

# Ray-Shooting Depth: Computing Statistical Data Depth of Point Sets in the Plane

Nabil H. Mustafa<sup>1</sup> \*, Saurabh Ray<sup>2</sup> \*\*, and Mudassir Shabbir<sup>3</sup>

<sup>1</sup> Dept. of Computer Science, LUMS, Pakistan. nabil@lums.edu.pk

<sup>2</sup> Max-Planck-Institut für Informatik, Saarbrücken, Germany.

saurabh@mpi-inf.mpg.de

<sup>3</sup> Dept. of Computer Science, Rutgers, USA. mudassir@cs.rutgers.edu

**Abstract.** Over the past several decades, many combinatorial measures have been devised for capturing the statistical data depth of a set of  $n$  points in  $\mathbb{R}^2$ . These include Tukey depth [15], Oja depth [12], Simplicial depth [10] and several others. Recently Fox *et al.* [7] have defined the Ray-Shooting depth of a point set, and given a topological proof for the existence of points with high Ray-Shooting depth in  $\mathbb{R}^2$ . In this paper, we present an  $O(n^2 \log^2 n)$ -time algorithm for computing a point of high Ray-Shooting depth. We also present a linear time approximation algorithm.

## 1 Introduction

The area of statistical data analysis deals with the following kind of question: given some multivariate data in  $\mathbb{R}^d$ , possibly with noise, what one point in  $\mathbb{R}^d$  best describes that data. There are several considerations to take into account when quantitatively formulating the measure that accurately captures the notion of ‘best’: robustness to noise, computational ease, invariance under translation and rotation, invariance to change-of-axis and so on. (see e.g. [13,3,1,2,10,11,7,16,12]).

A robust measure can be constructed by generalizing the concept of the *median* of numbers to points in  $\mathbb{R}^d$ . Given a set  $P$  of  $n$  points in  $\mathbb{R}^d$ , define the Tukey depth [15] of a point  $q \in \mathbb{R}^d$  as the smallest-number of points contained in any halfspace containing  $q$ . The classical Centerpoint theorem states that given any set  $P$  of  $n$  points in  $\mathbb{R}^d$ , there exists a point with Tukey depth at least  $n/(d+1)$ ; furthermore, this is the best possible. Finally, the Tukey depth of a point set  $P$  is defined as the maximum Tukey depth of any point in  $\mathbb{R}^d$ . There have been a sequence of papers related to finding efficient algorithms for computing a point with high Tukey depth, and a point realizing the Tukey

---

\* This work was done on a one-year visit to the wonderful DCG group at EPFL, Switzerland.

\*\* The first and second authors gratefully acknowledge support from the Bernoulli Center at EPFL and from the Swiss National Science Foundation, Grant No. 200021-125287/1.

depth of a given point set. In  $\mathbb{R}^2$ , the naive algorithm to find the point with the highest Tukey depth takes  $O(n^6)$  time, which was improved by Rouseaw and Ruts [14] to  $O(n^2 \log n)$ . This was further improved by Matousek to  $O(n \log^5 n)$ . Finally Chan [4] gave an  $O(n \log n)$  time randomized algorithm. A deterministic  $O(n \log^3 n)$  time algorithm was given by Langerman and Steiger [9]. A point with Tukey depth at least  $n/3$  can be computed in  $O(n)$  time [8]. In  $\mathbb{R}^d$ , the current-best algorithms for both finding a point of depth  $n/(d+1)$  and the highest depth point take  $O(n^{d-1})$  time [4].

Recently an elegant new measure, which we call Ray-Shooting depth, has been proposed in [7]. Given a set  $P$  of  $n$  points, let  $E$  be the set of all  $\binom{n}{d}$   $(d-1)$ -simplices spanned by  $P$ . Then define the Ray-Shooting depth (called RS depth from now on) of a point  $q$  as the minimum number of simplices in  $E$  that any ray from  $q$  must intersect. While the problem of existence of a point with high RS depth is open in  $\mathbb{R}^d$ ,  $d \geq 3$ , it is shown in [7] that given any  $P$  in  $\mathbb{R}^2$ , there exists a point with RS depth at least  $n^2/9$ . An easy construction shows that this is optimal. As the status of the problem is not known even for  $\mathbb{R}^3$ , from now onwards, we only consider the  $d = 2$  case in this paper.

Unfortunately the proof given in [7] is completely existential, as it follows from a variant of Brouwer’s fixed point theorem. A straightforward algorithm can be derived from it with running time  $O(n^5 \log^5 n)$  by an exhaustive search. The main focus of this paper will be to present faster algorithms, both exact and approximate, for computing RS depth of a planar point set. Admittedly, one would like an algorithm for computing points of high RS depth in higher dimensions as well. However, it is not clear how to extend the current topological machinery that we use for the  $d = 2$  case to higher dimensions.

*Our Contributions.* We show that one can compute a point of RS depth  $n^2/9$  in the plane in  $O(n^2 \log^2 n)$  time. One can also compute an approximation to such a point in linear-time. Specifically, one can compute a point with RS depth at least  $n^2/9 - o(n^2)$  in linear-time. These two results are presented in Section 2. We have also implemented the approximation algorithm, and made it available as a package in the statistical computing software R; this is explained in Section 4.

## 2 Computing a point of Ray-Shooting depth at least $n^2/9$

In this section, we present an algorithm that takes as input a finite set  $P$  of  $n$  points in the plane and returns a point  $p$  in the plane whose Ray-Shooting depth is at least  $n^2/9$ . Given a point  $q \in \mathbb{R}^2$  and a vector  $u \in \mathbb{S}^1$ , denote by  $\rho_{q,u}$  the closed ray emanating from  $q$  in the direction  $u$ . For any  $p \in \mathbb{R}^2$ ,  $p \neq q$ , denote the closed ray emanating from  $q$  and passing through  $p$  as  $\rho_{q,p}$ . We will denote the unit vector in the direction of  $\rho_{q,p}$  as  $\delta_{q,p}$ . A ray  $r$  is *bad* if it intersects fewer than  $n^2/9$  edges (segments) spanned by the input points, and *good* otherwise. The following lemma is from [7].

**Lemma 1.** *For any  $q \in \mathbb{R}^2$  and for any two bad rays  $\rho_1$  and  $\rho_2$  emanating from  $q$ , one of the two closed cones defined by  $\rho_1$  and  $\rho_2$  contains fewer than  $n/3$  input points.*

*Proof.* If both cones defined by  $\rho_1$  and  $\rho_2$  have at least  $n/3$  points, the rays  $\rho_1$  and  $\rho_2$  together intersect at least  $2n^2/9$  edges. This is a contradiction to the assumption that both of them are bad rays.  $\square$

Using the above lemma, we prove the following.

**Lemma 2.** *For any  $q \in \mathbb{R}^2$ , there exists a cone  $C_q$  with apex  $q$  containing more than  $2n/3$  input points, and where all the rays emanating from  $q$  and contained in  $C_q$  are good.*

*Proof.* If there are no bad rays emanating from  $q$  then we take  $C_q$  as the entire plane. Otherwise, let  $\rho$  be a bad ray emanating from  $q$ . Let us pick a ray  $\rho'$  emanating from  $q$  such that both the closed cones defined by  $\rho$  and  $\rho'$  contain at least  $n/2$  input points. Note that  $\rho'$  is a good ray (Lemma 1).

Imagine starting with the ray  $\rho'$ , and rotating it in either direction about  $q$  as long as it is good. See Figure 2. Let  $\rho_1$  (respectively  $\rho_2$ ) be the last good ray before we hit a bad ray, in the counter-clockwise (resp. clockwise) direction. Clearly  $\rho_1$  and  $\rho_2$  pass through input points. Let  $\rho'_1$  be a bad ray close-enough to  $\rho_1$  so that there are no points of  $P$  between these rays. Similarly let  $\rho'_2$  be a close-enough bad ray to  $\rho_2$ .

The cone  $C_1$  defined by  $\rho$  and  $\rho'_1$  (Figure 2) contains fewer than  $n/3$  input points (by Lemma 1 applied to  $\rho$  and  $\rho'_1$ , as the cone complement to  $C_1$  contains at least  $n/2$  points). Similarly, the cone  $C_2$  contains fewer than  $n/3$  points. Consider now the bad rays  $\rho'_1$  and  $\rho'_2$ . Let  $C_3$  be the cone defined by these rays that contains the ray  $\rho'$ .  $C_3$  cannot contain fewer than  $n/3$  points since  $C_1 \cup C_2 \cup C_3$  covers  $\mathbb{R}^2$  and  $C_1$  and  $C_2$  contain fewer than  $n/3$  points each. Hence the other cone defined by  $\rho'_1$  and  $\rho'_2$  contains fewer than  $n/3$ . Therefore, the cone  $C_q$  defined by  $\rho_1$  and  $\rho_2$  that contains  $\rho'$  contains more than  $2n/3$  points. By construction, all rays in  $C_q$  are good.  $\square$

For any point  $q \in \mathbb{R}^2$ , define the good cone at  $q$  to be the closed cone  $C$  with apex  $q$  containing the maximum number of input points such that all rays emanating from  $q$  and lying in  $C$  are good. Clearly, the cone  $C_q$  obtained in the proof of the previous lemma is the unique good cone at the point  $q$  since it contains more than  $2n/3 > n/2$  points and is maximal. For any  $u \in \mathbb{S}^1$ , if  $\rho_{q,u} \in C_q$ , we say that  $u$  is a *good direction* at  $q$ . Then the cone  $C_q$  defines the set  $D_q$  of good directions at  $q$ . We will call the set of input points lying in  $C_q$  the *good set* of  $q$  and denote it by  $\mathcal{G}_q$ .

For computational purposes, we will need a data structure which we now describe. Let  $X$  be a set of  $n$  distinct fixed points on a circle  $S$  which we will

refer to as *locations*. Let  $x_1, x_2, \dots, x_n$  be the circular order of these locations (the precise coordinates of these points does not matter). Let  $T = \{t_1, \dots, t_n\}$  be the set of  $n$  open intervals on  $S$  defined by consecutive locations in  $X$ . Let  $Y = \{Y_1, \dots, Y_n\}$  be a set of  $n$  points, each placed at one of the locations in  $X$ . For any points  $u, v \in S$ , we will denote by  $A_{u,v}$  the arc going from  $u$  to  $v$  counter-clockwise along  $S$  (so  $A_{u,v}$  is different from  $A_{v,u}$ ). Our data-structure, denoted by  $\Psi$ , will store  $Y$  at the locations in  $X$  and a number of arcs defined by points of  $Y$ .

Define the *depth* of any point  $q \in S$  as the number of arcs in  $\Psi$  containing  $q$ . Since each point in any of the open intervals  $t_i$  has the same depth, we can define the depth of  $t_i$  as the depth of any point in it. We need  $\Psi$  to support the following operations:

1. Insert an arc  $A_{u,v}$ ,  $u, v \in Y$  into  $\Psi$
2. Delete an arc  $A_{u,v}$ ,  $u, v \in Y$  from  $\Psi$
3. Move a point  $y \in Y$  to a neighboring location.
4. Given a query arc  $A_{u,v}$ ,  $u, v \in S$  and an integer  $k$ , report the first and last intervals, if any, on  $A_{u,v}$  with depth smaller than  $k$ .
5. Given an integer  $k$  report an interval, if any, with depth smaller than  $k$ .

Notice that the endpoints of the arcs we add or delete are in  $Y$ . When we move a point  $y \in Y$  to a neighboring location, the endpoints of the arcs incident to  $y$  move with it. However, in the fourth operation listed above, the endpoints of the query arc are arbitrary points in  $S$ . Using standard data-structuring techniques in computational geometry (augmented interval trees), it is possible to build the data structure in  $O(n \log n)$  time such that each of the operations take  $O(\log n)$  time. We skip the easy details. Let us see how we can use this data structure to compute the good cone  $C_z$  at any point  $z \in \mathbb{R}^2$ .

**Lemma 3.** *The good cone of any point  $z \in \mathbb{R}^2$  can be computed in  $O(n^2 \log n)$  time.*

*Proof.* We take a unit circle and fix  $X$  to be any  $n$  points  $x_1, \dots, x_n$ , in that order around  $S$ . We put a point  $y_i$  at the location  $x_i$  for  $i \in [1, n]$  as follows: angularly sort the input points around the point  $z$  and for every input point  $p_i$  put a representative point  $y_i$  in the location  $x_r$  where  $r$  is the rank of  $p_i$  in the sorted order.

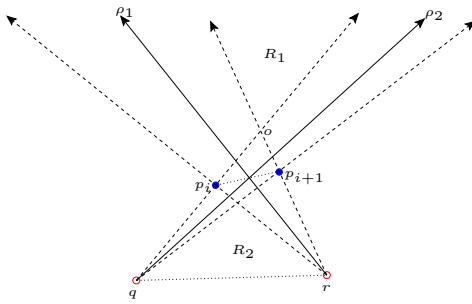
For each pair of input points  $p_i$  and  $p_j$ , if the line  $l_{i,j}$  through  $p_i$  and  $p_j$  does not pass through  $z$ , we insert either the arc  $A_{y_i, y_j}$  or  $A_{y_j, y_i}$  depending on whether  $z$  is to the left or right of the line  $l_{i,j}$  oriented in the direction from  $p_i$  to  $p_j$ . If  $l_{i,j}$  passes through  $z$  we either add both or none of the arcs depending on whether or not  $z$  lies on the edge  $p_i p_j$ . We then query the data structure to see if there are any intervals of depth smaller than  $n^2/9$ . If there are no such intervals, all rays emanating from  $z$  are good. If not, the data structure returns an interval  $I$  with depth less than  $n^2/9$ .

From any point on this interval, we obtain a corresponding bad ray  $r_b$  emanating from  $z$ . As in the proof of Lemma 2, we pick a good ray  $r_g$  such that each of the closed cones defined by  $r_b$  and  $r_g$  contain at least  $n/2$  points. As before, one can find the good cone at  $z$  by rotating  $r_g$  to either side and stopping at the last good ray before we hit bad rays. We can find the first bad interval (interval of depth smaller than  $n^2/9$ ) in either direction by using queries of type 4. Since we insert  $O(n^2)$  arcs and each insertion takes  $O(\log n)$  time, the time taken for inserting the arcs is  $O(n^2 \log n)$ . Once we have inserted all the arcs, it takes only a constant number of queries, each taking  $O(\log n)$  time, to determine the good cone. The overall running time is therefore  $O(n^2 \log n)$ .  $\square$

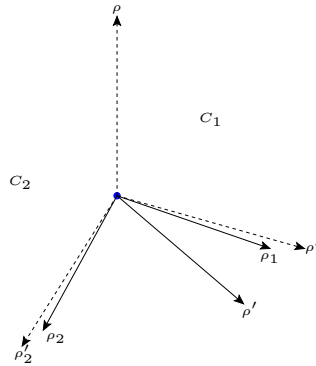
This is not the best algorithm for computing the good cone at a point. The good cone can be computed in  $O(n \log n)$  time but we will not need it for this paper since this is not the bottleneck for the main algorithm.

Let  $\mathcal{L}$  be the set of the  $\binom{n}{2}$  lines passing through each pair of input points. Let  $\mathcal{A}$  be the arrangement of the lines in  $\mathcal{L}$ .

**Lemma 4.** *If  $q, r \in \mathbb{R}^2$  lie in the interior of the same cell of  $\mathcal{A}$  then  $\mathcal{G}_q = \mathcal{G}_r$ . If  $q$  lies in the interior of a cell and  $r$  lies on the boundary of the same cell,  $\mathcal{G}_q \subseteq \mathcal{G}_r$ .*



**Fig. 1.** The good set is the same for all points in a cell.



**Fig. 2.** The good rays are solid, bad rays are dashed.

*Proof.* Let  $q$  and  $r$  be points in the interior of the same cell of  $\mathcal{A}$ . The angular order of the input points is the same around both points. Let  $p_1, \dots, p_k$  be the points in the good set of  $q$  in the angular order. Any ray in the good cone of  $q$  intersects some edge  $p_i p_{i+1}$  formed by two consecutive points in the good set of  $q$ . We will show that for each such edge, any ray emanating from  $r$  and intersecting that edge is also good. This will show that all points in the good set of  $q$  are also

in the good set of  $r$  i.e.  $\mathcal{G}_q \subseteq \mathcal{G}_r$ . The same argument with the roles of  $q$  and  $r$  exchanged shows that  $\mathcal{G}_r \subseteq \mathcal{G}_q$ , implying that  $\mathcal{G}_q = \mathcal{G}_r$ .

Let  $\rho_1$  be a ray emanating from  $r$  and intersecting the edge  $p_i p_{i+1}$ . Let  $\rho_2$  be a ray emanating from  $q$  and intersecting the edge  $p_i p_{i+1}$  at the same point as  $\rho_1$ . Since  $p_i$  and  $p_{i+1}$  are in the good set of  $q$ , we know that the  $\rho_2$  is good. We will show that  $\rho_1$  intersects all the edges that  $\rho_2$  intersects. Let  $C_1$  be the cone defined by  $\rho_1$  and  $\rho_2$  containing the region  $R_1$  and let  $C_2$  be the cone defined by them containing the region  $R_2$  (see Figure 1). If there is an edge that intersects,  $\rho_2$  but not  $\rho_1$  there must be an input point in at least one of these cones. Let  $A_q$  (respectively  $A_r$ ) be the cone with apex  $q$  (resp.  $r$ ), containing  $p_i p_{i+1}$  and bounded by rays through  $p_i$  and  $p_{i+1}$ . Since  $p_i$  and  $p_{i+1}$  are consecutive points in the angular order around the points  $q$  and  $r$ , there are no input points in the cones  $A_q$  and  $A_r$ . Therefore, if there are any input points in the cones  $C_1$  or  $C_2$ , they must lie in the regions  $R_1$  or  $R_2$ . However if there is any input point  $p$  in one of these regions then  $pp_i$  intersects the segment  $qr$  contradicting the assumption that  $q$  and  $r$  lie in the interior of the same cell. The same argument goes through if  $q$  lies in the interior of a cell and  $r$  lies on the boundary of a cell, except if  $r$  lies on the line through  $p_i$  and  $p_j$ . In this case, one can still show that  $\mathcal{G}_q \subseteq \mathcal{G}_r$  but  $\rho_1$  may intersect edges incident to  $p_i$  or  $p_{i+1}$ .  $\square$

In the following, to any continuous non-self-intersecting curve  $\omega$  with distinct endpoints  $a$  and  $b$ , we associate a continuous bijective function  $\hat{\omega} : [0, 1] \mapsto \omega$  so that  $\hat{\omega}(0) = a$  and  $\hat{\omega}(1) = b$ . Similarly to any continuous non-self-intersecting loop  $\omega$ , we associate a continuous bijective function  $\hat{\omega} : \mathbb{S}^1 \mapsto \omega$ . We orient  $[0, 1]$  from 0 to 1 and  $\mathbb{S}^1$  in the clockwise direction. This gives an orientation to  $\omega$ .

Let  $J \subseteq \mathbb{R}^2 \times \mathbb{S}^1$  be the set  $\{(w, u) : w \in \mathbb{R}^2, u \in D_w\}$ . Let  $\pi_1$  and  $\pi_2$  be projection functions that map a point  $(w, u)$  in  $J$  to  $w$  and  $u$  respectively. Let us also denote by  $\omega_i$ ,  $i \in \{1, 2\}$ , the curve defined by the function  $\hat{\omega}_i(t) = \pi_i(\hat{\omega}(t))$ . The domain of  $\hat{\omega}_i$  is the same as the domain of  $\hat{\omega}$  (either  $\mathbb{S}^1$  or  $[0, 1]$  depending on whether  $\omega$  is a closed loop). The orientation of  $\omega$  gives the orientation of  $\omega_i$ . When  $\omega$  is a closed loop, we define the winding number of  $\omega$  as the winding number of  $\hat{\omega}_2$ .

Let  $\gamma$  be a non-self-intersecting continuous curve in the plane with distinct endpoints and let  $\omega$  be a continuous curve in  $J$  so that  $\gamma = \omega_1$ . We call  $\omega$  a *walk* along  $\gamma$ . The next lemma shows that there exists a walk along any line segment in the plane.

**Lemma 5.** *Let  $s = (w_1, u_1)$  and  $t = (w_2, u_2)$  be two points in  $J$ . Let  $\sigma$  be the segment joining  $w_1$  and  $w_2$ . There exists a curve  $\omega$  in  $J$  joining  $s$  and  $t$  so that  $\omega_1 = \sigma$ , i.e.,  $\omega$  is a walk over  $\sigma$  with endpoints  $s$  and  $t$ . Furthermore,  $\omega$  can be computed in  $O(n^2 \log n)$  time.*

**Intuitive meaning:** The statement of the lemma uses a lot of notation in order to be precise. However, since this may make it seem more complicated, here is

the intuitive meaning. We want to move from  $w_1$  to  $w_2$  along the segment  $\sigma$  always maintaining a ray in the good cone of the current point. We start with the ray in the direction  $u_1$  when we are at  $w_1$  and the direction of the ray changes continuously as we move to  $w_2$  and we finish with the direction  $u_2$  at  $w_2$ . Along the motion from  $w_1$  to  $w_2$ , we are allowed to *stand* at a point  $p$  on  $\sigma$  and move the ray continuously within the good cone of  $p$ .

*Proof.* We first compute the intersection of  $\sigma$  with each line in  $\mathcal{L}$ . Let  $p_1, p_2, \dots, p_{k-1}$  be these points in the order of intersection. Let  $p_0 = w_1$  and  $p_k = w_2$ . Let  $I_i$  be the interval  $[p_i, p_{i+1}]$ . We will traverse  $\sigma$  from  $w_1$  to  $w_2$ , and construct  $\omega$  as we go along. The events in this traversal will be the intervals  $I_i$  and the points  $p_i$  in the order that they appear in the segment from  $w_1$  to  $w_2$ . As we sweep, we will maintain a data structure that gives us information about the good set of the interval or point that we are currently in. Using this information, we will compute a walk along each of the points  $p_i$  and each of the intervals  $I_i$  which when put together gives us a walk along  $\sigma$  with endpoints  $w_1$  and  $w_2$ . We construct  $2k$  points in  $J$ ,  $(p_1 = w_1, s_1), (p_1, t_1), \dots, (p_i, s_i), (p_i, t_i), \dots, (p_k = w_2, t_k)$ ; the curve they define gives the required walk.

Each of the intervals  $I_i$  lies entirely within a single cell of  $\mathcal{A}$ . Therefore, there is some  $g_i \in P$  so that  $g_i \in \mathcal{G}_x$  for all  $x \in I_i$ . For point  $i \in [1, \dots, k-1]$ , both  $g_{i-1}$  and  $g_i$  are in  $\mathcal{G}_{p_i}$ . Set  $s_i = \delta_{p_i, g_{i-1}}$  and  $t_i = \delta_{p_i, g_i}$ ; note that both these directions are in  $\mathcal{D}_{p_i}$ . Therefore, there is an interval  $K_i \subseteq \mathcal{D}_{p_i}$  with endpoints  $s_i$  and  $t_i$ . We define  $K_0$  as the interval contained in  $\mathcal{D}_{p_0}$  with endpoints  $u_1$  and  $\delta_{p_0, g_0}$  and we define  $K_k$  as the interval contained in  $\mathcal{D}_{p_k}$  with endpoints  $\delta_{p_k, g_{k-1}}$  and  $u_2$ . For the interval  $I_i$ , we define the walk  $\omega_{I_i} = \{(x, \delta_{x, g_i}) : x \in I_i\}$  and for each point  $p_i$  we define the walk  $\omega_{p_i} = \{(p_i, x) : x \in K_i\}$ . The walk  $\omega_{p_0}$  starts at the point  $(p_0, u_1)$  and ends at the point  $(p_0, \delta_{p_0, g_0})$  which is where  $\omega_{I_0}$  starts.  $\omega_{I_0}$  ends at  $(p_1, \delta_{p_1, g_0})$  which is where  $\omega_{p_1}$  starts and so on. Putting together these walks we get the required walk  $\omega$  from  $(w_1, u_1)$  to  $(w_2, u_2)$ .

In order to compute these walks, we will need to know the good cones in each of the intervals  $I_i$  and at each of the points  $p_i$ . We start by computing the good cone of  $p_0$  and the good cone of some point  $z$  in  $I_0$ . The good cone at  $z$  gives us  $g_0$  with which we compute  $\omega_{p_0}$  and  $\omega_{I_0}$ . We will update the data structure used to compute the good cone of  $z$  and obtain the good cone of  $p_1$ . In the data structure we have a representative  $y_i$  for each input point in  $p_i$  whose locations reflect their angular ordering around  $z$ . For convenience we will refer to the representatives by the points themselves. So, when we write “move  $p_i$  to a neighboring location”, we mean “move  $y_i$  to a neighboring location”. The point  $p_1$  is the intersection of  $\sigma$  with some line in  $\mathcal{L}$  passing through two input points  $a$  and  $b$ . There are two cases to consider depending on whether  $p_1$  lies on the edge  $ab$  or not. Assume that  $p_1$  lies on the edge  $ab$ . In this case we add the arc joining  $a$  and  $b$  which is not already in the data structure so that both arcs are present when we are at  $p_1$ . This reflects the fact that any ray emanating from  $p_1$  intersects the edge  $ab$ . When we move from  $p_1$  to  $I_1$ , we will remove the arc that was added first and keep the second one. Effectively as we move across  $p_1$ , we switch from one arc

formed by  $a$  and  $b$  to the other arc formed by  $a$  and  $b$ . Assume now that  $p_1$  does not lie on the edge  $ab$ . In this case, we move the point  $a$  to the location of  $b$  as we move from  $I_0$  to  $p_1$ . When we move from  $p_1$  to  $I_1$ , we move  $b$  to the previous location of  $a$ . This reflects the fact that as we move across  $p_1$  from  $I_0$  to  $I_1$ , the points  $a$  and  $b$  switch their positions in the angular order. When we are at  $p_1$ , they are at the same position. The rest of the sweep is done in a similar fashion. Each update takes  $O(\log n)$  time. Hence the total time required for the sweep is  $O(n^2 \log n)$ .  $\square$

Let  $\omega$  be a walk along a rectangle  $R$  in the plane and  $p_1 = (w_1, u_1)$  and  $p_2 = (w_2, u_2)$  be two points on  $\omega$  s.t.  $w_1$  and  $w_2$  lie on different edges of  $R$ . Let  $\sigma$  be a chord of  $R$  joining  $w_1$  and  $w_2$ . From Lemma 5, we obtain walk  $\tilde{\sigma}$  joining  $p_1$  and  $p_2$  in  $J$  such that  $\tilde{\sigma}_1 = \sigma$ . The points  $p_1$  and  $p_2$  split  $\omega$  into two arcs, one oriented from  $p_1$  to  $p_2$  and the other oriented from  $p_2$  to  $p_1$ . The curve  $\tilde{\sigma}$  splits the loop  $\omega$  into two loops  $\alpha$  and  $\beta$ . The loop  $\alpha$  traverses the arc of  $\omega$  from  $p_1$  to  $p_2$  followed by the curve  $\tilde{\sigma}$  from  $p_2$  to  $p_1$ . The loop  $\beta$  traverses curve  $\sigma$  from  $p_1$  to  $p_2$  followed by the the arc of  $\omega$  from  $p_2$  to  $p_1$ . Observe that the winding numbers of the loops  $\alpha$  and  $\beta$  add up to the winding number of the loop  $\omega$  because if we traverse  $\alpha$  followed by  $\beta$ , we traverse  $\tilde{\sigma}$  consecutively in opposite direction cancelling its effect with respect to the winding number. Therefore, if the winding number of  $\omega$  is non-zero, the winding number of one of the loops  $\alpha$  or  $\beta$  is non-zero. This gives us a way to find, from any loop of non-zero winding number, a *smaller* loop of non-zero winding number.

**Lemma 6.** *Let  $R$  be a rectangle with a walk  $\omega$  of non-zero winding number over it. There is point  $p$  inside  $R$  with Ray-Shooting depth at least  $n^2/9$ .*

*Proof.* Let  $\sigma$  be a vertical or horizontal chord of  $R$  that splits  $R$  along its longer side, into two rectangles  $R_1$  and  $R_2$  of equal area. From the above discussion, it follows that there is a walk of non-zero winding number along one of the rectangles  $R_1$  or  $R_2$ . We repeat this process with that rectangle. In this process, we get nested rectangles with smaller and smaller longer side and hence converge to a point  $p$  which has a non-zero winding number over it. This means that any ray emanating from  $p$  is good and hence  $p$  has Ray-Shooting depth at least  $n^2/9$ .  $\square$

The above lemma remains true even if  $R$  is any closed curve instead of a rectangle. We state it in terms of a rectangle because that is what we use for computational purposes.

Let  $R$  be a rectangle containing  $P$ . We will show that there is a walk  $\omega$  along  $R$  with a non-zero winding number. We will then split  $R$  into two rectangles  $R_1$  and  $R_2$  using a vertical chord  $\sigma$  of  $R$  which bisects the set of vertices of  $\mathcal{A}$  within  $R$ . We will find a walk  $\tilde{\sigma}$  along  $\sigma$  that splits  $\omega$  into two walks  $\alpha$  and  $\beta$  along  $R_1$  and  $R_2$  respectively. One of these will have a non-zero winding number. We will replace  $R$  by the rectangle that has a walk of non-zero winding number along

it and repeat the process. In each iteration, we reduce the number of vertices of  $\mathcal{A}$  in  $R$  by a factor of two. Since there are at most  $O(n^4)$  vertices to begin with, in  $O(\log n)$  iterations, we will find a rectangle  $R$  so that there is walk  $\omega$  along  $R$  with non-zero winding number and  $R$  has at most one vertex of  $\mathcal{A}$ . From Lemma 6, we can conclude that there is a point  $p$  of Ray-Shooting depth at least  $n^2/9$  in the region bounded by  $R$ . Since all points in the region bounded by  $R$  belong to a cell intersecting  $R$ , we can just check the  $O(n^2)$  cells intersecting  $R$  to find the required point. We will finally show that each iteration can be implemented in  $O(n^2 \log n)$  time. The overall running time of the algorithm will therefore be  $O(n^2 \log^2 n)$ .

We now show that there is a walk of non-zero winding number along any non-self-intersecting continuous loop  $\gamma$  enclosing the input point set. Let  $\mu = (3 - \sqrt{5})/6$  so that  $\mu(1 - \mu) = 1/9$ . It is easy to see that for any point  $p$  on  $\gamma$ , a ray  $r$  emanating from  $p$  is good if and only if it has at least  $\mu n$  input points on either side of the line containing  $r$ . This means that any point  $q \in \mathbb{R}^2$  with Tukey depth more than  $\mu n$  (w.r.t.  $P$ ) is in the good cone of every point  $p$  on  $\gamma$ . Since  $\mu < 1/3$ , the Centerpoint theorem guarantees that there is such a point  $o$ . For any point  $p$  on  $\gamma$ , let  $u_p \in \mathbb{S}^1$  be the direction  $\delta_{p,o}$ . Consider the walk  $\omega$  along  $\gamma$  such that  $\hat{\omega}(t) = (\hat{\gamma}(t), u_{\hat{\gamma}(t)})$ . Clearly the winding number of  $\omega$  is either  $+1$  or  $-1$  depending on whether  $\gamma$  itself has a winding number  $+1$  or  $-1$  around  $o$ .

For the purpose of our algorithm, we will start with a rectangle  $R$  which encloses the input points and none of the lines in  $\mathcal{L}$  intersect the horizontal sides of  $\mathcal{A}$ . We can assume without loss of generality that no two of the input points lie on the same vertical line. Therefore, such a rectangle always exists and can be computed in  $O(n^2)$  time. We then compute a point of Tukey depth at least  $\mu n$ . This can be done in  $O(n)$  time [8]. This gives us a walk  $\omega$  over  $R$ .

We can compute the number of vertices of  $\mathcal{A}$  between any two vertical lines  $l_1$  and  $l_2$  by comparing the top to bottom order of the intersection of the lines in  $\mathcal{L}$  with these two lines. Finding a vertical chord  $\sigma$  of  $R$  that splits the number of vertices evenly is therefore a simple slope selection problem and can be done in  $O(n \log n)$  time [6]. Let  $a$  and  $b$  be the endpoints on  $\sigma$  and let  $R_1$  and  $R_2$  be the rectangles that  $\sigma$  splits  $R$  into.

Using Lemma 5, we compute a walk  $\tilde{\sigma}$  over  $\sigma$  joining some points  $(a, u_a)$  and  $(b, u_b)$  on  $\omega$ .  $\tilde{\sigma}$  splits  $\omega$  into two walks  $\alpha$  and  $\beta$  along  $R_1$  and  $R_2$  respectively. Each of the walks consists of  $O(n^2)$  pieces which form  $\tilde{\sigma}$  and one piece along  $\omega$ . The curve  $\alpha_2$  (and similarly  $\beta_2$ ) is monotonic on each of these pieces. Hence the winding number of the walks  $\alpha$  and  $\beta$  can easily be computed from these pieces. We pick one of the rectangles  $R_1$  or  $R_2$  which has a walk of non-zero winding number along it and recurse. We do this until we have a rectangle that contains at most one vertex of  $\mathcal{A}$ . From Lemma 6, we know that there is a point of Ray-Shooting depth at least  $n^2/9$  inside the rectangle. Since the rectangle contains at most one vertex there is no cell contained completely in the interior of  $R$ . In our divide and conquer process, we have already checked all the cells

crossed by  $R$  and computed a good cone for a point in it. One of those points must have Ray-Shooting depth at least  $n^2/9$ . We have therefore shown that:

**Theorem 1.** *Given a set  $P$  of  $n$  points in the plane, a point of Ray Shooting depth at least  $n^2/9$  with respect to  $P$  can be computed in  $O(n^2 \log^2 n)$  time.*

### 3 Computing a point of Ray-Shooting depth approximately $n^2/9$

Suppose that we want to compute a point with Ray Shooting depth at least  $(1-\epsilon)n^2/9$ . We can do this by just taking a small random sample  $Q$  of the input point set  $P$  and computing a point of Ray-Shooting depth  $|Q|^2/9$  with respect to  $Q$  using Theorem 1. The same point has a Ray Shooting depth at least  $(1-\epsilon)n^2/9$  with respect to  $P$  with high probability. The sample  $Q$  is obtained by picking each point in  $P$  with probability  $p = O(\frac{1}{\epsilon^2} \log \frac{1}{\epsilon} \cdot \frac{1}{n})$ . The expected size of  $Q$  is  $pn = O(\frac{1}{\epsilon^2} \log \frac{1}{\epsilon})$  which is independent of  $n$  and depends only on the desired relative error  $\epsilon$ .

To prove that the above simple algorithm works, we will need to use the notion of  $\epsilon$ -approximations. Given a range space  $(V, \mathcal{R})$ , where  $V$  is the ground set and  $\mathcal{R}$  is a set of subsets of  $V$ , and a parameter  $0 < \epsilon < 1$ , an  $\epsilon$ -approximation for  $(V, \mathcal{R})$  is a subset  $X \subseteq V$  such that for each range  $R \in \mathcal{R}$ ,

$$\left| \frac{|R \cap X|}{|X|} - \frac{|R|}{|V|} \right| \leq \epsilon.$$

In other words, the relative size of  $R$  with respect to  $X$  does not differ from its relative size with respect to  $V$  by more than  $\epsilon$ . It is known that if the VC dimension of the range space is  $d$  then  $\epsilon$ -approximations of size  $O(d/\epsilon^2 \log 1/\epsilon)$  exist. In fact any random sample of this size is an  $\epsilon$ -approximation with high probability.

One way to exploit the existence of small  $\epsilon$ -approximations in our case would be to set  $V$  to be the set of all  $\binom{n}{2}$  edges defined by pairs of input points, and set  $\mathcal{R}$  to be precisely those subsets of the edges that could be intersected by a ray. Such a subset has a finite VC dimension and hence we could pick a random subset of the edges and compute a point of high Ray-Shooting depth with respect to these edges. Unfortunately, since in the random sample we have not picked all edges spanned by a subset of the points, it is not clear whether such a point exists and how it can be computed. Fortunately, it can be shown that instead of picking a random subset of the edges, it suffices to pick a random subset of the points and consider all edges spanned by them. This can be done by using the notion of product range spaces.

The following definitions and results are from [5]. Given two range spaces  $\Sigma_1 = (\mathcal{X}_1, \mathcal{R}_1)$  and  $\Sigma_2 = (\mathcal{X}_2, \mathcal{R}_2)$ , the *product range space*  $\Sigma_1 \otimes \Sigma_2$  is defined as

$(\mathcal{X}_1 \times \mathcal{X}_2, \mathcal{T})$  where  $\mathcal{T}$  consists of all subsets  $T \subseteq \mathcal{X}_1 \times \mathcal{X}_2$  such that each *cross-section*  $T_{x_2}^1 = \{x \in \mathcal{X}_1 : (x, x_2) \in T\}$  is a set of  $\mathcal{R}_1$  and each *cross-section*  $T_{x_1}^2 = \{x \in \mathcal{X}_2 : (x_1, x) \in T\}$  is a set of  $\mathcal{R}_2$ . It is shown in [5] that if  $A_1$  is an  $\epsilon_1$ -approximation for  $\Sigma_1$  and  $A_2$  is an  $\epsilon_2$ -approximation for  $\Sigma_2$ , then  $A_1 \times A_2$  is an  $\epsilon_1 + \epsilon_2$  approximation for  $\Sigma_1 \otimes \Sigma_2$ .

For any input point  $p \in P$  and any ray  $\rho$  emanating from a point  $q \in \mathbb{R}^2$ , the ray  $\rho$  and the line  $l$  through  $p$  and  $q$  define two closed cones (i.e., those contained in the halfspace defined by  $l$  that contains  $\rho$ ). Denote by  $C_{p,\rho}$  the cone that does not contain  $p$ . Let  $\Sigma = (P, \{C_{p,\rho} \cap P : p \in P, \rho \in \mathbb{R}^2 \times \mathbb{S}^1\})$  be the range space obtained by intersecting  $P$  with such cones.

For any ray  $\rho$  emanating from some point  $q$ , let  $E_\rho$  be the set of edges spanned by  $P$  that  $\rho$  intersects. Let  $\mathcal{T} = (P \times P, \{E_\rho : \rho \in \mathbb{R}^2 \times \mathbb{S}^1\})$  be the range space in which the ground set is the set of edges spanned by  $P$  and each range consists of the set of edges intersected by a ray.

It can be checked that  $\mathcal{T}$  is the product range space  $\Sigma \otimes \Sigma$ . Since  $\Sigma$  has a finite VC-dimension, picking a sample  $Q$  in which every point in  $P$  is chosen with probability  $p = O(\frac{1}{\epsilon^2} \log \frac{1}{\epsilon} \cdot \frac{1}{n})$  gives an  $\epsilon/2$  approximation for  $\Sigma$ . Therefore,  $Q \times Q$  gives an  $\epsilon$  approximation for  $\mathcal{T}$ . This justifies the algorithm described before. The running time of that algorithm is  $O(n + k^2 \log^2 k)$ , where  $k = pn = |Q|$ . It takes  $O(n)$  time to pick the sample  $Q$ . Once we have picked the sample, we run our previous algorithm on the sample  $Q$ . Thus we have the following theorem:

**Theorem 2.** *Given a set  $P$  of  $n$  points in the plane, a point  $z$  of Ray-Shooting depth at least  $(1-\epsilon)n^2/9$  with respect to  $P$  can be computed in  $O((1/\epsilon \log 1/\epsilon)^4 + n)$ .*

Setting  $\epsilon = \log n/n^{1/4}$  we see that a point of Ray Shooting depth at least  $n^2/9 - n^{7/4} \log n$  can be computed in linear time.

Once we have chosen a sample  $Q$  as above so that  $Q \times Q$  is an  $\epsilon$ -approximation for our product range space, we can also use it to compute a point of approximately the highest Ray-Shooting depth. This can be done by considering the set of lines  $\mathcal{L}$  defined by pairs of points in  $Q$  and computing the cell of highest depth in their arrangement  $\mathcal{A}$ . A brute force way is to separately compute the depth of each of the cells in  $\mathcal{A}$ . However, as we have seen in the proof of Lemma 5, the depth of all cells crossed by a line can be computed incrementally in  $O(k^2 \log k)$  time where  $k = |Q|$ . For each line  $l \in \mathcal{L}$ , we take two lines which are very close and parallel to  $l$ , one on each side of  $l$ . These lines together cross all cells of  $\mathcal{A}$ . We can therefore compute the cell of highest depth along each of these lines and then take the overall highest. This takes  $O(k^4 \log k)$  time. Since picking  $Q$  takes  $O(n)$  time, and  $k = O(1/\epsilon^2 \log 1/\epsilon)$ , we have:

**Theorem 3.** *Given a set  $P$  of  $n$  points in the plane, a point  $z$  of Ray-Shooting depth at least  $(1-\epsilon)d$ , where  $d$  is the maximum Ray-Shooting depth of any point with respect to  $P$ , can be computed in  $O(1/\epsilon^8 \log^5 1/\epsilon + n)$  time.*

## 4 Implementation

We have implemented the sampling-based algorithm from Theorem 3 to approximately find the point of highest RS depth. This has been integrated as a package in the popular statistical computing software R, and is available here:

<http://cran.r-project.org/web/packages/rsdepth/>.

*Acknowledgements.* The first author would like to thank Janos Pach for several enlightening discussions. The second author is thankful to Hans Raj Tiwary and Ujjyini Singh Ray for important ideas in the paper.

## References

1. A. Basit, N. H. Mustafa, S. Ray, and S. Raza. Centerpoints and Tverberg’s technique. *Computational Geometry: Theory and Applications*, 43(7):593–600, 2010.
2. A. Basit, N. H. Mustafa, S. Ray, and S. Raza. Hitting simplices with points in  $\mathbb{R}^3$ . *Discrete & Computational Geometry*, 44(3):637–644, 2010.
3. E. Boros and Z. Füredi. The number of triangles covering the center of an n-set. *Geometriae Dedicata*, 17(1):69–77, 1984.
4. T. M. Chan. An optimal randomized algorithm for maximum tukey depth. In *SODA*, pages 430–436, 2004.
5. B. Chazelle. *The Discrepancy Method*. Cambridge University Press, Cambridge, U.K., 2000.
6. R. Cole, J. S. Salowe, W. L. Steiger, and E. Szemerédi. An optimal-time algorithm for slope selection. *SIAM J. Comput.*, 18(4):792–810, 1989.
7. J. Fox, M. Gromov, V. Lafforgue, A. Naor, and J. Pach. Overlap properties of geometric expanders. In *SODA*, 2011.
8. S. Jadhav and A. Mukhopadhyay. Computing a centerpoint of a finite planar set of points in linear time. In *Symposium on Computational Geometry*, pages 83–90, 1993.
9. S. Langerman and W. L. Steiger. Optimization in arrangements. In *STACS*, pages 50–61, 2003.
10. R. Liu. A notion of data depth based upon random simplices. *The Annals of Statistics*, 18:405–414, 1990.
11. N. H. Mustafa and S. Ray. An optimal extension of the centerpoint theorem. *Computational Geometry: Theory and Applications*, 42(7):505–510, 2009.
12. H. Oja. Descriptive statistics for multivariate distributions. *Statistics and Probability Letters*, 1:327–332, 1983.
13. R. Rado. A theorem on general measure. *J. London. Math. Soc.*, 21:291–300, 1947.
14. P. J. Rousseeuw and I. Ruts. Constructing the bivariate tukey median. *Statistica Sinica*, 8:827–839, 1998.
15. J. Tukey. Mathematics and the picturing of data. In *Proc. of the international congress of mathematicians*, pages 523–531, 1975.
16. Y. Vardi and C. Zhang. The multivariate L1-median and associated data depth. *Proceedings of the National Academy of Sciences of the United States of America*, 97(4):1423, 2000.