Optimization in Arrangements

Stefan Langerman[1*] and William Steiger[2]

[1] School of Computer Science, McGill University
sl@cgm.cs.mcgill.ca
[2] Computer Science, Rutgers University
steiger@cs.rutgers.edu

Abstract. Many problems can be formulated as the optimization of functions in R^2 which are implicitly defined by an arrangement of lines, halfplanes, or points, for example linear programming in the plane. We present an efficient general approach to find the optimum exactly, for a wide range of functions that possess certain useful properties. To illustrate the value of this approach, we give a variety of applications in which we speed up or simplify the best known algorithms. These include algorithms for finding robust geometric medians (such as the Tukey Median), robust regression lines, and ham-sandwich cuts.

1 Introduction

Given a set L of n non-vertical lines $\ell_1, \ldots, \ell_n$, where line ℓ_i has equation $y = \ell_i(x) = a_i x + b_i$, many problems can be formulated as the optimization of some function $f_L : R^2 \to R$ implicitly defined by L. Similarly, given a set P of n points $p_1, \ldots, p_n$, many problems can be formulated as the optimization of some function $f_P : R^2 \to R$ implicitly defined by P. Without loss of generality, we will restrict ourselves to minimization problems.

We outline a general approach to design simple and efficient algorithms to optimize functions that possess certain good properties. First, these functions should be somehow connected to the combinatorial structure of the arrangement of lines or of points on which they are defined. For a function f_L on an arrangement of lines, we assume it is known how to find an optimum point inside a cell of the arrangement. For a function f_P on a set of points, the combinatorial structure is more complex. Consider the $\binom{n}{2}$ lines joining every pair of points. These lines decompose R^2 into $O(n^4)$ cells. We assume that it is known how to find the optimum point inside each of these cells. Note that in each cell, the ordering of the slopes of the lines from q to the points in P is the same for every q in the cell. The time to find the optimum inside a cell will be denoted $T_C(n)$. We also assume that within the same time bounds, we can find an optimum of f_P restricted to a line segment, or to a small convex polygon Q that does not intersect any line of the arrangement. We call the function that returns an optimum inside a cell the *cell tool*. Moreover, we assume we have access to one of the following tools:

* chargé de recherches du FNRS à l'Université Libre de Bruxelles

Sidedness: Given a line ℓ, the sidedness tool decides in time $T_S(n)$ which of the two closed halfplanes bounded by ℓ contains an optimum point for f.

Restricted sidedness: This applies to a function f_L on an arrangement of lines. Given a line ℓ and a convex polygon Q (given as a list of its vertices) known to contain an optimum point, restricted sidedness performs sidedness on ℓ in time $T_R(m)$, where m is the number of lines in L that intersect Q.

Witness: Given a point $p \in R^2$, the witness tool returns in time $T_W(n)$ a halfplane h containing p such that $f(q) \geq f(p)$ for all $q \in h$. If the function f is *level convex*, i.e. the set $\{q \in R^2 | f(q) \leq t\}$ is convex for all t, then a witness is guaranteed to exist for every point p.

The main results of this paper are stated in terms of these primitive operations:

Theorem 1. *Suppose we have a restricted sidedness tool for a function f_L on a set L of n lines, as well as a cell tool for f_L. Then we can find an optimum of f_L in time $O(n + T_R(n) + T_C(n))$. If we only have a sidedness tool the complexity to optimize f_L is $O((n + T_S(n)) \log n + T_C(n))$*

Theorem 2. *Suppose we have a function f_P on a set P of n points, and a sidedness tool for f_P. We can obtain an optimum in time $O(n \log^3 n + T_S(n) \log n + T_C(n))$.*

Theorem 3. *Suppose we have a level convex function f_P on a set P of n points, and a witness tool for f_P. We can obtain an optimum in time $O((n \log n + T_W(n)) \log^2 n + T_C(n) \log n)$.*

We prove these statements via algorithms shown to have the asserted complexity. For arrangements of points, the algorithms we describe are probabilistic, but they can be derandomized within the same time bounds using parametric search. Note that the randomized algorithms are extremely simple and should be quite easy to implement.

These theorems were motivated by several specific geometric optimization problems. We present some of these applications in Section 2 as an important part of the contribution of the paper, and to illustrate the potential value and range of applicability of our methods. Some of the applications significantly improve previously known algorithms for those problems. We also present applications which, though they only achieve the time bounds of known methods, do so using much simpler algorithms. The proofs of the theorems appear in Section 3 and Section 4, and contain some results of independent interest.

Throughout the paper we apply the duality transform where a point $q = (a, b)$ maps to the line $D_q = \{(x, y) : y = ax + b\}$ and the non-vertical line $\ell = \{(x, y) : y = mx + b\}$ maps to the point $D_\ell = (-m, b)$ (and a vertical line $x = t$ would map to the point (t) at infinity, the point incident with all lines of slope t; the point (∞) at vertical infinity is incident with all vertical lines). It is familiar that if q is (i) above, (ii) incident with, or (iii) below line ℓ then also D_q is (i) above, (ii) incident with, (iii) below D_ℓ, and also that the vertical distance of q from ℓ is preserved under D.

2 Some Applications

We show how to use the three main theorems to solve a variety of geometric optimization problems. In addition to presenting these algorithms themselves, we want to illustrate the applicability and utility of our methods. For each application we give a brief background and state the best current solution. Then we describe our new algorithm via the relevant theorem, give its complexity, and try to indicate how the particular sidedness, witness, and cell tools can be constructed. The full details will appear in the final paper.

The first four pertain to functions defined by arrangements of points and rely on Theorems 2 and 3. The resulting algorithms improve and simplify previous algorithms for these tasks. They also constitute the first known efficient algorithms for these problems that are simple enough to be implemented.

1. Tukey median for points: A given set P of n points in R^2 is used to define depth. The *Tukey depth* [27], or *location depth* of a point $q \in R^2$ is the smallest number of points of P in any halfplane containing q. That is,

$$\tau_P(q) = \min_{\text{halfspace } h \ni q} |h \cap P| \tag{1}$$

A Tukey median is a point of maximum depth.

A well known consequence of Helly's Theorem (e.g., [14]) is that there is a point $x \in \mathbb{R}^2$ (not necessarily in P) of depth at least $\lceil n/3 \rceil$. Such a point is called a *centerpoint*. The *center* is the set of all centerpoints.

Cole, Sharir and Yap [10] described an $O(n(\log n)^5)$ algorithm to construct a centerpoint, and subsequent ideas of Cole [9] could be used to lower the complexity to $O(n(\log n)^3)$. Recently, Jadhav and Mukhopadhyay [17] described a linear time algorithm to find a centerpoint.

Matoušek [20] attacked the harder problem of computing a Tukey median and presented an $O(n(\log n)^5)$ algorithm for that task. The algorithm uses two levels of parametric search and ε-nets, and would be quite difficult to implement. Despite the fact that the Tukey median is of genuine interest in statistical applications, there is no really usable implementation. The fastest algorithms actually implemented for this task have complexity $\Theta(n^2)$ [22]. We can prove

Lemma 1. *Given a set P of n points in R^2, a Tukey median can be found in $O(n \log^3 n)$ time.*

Proof. Given $q \in R^2$, we sort the points in P in radial order from q. Thus the depth of q and a witness halfplane is obtained in $T_W(n) = O(n \log n)$. Every cell in the arrangement of the $\binom{n}{2}$ lines joining pairs of points in P has all its points of the *same* depth, so the cell tool can return any point in $T_C(n) = O(1)$. Finally since τ_P is level convex, Theorem 3 may be applied. □

Besides being much faster than other known methods, this algorithm could be implemented easily.

2. **A Tukey median for convex polygons:** Applications in computer vision motivated a variant of Tukey depth that is defined with respect to the area within a given polygon Q. The depth (in Q) of $p \in R^2$ is defined by

$$\tau_Q(p) = \min_{\text{halfspace } h \ni p} Area(h \cap Q) \tag{2}$$

A median is a point in Q of maximal depth.
Diaz and O'Rourke [11–13] developed an $O(n^6 \log^2 n)$ algorithm for finding the Tukey median, n being the number of vertices of Q. If Q is convex Brass and Heinrich-Litan were able to compute a median in time $O((n^2 \log^3 n\alpha(n))$ [6]. We can show that

Lemma 2. *Given a convex polygon $Q \subset R^2$ with n vertices, a median may be found in time $O(n \log^3 n)$.*

Proof. For a convex polygon with its points given in clockwise order, a witness can be found in $O(n)$ time, and it can be shown that the optimum of τ_Q inside a cell can be found in $O(n)$ time. The statement now follows from Theorem 3. □

Recently, Braß, Heinrich-Litan and Morin gave a randomized linear time algorithm for this task [7].

3. **Oja median:** Given a set P of n points, the Oja depth of a point $q \in R^2$ is the sum of the areas of the triangles formed by q and every pair of points in P. An Oja median is a point if minimum depth.
The first algorithm for finding the Oja median was presented by Niinimaa, Oja and Nyblom [24] and ran in time $O(n^6)$. This was then improved to $O(n^5 \log n)$ by Rousseeuw and Ruts, and then to $O(n^3 (\log n)^2)$ and $O(n^2)$ space by Aloupis, Soss and Toussaint [2].
The Oja median has to lie on the intersection between two lines of $\binom{P}{2}$ [24], the Oja depth function can be shown to be level convex [2], and a witness can be computed in $T_W(n) = O(n \log n)$ time, and so, using Theorem 3, it is shown in [1] that optimum can be found in $O(n \log^3 n)$ time.

4. **Ham-sandwich cut in arrangements:** A given set $L = \{\ell_1, \ldots, \ell_n\}$ of non-parallel, non-vertical lines in the plane defines $\binom{n}{2}$ vertices $V = \{\ell_i \cap \ell_j, i < j\}$. Consider subsets A and B of those points each specified by some convex polygonal region, or by the union of a constant number of such regions. A ham-sandwich cut for A and B is a line that simultaneously splits the vertices in both A and B exactly in half. We can show that using a duality transform, this problem corresponds to minimizing a function for an arrangement of points for which sidedness can be decided in $T_S(n) = O(n \log^2 n)$ time. Applying Theorem 2 we get

Lemma 3. *A ham-sandwich cut for subsets A and B of the vertices of an arrangement of n lines can be found in $O(n \log^3 n)$ time.*

This compares to $O(n^2)$ complexity of the optimal planar ham sandwich cut algorithm when applied to $O(n^2)$ points in R^2 [19]. Details are omitted here.

The next three applications pertain to functions defined on line arrangements. Although they do not improve the time bounds of known algorithms, they are simple, and they are of interest in illustrating the range of applicability of the technique.

5. Huber M-estimators: Huber (see [16]) proposed an interesting and useful method of robust regression: Given n points (x_i, y_i) find a line $y = mx + b$ to minimize

$$\sum_{i=1}^{n} \rho_k(y_i - (mx_i + b)),$$

where $\rho_k(t) = t^2/2$ if $|t| \leq k$ and $k|t| - k^2/2$ otherwise; k is a given non-negative constant. In the dual we seek $p = (p_x, p_y) \in R^2$ to minimize

$$f_L(p) = \sum_{i=1}^{n} \rho_k(p_y - (m_i p_x + b_i)),$$

given lines $\ell_i = \{(x, y) : y = m_i x + b_i\}$. Restricted sidedness can be performed in $O(m)$ time and the optimum in a cell of the arrangement of the n given lines can be found in $O(n)$ time. Using Theorem 1 we obtain:

Lemma 4. *The Huber M-estimator for n given points in the plane can be computed in time $O(n)$.*

The first finite algorithm for the Huber M-estimator ([8]) was not even known to be polynomial. If we take $k = 0$ we also get a linear time algorithm for L_1 regression. A linear time algorithm for computing the L_1 regression line was first discovered by Zemel [29] by reducing the problem to an instance of the Multiple Choice L.P. Problem (MCLPP), which Megiddo's algorithm for linear programming [21] can solve in linear time.

6. Fermat-Torricelli for lines: The Fermat-Torricelli points for a set $L = \{\ell_1, \ldots, \ell_n\}$ of lines are defined to be the points that minimize:

$$f_L(p) = \sum_{i=1}^{n} d_\perp(p, \ell_i)$$

where $d_\perp(p, \ell)$ denotes the perpendicular distance between point p and line ℓ.

Roy Barbara [5] showed that a Fermat-Torricelli point for a set of n lines can always be found at a vertex of the arrangement of L. He then proposed to evaluate f_L at every vertex of the arrangement, obtaining an $O(n^3)$ algorithm. Using Theorem 1, Aloupis et. al. [1] show:

Lemma 5. *Given a set L of n lines, a Fermat-Torricelli point for L can be found in $O(n)$ time.*

In their proof they show that restricted sidedness can be performed in $O(m)$ time. The statement now follows easily by applying Theorem 1.

7. Hyperplane depth: The hyperplane depth $\delta_L(p)$ of a point p with respect to a set L of lines is the minimum number of lines that intersects any ray from p. The Hyperplane median is a point p that maximizes $\delta_L(p)$. Note that the depth of all the points inside a cell is the same.

Hyperplane depth was introduced by Rousseeuw and Hubert [25], motivated by problems in robust regression. Using the duality transform, the lines in L are mapped to points and the *regression depth* of a line in dual space is the minimum number of points the line must meet in a rotation to vertical. This corresponds exactly to the hyperplane depth of the corresponding point in the original problem.

Rousseeuw and Hubert point out that $\delta_L(p)$ can be computed in time $O(n \log n)$ for any p, and since there are $O(n^2)$ vertices in the arrangement, the hyperplane median can be found in $O(n^3 \log n)$. They also mention an $O(n^3)$ algorithm [26]. In Amenta et. al. [3] it was observed that the arrangement of L can be constructed in $O(n^2)$ and then, using breadth-first-search on the graph of adjacent cells, the depth of every cell is obtained in the same $O(n^2)$ time. Van Kreveld et.al. [28] described an $O(n \log^2 n)$ algorithm for finding the hyperplane median, using an $O(n \log n)$ algorithm to perform sidedness on a line.

In [18], we showed that restricted sidedness could be performed in $O(m \log m)$ time. Using Theorem 1, we obtained:

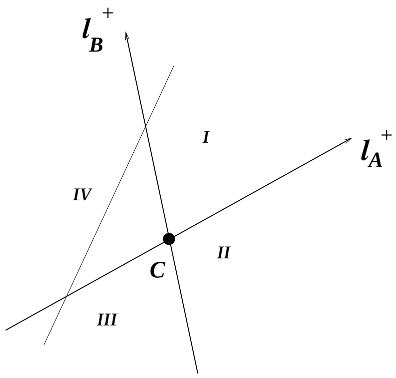
Lemma 6. *Given a set L of n lines, a cell of maximum depth can be found in time $O(n \log n)$.*

3 Arrangements of lines

3.1 Equipartitions

To do efficient pruning in the line arrangement, we needed a partitioning method. Given lines $L = \{\ell_1, \ldots, \ell_n\}$ in general position in R^2 and two other non-parallel lines ℓ_A and ℓ_B, not in L, nor parallel to any line in L, we write $C = \ell_A \cap \ell_B$. Every line in L crosses both ℓ_A and ℓ_B. Partition the lines in L into 4 sets, one for each quadrant ($I = +,+$, $II = +,-$, $III = -,-$, $IV = -,+$), depending on whether the line crosses ℓ_A above or below C, and whether it crosses ℓ_B above or below C. A line that belongs to one of these sets (quadrants) avoids the opposite quadrant. We say that ℓ_A and ℓ_B form an *α-partition* of L if each of the four groups contains at least α lines. We prove the following result, of independent interest.

Lemma 7. *Let $L = \{\ell_1, \ldots, \ell_n\}$ be a set of n lines in general position in R^2. There exists an $\lfloor n/4 \rfloor$-partition of L and it can be found in time $O(n)$.*



Proof. Let μ_j denote the j^{th} smallest slope of the lines in L and let $\mu = (\mu_{\lfloor n/2 \rfloor} + \mu_{1+\lfloor n/2 \rfloor})/2$. The median level of the lines in L with slope less than μ (call this set L_1) and the median level of the lines in L with slope greater than μ (call this set L_2) cross at a unique point $C \in R^2$. Take ℓ_A to be the line of slope μ incident with C and ℓ_B to be the vertical incident with C. They form an $\lfloor n/4 \rfloor$-partition of L since half the lines in L_1 meet ℓ_A to the left of C and ℓ_B below C, and half meet ℓ_B above C and meet ℓ_A to the right of C. Similarly the lines of L_2 split into the two remaining quadrants.

In fact the construction could choose ℓ_A to be a line of arbitrary slope. The set L_1 would be the next $n/2$ lines in the (clockwise) radial ordering, and L_2 the remaining $n/2$ lines. The plane is then rotated so that vertical direction separates the last line in L_1 (in clockwise ordering) from the first line of L_2. Finally we point out that C is the dual of the ham-sandwich cut for the (separated) sets of points to which L_1 and L_2 are dual and so can be found in $O(n)$ time [19]. $\square$

There are analogous existence statements about equipartitions of *points* in R^d by d hyperplanes when $d \leq 3$. When $d > 4$ equipartitions of points do not always exist; the case $d = 4$ is open (see e.g., Avis [4]). It would be interesting to know if there exist three hyperplanes that equipartition n given hyperplanes in R^3.

Even though there is an algorithm that finds a $n/4$-partition in linear time, it is not trivial to implement. However there is an extremely easy way to randomly generate a $n/8$-partition with positive constant probability. Pick a line ℓ uniformly at random in L. Then pick randomly in L one line with slope smaller than ℓ and one line with slope larger than ℓ and let C be the intersection of those two lines. Then pick ℓ_A to be the line with the same slope as ℓ through C, and ℓ_B to be the vertical line through C. We can show that with constant probability $p > 0$, ℓ_A and ℓ_B form a $n/8$-partition. This then gives a Las Vegas algorithm in the usual way.

3.2 Line pruning

With the notion of equipartitions, the algorithm for Theorem 1 is quite easy to state.

Algorithm LineOpt(L)

$\hat{L} \leftarrow L$

$Q \leftarrow R^2$

while $|\hat{L}| > 10$

Find an $|\hat{L}|/c$**-partition** ℓ_A, ℓ_B, **for some constant** $c > 1$

Perform a *sidedness test* on ℓ_A, ℓ_B **restricted to** Q

Let h_A **and** h_B **be the two halfplanes returned by the sidedness test.**

$Q \leftarrow Q \cap h_A \cap h_B$

$\hat{L} \leftarrow \{\ell \in \hat{L} | \ell \cap h_A \cap h_B \neq \phi\}$

endwhile

perform a sidedness test on all the lines in $\hat{L}$.

return the optimum in the cell that contains the intersection of all the halfplanes returned and Q.

For correctness, the invariant that Q contains an optimum point is maintained throughout the algorithm, and at the end, all the cells remaining in Q are searched for the optimum point. Since at every step $\hat{L}$ is reduced by a constant factor $\geq 1/4$, the total number of steps is $O(\log n)$ and the running time is $O((n + T_S(n)) \log n + T_C(n))$ if sidedness is used. If restricted sidedness is used, the running time inside the loop forms a geometric progression, and the total running time is $O(n + T_R(n) + T_C(n))$. If the randomized algorithm is used to find the partition, then the running times of the algorithm are the same in the expected sense. This completes the proof of Theorem 1. □

4 Arrangements of points

4.1 Best of candidates

Functions for arrangements of points seem to be more difficult to optimize. We first show an algorithm for a slightly simpler problem: suppose we are given a set A^* of candidate points, the algorithm returns a point p such that $f_P(p) \leq f_P(q)$ for every $q \in A^*$. The algorithm maintains a set A of points "to beat" and uses the witness tool. We will again need a pruning tool, but this time for points. Recall (from the introduction) that the Tukey depth $\tau_P(q)$ of a point q with respect to a set P of points is the minimum number of points of P that are contained in any halfplane that contains q. It is known that a point of depth $|P|/3$ always exists and can be found in linear time [17]. There is again a very simple way to obtain a point of depth $|P|/8$ with positive constant probability: pick a point at random in P and draw a vertical line through it. Then pick at random a point to the left, and a point to the right of that line. Join the two points by a line and call p the intersection of that line and the vertical line. It can be shown that with positive constant probability, $\tau_P(p) \geq |P|/8$.

Algorithm Bestof(A^*)

$A \leftarrow A^*$; $p^* \leftarrow$ **some arbitrary point of** A.
$Q \leftarrow R^2$
while $A \neq \phi$ **repeat**
  find a point p **such that** $\tau_A(p) \geq c|A|$ **for some constant** $c \leq 1/3$.
  compute $f_P(p)$. **Let** h **be a witness for** p.
  if $f_P(p) \leq f_P(p^*)$ **then** $p^* \leftarrow p$ **endif.**
  $Q \leftarrow Q - h$
  $A \leftarrow A - (A \cap h)$
endwhile

The invariant of the algorithm is that $f_P(p^*) \leq f_P(q)$ for all $q \in A^* - A$. The invariant is true at the beginning since $A^* - A = \phi$, and the invariant is preserved after each step of the while loop: $f_P(p^*) \leq f_P(p) \leq f_P(q)$ for all $q \in (A \cap h)$. As for the running time of the algorithm, we know that $|A \cap h| \geq c|A|$ because $\tau_A(p) \geq c|A|$. Thus the size of A reduces by a constant factor at every step of the loop, and so the number of loop steps is bounded by $O(\log |A^*|)$. Each step of the loop takes time $O(T_D(A) + T_W(n))$ where $T_D(A^*)$ is the time needed to compute a point p of depth $\tau(p) \geq |A^*|/c$ for some constant $c \leq 1/3$. The overall running time is $O((T_D(A^*) + T_W(n)) \log |A^*|)$ or $O(T_D(A^*) + T_W(n) \log |A^*|)$ if $T_D(A^*) = \Omega(|A^*|^\varepsilon)$ for some $\varepsilon > 0$. This is also $O(|A^*| + T_W(n) \log |A^*|)$ since we may take $T_D(A) = O(|A|)$ (e.g. use the algorithm of Jadhav and Mukhopadhyay [17] for finding a centerpoint, or the randomized method shown above).

For example, if we set A^* to the set of all $O(n^4)$ optima for each cell of the arrangement of lines in $\binom{P}{2}$, the algorithm will find the optimum for f_P, but the running time would be $O(n^4)$ if we have to enumerate all of A^* to use the centerpoint algorithm. One could think of designing a faster algorithm to find a point of Tukey depth $\geq c|A|$ for the candidate points inside Q without explicitly maintaining A, but this seems difficult even if the candidates are the vertices of the arrangement of lines in $\binom{P}{2}$, because even deciding whether Q contains any vertex of the arrangement is 3SUM-hard (for a definition of 3SUM-hardness, see [15]). Details of this fact appear in the full version.

4.2 Witness to sidedness

In this section, we show that the witness tool can be used to perform a sidedness test on any given line.

Lemma 8. *If a witness tool is available for a function f_P on an arrangement of n points P, then a sidedness test for a line ℓ can be computed in time $T_S(n) = O((n \log n + T_W(n)) \log n + T_C(n))$.*

Proof. Consider the set of lines $\binom{P}{2}$ and let A^* be the set of the $\binom{n}{2}$ intersections between those lines and ℓ. Those intersections divide ℓ into $O(n^2)$ regions. If we run the algorithm of the previous section, at the end of the algorithm, $s = Q \cap \ell$ is a segment that does not intersect any of the $\binom{n}{2}$ lines. We can then use the cell tool to find the optimum point p^* on s in time $T_C(n)$. Neither the witness infinitesimally to the left of p^* nor the one infinitesimally to its right contain p^*, so they exclude one side of the line.

Since $|A^*| = \Theta(n^2)$, using the usual centerpoint construction algorithm for the set A in the main loop of the algorithm would take $O(n^2)$ time. We will reduce this to $O(n \log n)$ by exploiting the structure of the problem.

First notice that all the points in A^* lie on the straight line ℓ. This implies that there is a point on ℓ with Tukey depth $|A|/2$ with respect to any $A \subseteq A^*$. At any moment of the high depth algorithm, $A = A^* \cap R$ for some convex polygon R since every step eliminates a halfplane. In our case, this means that A corresponds to the points of A^* that lie inside a segment on ℓ. We will construct a point of Tukey depth $|A|/2$ with respect to A using the duality transform. First, rotate the plane so that ℓ is vertical, then apply the duality transform. P becomes a set of lines, and ℓ becomes a point T_ℓ at infinity. Rotate the dual plane so that that point becomes the point at vertical infinity. Any point a in A^* is at the intersection between ℓ and a line ℓ_{ij} that connects the two points p_i and p_j of P. Thus $T_{\ell_{ij}}$ is the vertex of the dual arrangement at the intersection of the lines T_{p_i} and T_{p_j}, and a is the vertical line passing through this vertex and the point T_v at vertical infinity. From this, we conclude that the line segment in v corresponding to A at some point of the algorithm is a vertical slab in the transformed space, and a point of Tukey depth $|A|/2$ with respect to A can be found using vertex selection inside that slab. This can thus be done in $O(n \log n)$ time. More easily we can generate a point at random inside that slab by counting the number of inversions between the ordering of the lines along the left boundary of the slab, and the ordering of the lines along the right boundary of the slab, in $O(n \log n)$ time. $\square$

4.3 Line pruning

Now that we have a sidedness test, we could use the algorithm from Section 3 to prune the lines, except that we have to be able to generate a $|\hat{L}|/c$-partition quickly even though $\hat{L}$ might be $\Theta(n^2)$. For this, we maintain $\hat{L}$ implicitly as the set of lines in $\binom{P}{2}$ that intersect Q, and we maintain Q.

As we saw before, generating a good partition randomly only requires to be able to generate random lines from $\hat{L}$. Using the duality transform, one can notice that this is equivalent to generating a random vertex in an arrangement amongst the ones inside a certain region. But this can be done in $O(n \log n)$ time using a vertex counting algorithm of Mount and Netanyahu [23]. Their algorithm is itself quite simple and easy to implement. This completes the proof of Theorems 2 and 3. $\square$

References

1. G. Aloupis, S. Langerman, M. Soss, and G. Toussaint. Algorithms for bivariate medians and a fermat-torricelli problem for lines. In *Proc. 13th Canad. Conf. Comput. Geom.*, 2001.

2. G. Aloupis, M. Soss, and G. Toussaint. On the computation of the bivariate median and the fermat-torricelli problem for lines. Technical Report SOCS-01.2, School of Computer Science, McGill University, Feb. 2001.
3. N. Amenta, M. Bern, D. Eppstein, and S.-H. Teng. Regression depth and center points. *Discrete Comput. Geom.*, 23(3):305–323, 2000.
4. D. Avis. On the partitionability of point sets in space. In *Proc. 1st Annu. ACM Sympos. Comput. Geom.*, pages 116–120, 1985.
5. R. Barbara. The fermat-torricelli points of n lines. *Mathematical Gazette*, 84:24–29, 2000.
6. P. Brass and L. Heinrich-Litan. Computing the center of area of a convex polygon. Technical Report B 02-10, Freie Universität Berlin, Fachbereich Mathematik und Informatik, March 2002.
7. P. Brass, L. Heinrich-Litan, and P. Morin. Computing the center of area of a convex polygon. Technical report, 2002.
8. D. I. Clark and M. R. Osborne. Finite algorithms for Huber's M-estimator. *SIAM J. Sci. Statist. Comput.*, 7(1):72–85, 1986.
9. R. Cole. Slowing down sorting networks to obtain faster sorting algorithms. *J. ACM*, 34(1):200–208, 1987.
10. R. Cole, M. Sharir, and C. K. Yap. On k-hulls and related problems. *SIAM J. Comput.*, 16:61–77, 1987.
11. M. Díaz and J. O'Rourke. Computing the center of area of a polygon. In *Proc. 1st Workshop Algorithms Data Struct.*, volume 382 of *Lecture Notes Comput. Sci.*, pages 171–182. Springer-Verlag, 1989.
12. M. Díaz and J. O'Rourke. Chord center for convex polygons. In B. Melter, A. Rosenfeld, and P. Bhattacharyai, editors, *Computational Vision*, pages 29–44. American Mathematical Society, 1991.
13. M. Díaz and J. O'Rourke. Algorithms for computing the center of area of a convex polygon. *Visual Comput.*, 10:432–442, 1994.
14. H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Heidelberg, West Germany, 1987.
15. A. Gajentaan and M. H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom. Theory Appl.*, 5:165–185, 1995.
16. P. Huber. *Robust Statistics*. John Wiley, NY, 1981.
17. S. Jadhav and A. Mukhopadhyay. Computing a centerpoint of a finite planar set of points in linear time. *Discrete Comput. Geom.*, 12:291–312, 1994.
18. S. Langerman and W. Steiger. An optimal algorithm for hyperplane depth in the plane. In *Proc. 11th ACM-SIAM Sympos. Discrete Algorithms*, 2000.
19. C.-Y. Lo, J. Matoušek, and W. L. Steiger. Algorithms for ham-sandwich cuts. *Discrete Comput. Geom.*, 11:433–452, 1994.
20. J. Matoušek. Computing the center of planar point sets. In J. E. Goodman, R. Pollack, and W. Steiger, editors, *Computational Geometry: Papers from the DIMACS Special Year*, pages 221–230. American Mathematical Society, Providence, 1991.
21. N. Megiddo. Linear-time algorithms for linear programming in R^3 and related problems. *SIAM J. Comput.*, 12:759–776, 1983.
22. K. Miller, S. Ramaswami, P. Rousseeuw, T. Sellars, D. Souvaine, I. Streinu, and A. Struyf. Fast implementation of depth contours using topological sweep. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 690–699. ACM Press, 2001.
23. D. M. Mount and N. S. Netanyahu. Efficient randomized algorithms for robust estimation of circular arcs and aligned ellipses. Technical report, Dec. 1997.

24. A. Nniinimaa, H. Oja, and J. Nyblom. The oja bivariate median. *Applied Statistics*, 41:611–617, 1992.
25. P. J. Rousseeuw and M. Hubert. Depth in an arrangement of hyperplanes. *Discrete Comput. Geom.*, 22(2):167–176, 1999.
26. P. J. Rousseeuw and M. Hubert. Regression depth. *J. Amer. Statist. Assoc.*, 94(446):388–402, 1999.
27. J. W. Tukey. Mathematics and the picturing of data. In *Proceedings of the International Congress of Mathematicians (Vancouver, B. C., 1974), Vol. 2*, pages 523–531. Canad. Math. Congress, Montreal, Que., 1975.
28. M. van Kreveld, J. Mitchell, P. Rousseeuw, M. Sharir, J. Snoeyink, and B. Speckmann. Efficient algorithms for maximum regression depth. In *Proc. 15th ACM Symp. Comp. Geom.*, pages 31–40, 1999.
29. E. Zemel. An $O(n)$ algorithm for the linear multiple choice knapsack problem and related problems. *Inform. Process. Lett.*, 18(3):123–128, 1984.