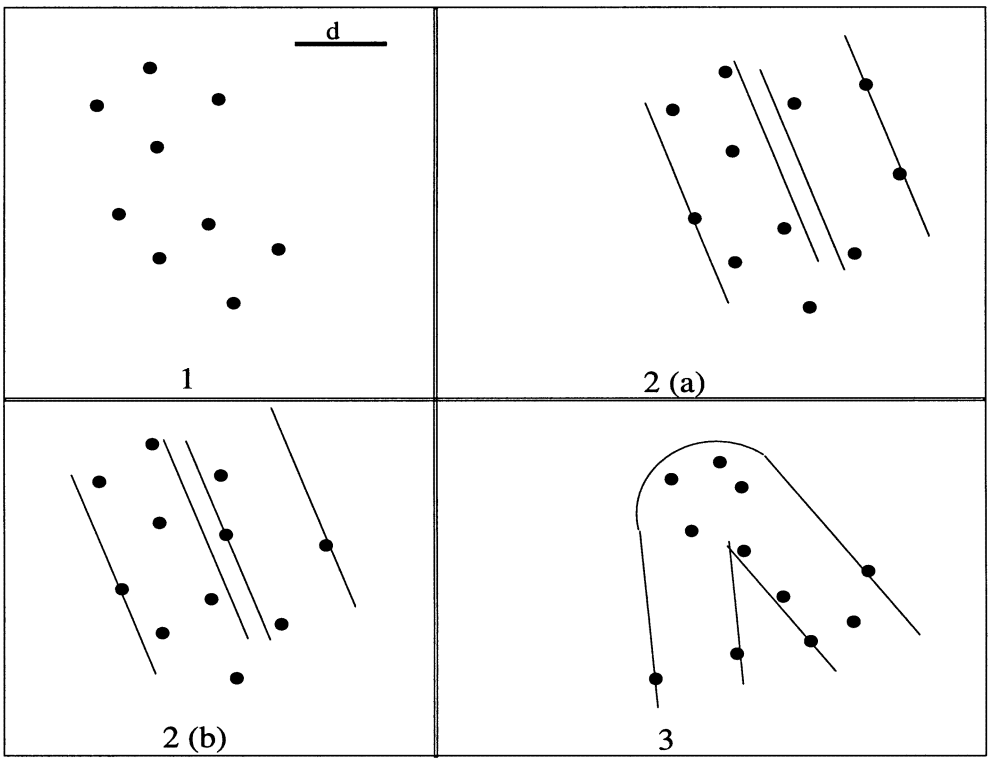

Fig. 5. The special configurations, as in Theorem 1.

3. For each pair of points (p_1, p_2), where $p_1, p_2 \in S$, and p_1 is on $CH(S)$, find the at most two directions such that the strip defined by L_1 and L_2 passing through these points respectively is of width $2d$ and L_1 is a supporting line of $CH(S)$ at p_1. Find the supporting line to $CH(S)$, L_4, which is parallel to L_1. These lines and d determine uniquely the line L_3. Check whether S is covered by these strips.

The complexity of Steps 1-3 in the algorithm: Since algorithm B is more complex, we can afford the most trivial approach for these three steps. Step 1 can be computed in time $O(n \log n)$, Step 2 in time $O(n^2)$ and Step 3 in time $O(n^3)$. Note that with some more care these steps can be performed in $O(n^2 \log n)$ time.

3.2. Preliminaries for Algorithm B

We now want to find if there is a double-ray configuration with distance d and with a point of S on each infinite edge of H.

Consider the special configuration $\mathcal{C} = (O, r_1, r_2)$ where four points of S, $p_1, \ldots, p_4$ are placed, one on each of the infinite rays, $b_1, \ldots, b_4$ of H, respectively (case 3 in Theorem 1). Assume without loss of generality that the ray r_1 is horizontal (see Figure 6), and that b_1 and b_2 are parallel to r_1, and b_3 and b_4 are parallel to r_2. Denote the lines passing through $b_1, \ldots, b_4$, respectively, by $L_1, \ldots, L_4$. We denote by α the angle between the rays r_1 and r_2. We denote the circular arc of H by μ and the points where the arc μ meets the rays b_1 and b_4 by q and t respectively. Let

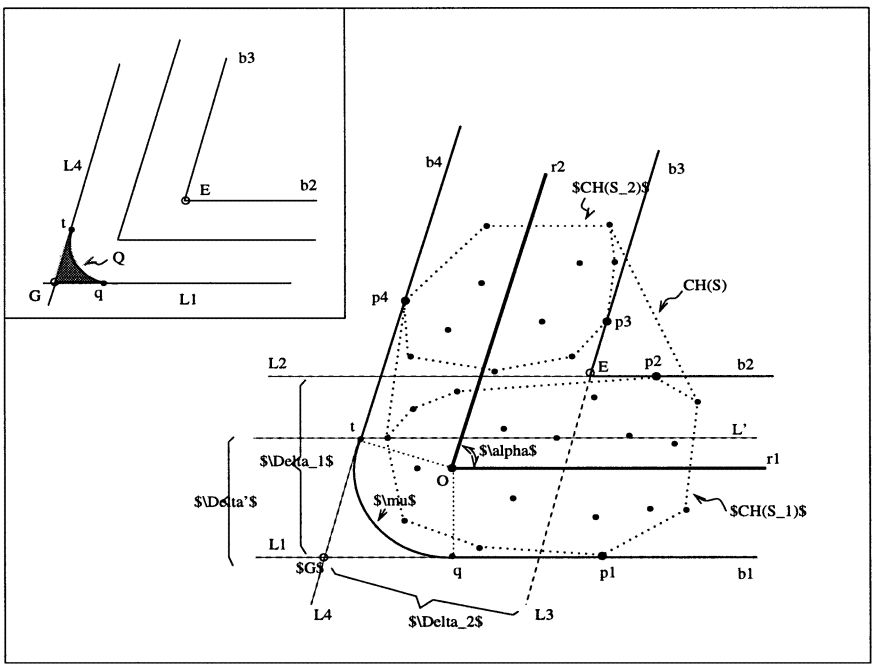

Fig. 6. The third special configuration.

L' be a horizontal line passing through the point t. Let G be the intersection point of L_1 and L_4, and E be the intersection point of L_2 and L_3. Let Q be a triangular shape with one circular edge consisting of the line segments $[G,t]$, $[G,q]$ and the arc μ. Let $I = H \cup Q$. The lines L_1 and L' form a horizontal strip Δ'. We denote the set of points in Δ' by S'.

The general idea behind this algorithm is to start with aligning a supporting line L_1 to $CH(S)$. This line defines a line L_2 which intersects $CH(S)$, is parallel to L_1, and is in distance $2d$ from L_1. Now the pair of lines is rotated about $CH(S)$, maintaining distance of $2d$ between them, such that L_1 always supports $CH(S)$. A *global event* occurs when L_2 meets a point of S. Then the pair L_1, L_2 defines a strip Δ_1 that contains a subset S_1 of the points in S where two points of S_1 are on the boundaries of the strip. For this global event consider the set of points $S_2 = S - S_1$. We find another strip, Δ_2, delimited by the parallel lines L_3 and L_4, by rotating L_4 about $CH(S)$ and rotating L_3 in parallel, so that it supports $CH(S_2)$, until the distance between L_3 and L_4 becomes $2d$. This is called a *local event*, and the configuration is a *candidate* double-ray configuration. Note that $I = H \cup Q$, as defined above (see Figure 6), contains all the points of S. But we need to check whether Q, the triangular shape with one circular edge contains points of S or not. If it does not contain points of S then we have a double-ray configuration of distance d. If it does, we proceed to the next local event.

The following preliminaries will be needed for fast implementation of Algorithm B: Let $\mathcal{A}(d)$ be the arrangement induced by n circles of radius d in the plane. $\mathcal{A}(d)$ partitions the plane into $O(n^2)$ faces with $O(n^2)$ vertices and (circular) edges.[5,7] The zone of a line L in $\mathcal{A}(d)$ is the region obtained by the union of all the cells of $\mathcal{A}(d)$ which are crossed by L, its complexity is $O(n\alpha(n))$.[7] The vertical decomposition $\mathcal{A}^*(d)$ of $\mathcal{A}(d)$ is a refinement of $\mathcal{A}(d)$ obtained by extending two vertical segments up and down from each vertex of $\mathcal{A}(d)$ and from the two points of vertical tangency of each of the circles, until they meet the boundary of a neighboring circle. The combinatorial complexity of the graph $\mathcal{A}^*(d)$ and a zone in $\mathcal{A}^*(d)$ are bounded by $O(n^2)$ and $O(n\alpha(n))$ respectively.

Consider a horizontal strip Δ of width $2d$ and the semi-circles $\mathcal{H}^l$ and $\mathcal{H}^r$ which are the left and right halves of a circle of radius d cut by its vertical diameter. Let the boundary lines of the strip Δ be the tangents to the semi-circle $\mathcal{H}^l$ in its uppermost and bottommost points respectively, and let $\mathcal{H}^l$ move from left to right inside Δ. Assume further that there is a subset S' of points of S inside Δ. We define a full order $\aleph^l$ on the set S' as follows: $\forall p_1, p_2 \in S'$, $p_1 \leq_{\aleph^l} p_2$ if the semi-circle $\mathcal{H}^l$ meets p_1 before it meets p_2 when moving from left to right inside the strip Δ. We associate with each point $p \in S'$ the placement of the bottommost point of the semi-circle $\mathcal{H}^l$ when it meets p during its motion inside the strip Δ. We define $\aleph^r$ symmetrically.

Lemma 3 *Given the placement of the strips Δ_1 and Δ_2 as defined above they define the points q and t and the strip Δ'. Assume without loss of generality that the orientation of the ray r_1 coincides with the x-axis, then there is a configuration $\mathcal{C} = (O, r_1, r_2)$, of distance d, corresponding to Δ_1 and Δ_2 if and only if $q \leq_{\aleph^l} p$ for*

all points p of S in Δ'.

Proof. If $q \leq_{\aleph^l} p$ for all points p of S in Δ' then a left half circle $\mathcal{H}^l$ meets q before it meets any other point in Δ', thus there are no points in Q and there is a configuration $\mathcal{C}(r_1, r_2, d)$ corresponding to Δ_1 and Δ_2. Conversely, if there is a configuration $\mathcal{C}$, then Q is empty and thus $q \leq_{\aleph^l} p$ for all points p of S in Δ'. □

We denote by $\mathcal{L}$ the list of lines passing through pairs of points in S sorted by their slopes. Clearly, there are $n(n-1)/2$ such lines. We provide each line $l_i \in \mathcal{L}, (i = 1 \ldots \frac{n(n-1)}{2})$ with another line $\bar{l}_i \perp l_i$. We assign an array $\mathcal{M}_i$, of the points of S, to l_i. The points in $\mathcal{M}_i$ are ordered from left to right according to their orthogonal projections on $\bar{l}_i$ (their distances from l_i).

Observation 1 *(Ref. [6]) Given two consecutive lines $l_i, l_{i+1} \in \mathcal{L}$ and the array $\mathcal{M}_i$, then the array $\mathcal{M}_{i+1}$ can be obtained from $\mathcal{M}_i$ by a single swap of points $p', p'' \in \mathcal{M}_i$, where p' and p'' is the pair of points which defines the line l_{i+1}.*

We associate with each line $l_i \in \mathcal{L}$ its slope, the indices of the corresponding points p' and p'' as above, and the pointer to the corresponding array $\mathcal{M}_i$. We call this structure the *permutation diagram* of the set S.

Algorithm B below follows the general approach of Ref. [11]. Namely, it passes over all the possible placements of the first strip Δ_1, and for each such placement it looks for a second strip Δ_2 such that together they form a double-ray configuration $\mathcal{C} = (O, r_1, r_2)$ of type (3).

Algorithm B.

1. Construct the permutation diagram of S.
2. Construct the arrangement $\mathcal{A}(d)$ of circles of radius d centered at the points of S. Construct the vertical decomposition $\mathcal{A}^*(d)$ of $\mathcal{A}(d)$.
3. Perform a circular sweep of strip Δ_1 of fixed width $2d$, such that its boundary line L_1 is rotated about $CH(S)$ until L_1 returns to its starting position. Stop whenever the second boundary line L_2 meets a point of S (global event).

   For each global event do:

   (a) Define the orders $\aleph^l$ and $\aleph^r$ on the set S_1 of points of S within Δ_1, along the ray r_1 using the arrangement $\mathcal{A}^*(d)$. Call the orientation of the strip Δ_1 the horizontal orientation.

   (b) Obtain the sorted order of points of the set S_1 along the y-axis using the permutation diagram of S.

   (c) Update $CH(S_2)$ – the convex hull of the points of S out of Δ_1.

   (d) Perform an anti-clockwise (and later clockwise) rotational sweep of two parallel lines L_3 and L_4, such that L_3 is rotated about $CH(S_2)$ and L_4 about $CH(S)$, starting from the placement, where L_3, L_4 are parallel to L_1, until L_4 coincides again with L_1. Stop whenever either (i) the distance between the lines L_3 and L_4 becomes equal to $2d$ (L_3 and L_4

form the strip Δ_2), or (ii) the horizontal line L' which is determined by the current placement of L_4 meets a point $p' \in S_1$. These are local events of types (i) and (ii).

According to the type of the local event do:

(i) Check whether the configuration $\mathcal{C}$ formed by the current strips Δ_1 and Δ_2 satisfies $h(S,\mathcal{C}) \leq d$, by testing whether there is a point p of S' inside the region Q. If it does not contain any point of S' then stop, else proceed to the next local event.

(ii) Delete the point p' from the set S' and update the order $\aleph^l$ ($\aleph^r$) defined on S'. Proceed to the next local event.

3.3. Analysis of Algorithm B

Steps 1 and 2 are preprocessing steps of algorithm B. The permutation diagram of S can be constructed in $(n^2 \log n)$ time. We construct the permutation diagram only once, since it does not depend on the value of the parameter d. The arrangements $\mathcal{A}(d)$ can be constructed in $O(n^2)$ time and then be vertically decomposed into $\mathcal{A}^*(d)$ in $O(n^2)$ time.[2,5] Step 3 is repeated $O(n^2)$ times since each pair of points of S defines at most two strips of width $2d$ (global events). In order to evaluate the runtime of Step 3 we need the following:

Lemma 4 *Given a ray r_1 in the positive direction of the x-axis, the associated strip Δ_1 and the arrangement $\mathcal{A}(d)$ as above, then the order $\aleph^l$ defined on the set of points S_1 is the same order in which the ray r_1 meets the right semi-circles of the arrangement $\mathcal{A}(d)$.*

Proof. The ray r_1 intersects only the circles of $\mathcal{A}(d)$ whose centers lie inside the strip Δ_1. Consider the semi-circle $\mathcal{H}^l$ when it moves leftwards (from $-\infty$) inside the strip Δ_1. During this motion the center O' of $\mathcal{H}^l$ slides along the ray r_1. Every time the semi-circle $\mathcal{H}^l$ meets a point $p \in S_1$, there is a right semi-circle $\mathcal{H}^r$ centered at p which intersects the ray r_1 in the point O'. □

According to Lemma 4, we can obtain the order $\aleph^l$ (or $\aleph^r$) on the set of points S_1 by obtaining the order in which the ray r_1 meets the boundaries of the circles of $\mathcal{A}(d)$. We traverse the zone of a line L passing through the ray r_1 in the arrangement $\mathcal{A}^*(d)$. The arrangement $\mathcal{A}^*(d)$ has the property that each cell of $\mathcal{A}^*(d)$ is of constant complexity. Thus, traversing $\mathcal{A}^*(d)$ can be done in time proportional to the number of cells in zone of L in $\mathcal{A}^*(d)$, namely $O(n\alpha(n))$. It implies that Step 3(a) runs in $O(n\alpha(n))$ time. Step 3(b) can be done in $O(n^3)$ overall time: as we rotate L_1 about $CH(S)$ we stop whenever the slope of L_1 coincides with a slope $l_j \in \mathcal{L}$, update $\mathcal{M}_j$ (by one swap) until we reach a global event. Assume we reached a global event right after slope l_i. Then we retrieve the sorted order of the points of S_1 along the y-axis in $O(n)$ time from the corresponding array $\mathcal{M}_i$.

Step 3(c) can be done in $O(n)$ time by retrieving the order along the y-axis of all points of S which are out of Δ_1 from the array $\mathcal{M}_i$ and then applying an $O(n)$ algorithm to compute the convex hull of these points.[10]

Lemma 5 *Given the ray r_1 in the positive direction of the x-axis and the strip Δ_1 associated with it, then the y-coordinate of the point t , as defined above, is a monotone decreasing function of the angle α between the fixed orientation of the strip Δ_1 and the orientation of strip Δ_2.*

Proof. Consider the lines L_1 with fixed horizontal orientation and the line L_4 rotated about $CH(S_2)$. Clearly, the y-coordinate of the point t is expressed by $y = d\cos\alpha + d$, where $\alpha \in [0, \pi]$. The value of y is a decreasing function of the angle α, since $\cos\alpha$ is a decreasing function of the angle α for $\alpha \in [0, \pi]$. □

Since a convex polygon with n vertices has $O(n)$ antipodal pairs of vertices, it is clear that there are only $O(n)$ antipodal pairs when L_4 supports $CH(S)$ and L_3 supports $CH(S_2)$, therefor there are $O(n)$ local events of type (i) in Step 3(d). Moreover, since the sides of $CH(S_2)$ are ordered by their slopes, we can represent the sequence of these events as a discrete function of the increasing angle $\alpha \in [0, \pi]$ in $O(n)$ time. Lemma 5 implies that during Step 3(d) the line L' slides downwards only. Thus there are $O(n)$ local events of type (ii). The information about the order of the points of S_1 obtained in Step 3(d) permits us to represent all local events of type (ii) as a discrete decreasing function of the increasing angle $\alpha \in [0, \pi]$ in $O(n)$ time. Thus we can correctly order the sequence of local events of both types by merging their corresponding discrete functions in $O(n)$ time. By Lemma 3 the feasibility test of the configuration $\mathcal{C}$ in each local event of type (i) can be reduced to finding an extremal point in the set S' in the order $\aleph^l$ (or $\aleph^r$) defined on S'. When the algorithm starts processing the local events in Step 3(d) the set S' coincides with the set S_1 which already has its points ordered along the y-axis and also in the order $\aleph^l$ and $\aleph^r$. During Step 3(d) the horizontal line L' slides down towards the line L_1. On each event of type (ii) the algorithm deletes a point p' from the set S' updating the order $\aleph^l$ (or $\aleph^l$) in S'. Thus Step 3(d) consists of the maintenance of the set S' under the given set of $O(n)$ deletions and $O(n)$ extremal point queries, which can be done in $O(n)$ time using cross-indexed doubly linked lists.

It follows from the analysis of algorithm B that the complexity of Step 3 is dominated by Step 3(a), which consumes $O(n\alpha(n))$ time. Thus in total, the runtime required by algorithm B is $O(n^3\alpha(n))$.

Theorem 2 *Given a set S of n points in the plane, the decision algorithm for the double-ray center problem runs in $O(n^3\alpha(n))$ time.*

3.4. The Parallel Version of the Decision Algorithm

In this section we show that there exists a parallel version of the decision algorithm, which uses $O(n^3)$ processors and runs in time $O(\log n)$.

The parallelization of algorithm A is easy, so we shall concentrate on algorithm B. In order to produce an efficient parallel algorithm we need to avoid the incremental calculations of the sequential algorithm B, namely, step 3(d).

Lemma 6 *Given a set S of n points in the plane with two different full orders $\aleph_1$ and $\aleph_2$ defined on S, and the points of S sorted in order $\aleph_1$. Then, after $O(n)$ preprocessing, we can answer the following query in $O(\log n)$ time: for any two*

points $p', p'' \in S$ *such that* $p' \leq_{\aleph_1} p''$, *find the extremal points in order* $\aleph_2$ *among all the points* $p \in S$ *such that* $p' \leq_{\aleph_1} p \leq_{\aleph_1} p''$.

Proof. We construct a balanced binary search tree T. The leaves of the tree T store the points of the set S in order $\aleph_1$. Each internal node γ stores the two extremal points among the points at the leaves of subtree of T rooted in γ, in the order $\aleph_2$. Using the indices of points p' and p'' in order $\aleph_1$ we traverse the two paths π' and π'' from the root of the tree T to the leaves corresponding to the points p' and p''. Obviously, all the points of S stored in the leaves of T between these leaves satisfy $p' \leq_{\aleph_1} p \leq_{\aleph_1} p''$. Since T is a balanced tree there are $O(\log n)$ disjoint subtrees of T (rooted at the siblings of the nodes lying along the paths π' and π'') that together cover all the leaves of T between the leaves corresponding to p' and p''. We visit now the roots of these subtrees and choose the two extremal points in order $\aleph_2$. We can construct T in $O(n)$ sequential time and the query can be answered in $O(\log n)$ time. □

Corollary 2 *Given* $O(n)$ *processors one can construct the tree* T *defined above in* $O(\log n)$ *parallel steps.*

Proof. We construct the tree T in a bottom-up manner. Each level of T can be constructed from the previous level in $O(1)$ parallel time with the help of $O(n)$ processors. Since the tree T is balanced there are $O(\log n)$ levels in the T. □

The parallel version B_p of Algorithm B.

1. Assign a set P of $O(n)$ processors to each of the $O(n^2)$ possible placements of the strip Δ_1 (corresponding to each global event of algorithm B).
2. Each set of processors P performs the following operations:
   (a) Partition the points of S into subsets S_1 and S_2, where S_1 is the set of points inside the strip Δ_1 and $S_2 = S - S_1$.
   (b) Associate the orientation of the strip Δ_1 with the positive direction of the x-axis. Sort the points of S_1 along the y-axis.
   (c) Construct the tree T on the set S_1 as described in Lemma 6 such that the order of the points of S_1 along the y-axis stands for the order $\aleph_1$ and the order $\aleph^l$ (or $\aleph^r$) stands for the order $\aleph_2$.
   (d) Construct $CH(S_2)$.
   (e) Assign each processor $\eta \in P$ to an antipodal pair $(p, q) \in CH(S_2)$. Each processor $\eta \in P$ performs the following operations:

   (i) Check whether there is a strip of the width $2d$ defined by the antipodal pair (p,q) associated with the processor η (there can be at most two such strips). If there is no such strip then stop, otherwise proceed to the next step (the obtained strip is a candidate to be the strip Δ_2).

   (ii) Check whether the configuration $\mathcal{C}$ formed by the current strips Δ_1 and Δ_2 satisfies $h(S, \mathcal{C}) \leq d$, by testing whether there is a point $p \in S'$ inside the region Q. (This test is done efficiently using the tree T).

3.5. Analysis of Algorithm B_p

Step 1 of the algorithm B_p is a straightforward processor allocation step. We assign each processor $\eta \in P$ to a point $p \in S$. Then Step 2(a) can be done in $O(1)$ parallel time, since each processor η needs to check whether a point p associated with η is in the strip Δ_1 or not. Clearly, Step 2(b) is done in constant time. Step 2(c) requires $O(\log n)$ parallel time by Corollary 2. The convex hull of the set S_2 can be constructed in $O(\log n)$ time with the help of $k = O(|S_2|)$ processors.[1,4]

Each antipodal pair $(p, q) \in CH(S_2)$ invokes at most two strips of width $2d$. Thus, in Step 2(e)(i), the strips of width $2d$ defined by pair $(p, q) \in CH(S_2)$ can be found analytically in constant time by a single processor $\eta \in P$. Given the placements of the strips Δ_1 and Δ_2 each processor $\eta \in P$ can perform the test of Step 2(e)(ii) in $O(\log n)$ time. This follows from Lemma 3 and Lemma 6. Algorithm B_p performs a constant number of a steps (2(a) – 2(e)), each requires at most $O(\log n)$ time per step, hence we have proved the following theorem:

Theorem 3 *Given a set S of n points in the plane the algorithm for the double-ray decision problem runs in $O(\log n)$ parallel time using $O(n^3)$ processors.*

Finally applying the parametric search paradigm,[16] we obtain the following result:

Theorem 4 *Given a set S of n points in the plane the double-ray center of S can be computed in $O(n^3\alpha(n)\log^2 n)$ time.*

References

1. M.J. Atallah and M.T. Goodrich, " Efficient parallel solutions to some geometric problems", *J. Parallel and Distributed Computing*, **3**(1986) 492–507.
2. P. K. Agarwal and M. Sharir, "Planar geometric location problem and maintaining the width of a planar set", *Algorithmica*, **11**(1994) 185–195.
3. M. de Berg, H. Meijer, M. Overmars and G. Wilfong, "Computing the angularity tolerance", *Proc. 8th Canad. Conf. Comput. Geom.*, 1994 331–336.
4. R. Cole and M. T. Goodrich, "Optimal parallel algorithms for polygon and point set problems", *Algorithmica*, **7**(1992) 3–23.
5. B. Chazelle and D.T. Lee, "On a circle placement problem", *Computing*, **1**(1986) 1–16.
6. H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, (Springer-Verlag, Heidelberg, Germany 1987).
7. H. Edelsbrunner, L. Guibas, J. Pach, R. Pollack, R. Seidel, M. Sharir, "Arrangements of curves in the plane – topology, combinatorics and algorithms", *Theoretical Computer Science*, **92**(1992) 319–336.
8. D. Eppstein, "Faster construction of planar two-centers", *8th ACM-SIAM Symp. Discrete Algorithms*, New Orleans (1997) 131–138.
9. A. Glozman, K. Kedem and G. Shpitalnik, "On some geometric selection and optimization problems via sorted matrices", *Fourth Workshop on Algorithms and Data Structures*, S.G. Akl, F. Dehne, J. Sack and N. Santoro, editors, Lecture Notes in Computer Science 955(1995), Springer-Verlag 26–35.
10. R.L. Graham and F.F. Yao, "Finding the convex hull of a simple polygon", *J. Algorithms*, **4**(1983) 324–331.

11. J. Hershberger and S. Suri. "Finding tailored partitions", *J. Algorithms*, **12** (1991) 431–463.

12. J.W. Jaromczyk and M. Kowaluk, "The two-line center problem from a polar view: a new algorithm and data structure", *Fourth Workshop on Algorithms and Data Structures*, S.G. Akl, F. Dehne, J. Sack and N. Santoro, editors, Lecture Notes in Computer Science 955(1995), Springer-Verlag 13–25.

13. M.J. Katz, "Improved algorithms in geometric optimization via expanders", *Proc. 3rd Israel Symposium on Theory of Computing and Systems*, (1995) 78–87.

14. M.J. Katz and M. Sharir, "An expander-based approach to geometric optimization", *SIAM J. of Comput.*, **26**(1997) 1384-1408.

15. D.T. Lee and Y. Wu, "Geometric complexity of some location problems", *Algorithmica*, **1**(1986) 193–211.

16. N. Megiddo, "Linear-time algorithms for linear programming in R^3 and related problems", *SIAM J.of Comput.*, **12**(1983) 759–776.

17. M. Sharir, "A near-linear algorithm for the planar 2-center problem", *Discrete Comput. Geom.* **18**(1997) 125–134.