Algorithms for Bivariate Majority Depth

Dan Chen[*] Pat Morin[*]

Abstract

The majority depth of a point with respect to a point set is the number of major sides it is in. An algorithm for majority depth in $\mathbb{R}^2$ is given in this paper, and it is the first algorithm to compute the majority depth. This algorithm runs in $O((n+m)\log n)$ time with Brodal and Jacob's data structure, and in $O\left((n+m)\frac{\log n}{\log\log n}\right)$ time in the word RAM model.

1 Introduction

A data depth is a measure of the centrality of a point with respect to a given data cloud in $\mathbb{R}^d$. Many depth notions have been introduced, such as *Tukey depth* [21], *Oja depth* [17], *Simplicial depth* [15], and *majority depth* [18, 16]. For the introduction of these notions, one can refer to the surveys by Small [19] and Aloupis [2]. In this paper we give an algorithm for the majority depth. Let S be a set of points in $\mathbb{R}^d$. If the points in S are in general position (no $d+1$ points of S lie on a common hyperplane), any d points in S define a unique hyperplane $\hbar$. With $\hbar$ as the common boundary, two closed half-spaces are obtained. The one containing more than or equal to $\frac{n+d}{2}$ points is called the major side of $\hbar$. Note that halving hyperplanes have two major sides. Given a finite set S of n points and a point p in $\mathbb{R}^d$, the majority depth of p is the number of major sides it is in.

In this paper we consider the problem of computing the majority depth of a point p with respect to a set S of n points in $\mathbb{R}^2$. We assume that the points in S are in general position. The tools for the algorithm are given in Section 2 and Section 3, and the algorithm is given in Section 4.

2 Dual Arrangement

Let H be a set of n hyperplanes in $\mathbb{R}^d$. We say that H is in general position, if every subset of d hyperplanes intersect in one point, and no $d+1$ hyperplanes intersect in one point. We say a hyperplane is *vertical* if it contains a line parallel to the x_d-axis. Without loss of generality, we assume that no hyperplane in H is vertical. The *arrangement* $\mathcal{A}(H)$ of H is the partitioning of $\mathbb{R}^d$ induced by H into *vertices* (intersections of any d hyperplanes in H), *faces* (each flat in $\mathcal{A}(H)$ is divided into pieces by the hyperplanes in H that do not contain the flat, a j-face is a piece in a j-flat), and *regions* (connected components in $\mathbb{R}^d$ separated by hyperplanes in H). We call $\mathcal{A}(H)$ a simple arrangement if H is in general position.

In an arrangement, we say a point p is at the k-level [1, 11, 14], if there are k hyperplanes in H lying vertically below p. (Above and below are with respect to the x_d coordinate.) The k-level of $\mathcal{A}(H)$ is the closure of all the points of H at level k. Let m be the number of vertices of the k-level. Tight bounds for m are still open problems. In $\mathbb{R}^2$ the best known upper bound of m is $O(nk^{1/3})$ [9], and the best known lower bound for m is $n2^{\Omega(\sqrt{\log k})}$ [20]. In $\mathbb{R}^2$, constructing the k-level takes $O((n+m)\log n)$ time using Edelsbrunner and Welzl's algorithm [13] with the data structure in [3], and it takes $O(n\log n+nk^{1/3})$ expected time with Chan's randomized algorithm [4] which is output insensitive. In the word RAM model, the construction takes $O\left((n+m)\frac{\log n}{\log\log n}\right)$ time [8].

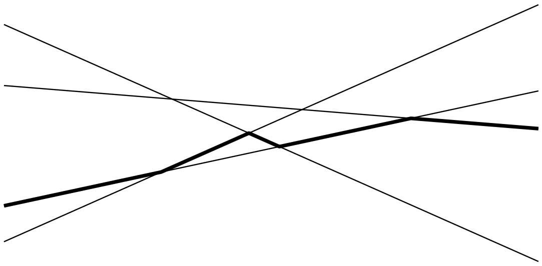

Figure 1: The 1-level of an arrangement in $\mathbb{R}^2$

Let $\mathcal{A}(T)$ be the dual arrangement [1, 11, 14] of S, where T is a set of dual hyperplanes of the points in S. For a hyperplane $\hbar$ determined by d points in S, the major side of $\hbar$ contains at least $\lceil\frac{n-d}{2}\rceil$ points in its interior, and they are either above or below $\hbar$. Let $\hbar^*$ be the dual image of $\hbar$ in $\mathcal{A}(T)$. Then, below or above $\hbar^*$ there are the same number of hyperplanes. We define the major side of $\hbar^*$ as a direction of the x_d-axis along which the ray from $\hbar^*$ intersects at least $\lceil\frac{n-d}{2}\rceil$ hyperplanes. The directions of the major sides of $\hbar$ and $\hbar^*$ are opposite since the relative position between a point and a hyperplane is reversed in the dual space. However, if a point is in the major side of $\hbar$, the dual

[*] School of Computer Science, Carleton University, Ottawa, Ontario K1S 5B6, Canada, dchen4@connect.carleton.ca, morin@scs.carleton.ca.

image of the point (a hyperplane) is on the major side of $\hbar^*$.

In the dual arrangement, we call vertices red if they have major side facing down, blue if the have major side facing up, and purple if they have major side facing both up and down. Then the majority depth of p is equal to the number of purple vertices plus the number of red vertices above p^* plus the number of blue vertices below p^*.

When n is odd, the vertices with level less than $\lceil \frac{n-d}{2} \rceil$ in $\mathcal{A}(T)$ are blue, and the ones with level more than that are red (see Figure 2). For each vertex on the $\lceil \frac{n-d}{2} \rceil$-level, if the convex angle of its two adjacent segments faces up it is blue, and if it faces down it is red.

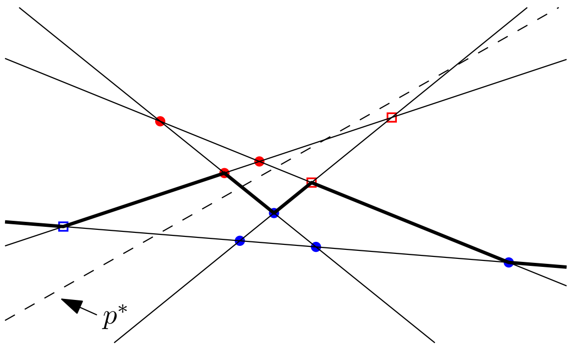

Figure 2: The vertices and major sides when n is odd

When n is even, the situation is a little different. As shown in Figure 3, the vertices with level less than $\lceil \frac{n-d}{2} + 1 \rceil$ are blue, and the ones with level more than $\lceil \frac{n-d}{2} \rceil$ are red. The ones on both of these two levels are purple.

Computing the majority depth of p with respect to S is to count the number of major sides p is in. Since the total number of vertices in a simple arrangement is $\binom{n}{d}$, to compute the majority depth it is sufficient to count the number of vertices in $\mathcal{A}(T)$ whose major side does not contain p^*. This problem involves counting the number of vertices of $\mathcal{A}(T)$ that are contained in a set of polygons whose boundary is determined by p^* and the median level of $\mathcal{A}(T)$. We study this problem in the next section.

3 Counting Vertices

In this section we discuss how to count the vertices of a 2-dimensional arrangement of n line segments confined by a simple polygon (see Figure 4). Since there can be $\Omega(n^2)$ intersections in this arrangement, a sweep line algorithm would take too much time. In the following we discuss a couple of more efficient ways of counting the vertices.

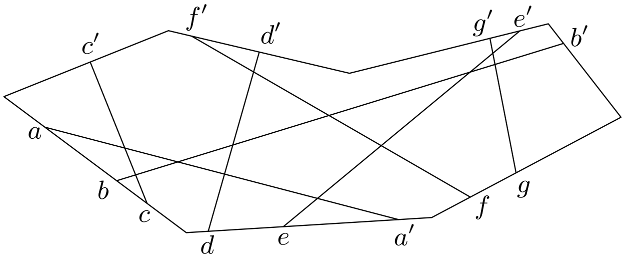

Figure 4: An arrangement in a simple polygon

We first transform the arrangement into a structure as shown in Figure 5, which makes the pattern of intersections clearer to us. In this structure, the polygon is cut at some point and laid flat, and all the line segments are bent into arcs, so that no two arcs intersect twice. The number of intersections in the new structure is the same as that in the original one, because, for any two line segments intersecting in the polygon, the corresponding arcs intersect once.

Notice that for an arc a, any other arc that intersects a has an endpoint laying between the two ends of a. To count the intersections in the new structure, we can use a queue. Starting from one end of the new structure, we add the endpoints of the arcs to the queue. Once the other end of an arc is in the queue, we count the number of endpoints between the two endpoints of the arc, which is the number of intersections the arc contributes. We then remove the two ends from the queue. Upon reaching the other end of the structure, the queue will be empty and all the intersections will be counted. If we implement the queue with an augmented binary tree [7, Chapter 14.1], finding the distance between the two ends of an arc and deleting the other end of the arc takes $O(\log n)$ time, so the number of intersections can be counted in $O(n \log n)$ time.

Another way to count the intersections is to use an array A of size $2n$. Starting from one end of the structure, we walk to the other end. Once we come across a starting end of an arc, we append a 1 to A. Once we come across a finishing end of an arc a, we append a 0 to A. Let the index of the starting end of a in A be i, and that of the finishing end be j. We then set $A[i]$ to 0. Let $sum(k)$ denote $\sum_{l \leq k} A[l]$. The number of the intersections that a contributes is the number of endpoints we came across between $A[i]$ and $A[j]$, which is $sum(j) - sum(i)$. We can compute $sum(k)$ in $O\left(\frac{\log n}{\log \log n}\right)$ time with Dietz's algorithm [10] in the word RAM model. Then counting all intersections takes $O\left(n \frac{\log n}{\log \log n}\right)$ time.

4 The Algorithm

In this section we show how to use the intersection counting structure of the previous section to obtain an efficient algorithm for the majority depth problem in

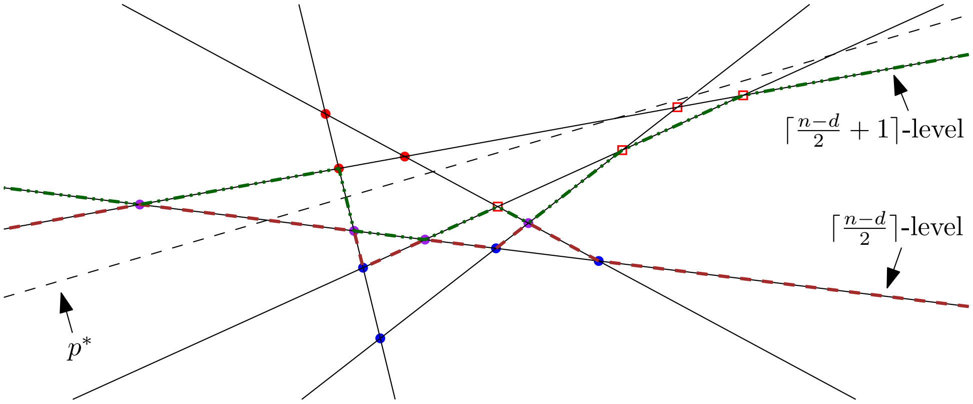
$\lceil \frac{n-d}{2}+1 \rceil$-level
$\lceil \frac{n-d}{2} \rceil$-level
p^*

Figure 3: The vertices and major sides when n is even

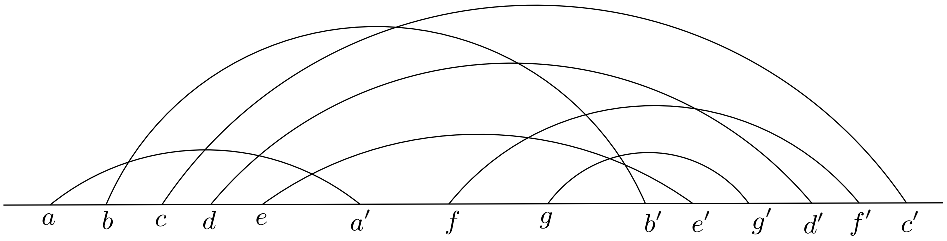

Figure 5: The transformed arrangement

$\mathbb{R}^2$. In the following we will first describe the algorithm when n is odd, then we describe the modifications needed when n is even.

If n is odd, we first compute the median level of the dual arrangement of S and the intersections between it and p^*. If they intersect, p^* splits the levels into sections (as the schematic example shown in Figure 6). Each section along with p^* form a simple polygon except the leftmost and rightmost sections, which form unbounded regions. In order to count the vertices in the unbounded region with the methods in Section 3, we need to find the leftmost and rightmost vertices. Since the extreme points of the set of vertices of $\mathcal{A}(T)$ can be found in $O(n \log n)$ time by sorting the lines by slope [6], we can find those two vertices in $O(n \log n)$ time. Then we can add a vertical line to the left of the leftmost vertex, and one to the right of the rightmost vertex to bound the unbounded region (An example is shown in Figure 7). Now we can count the vertices in each polygon, and the ones on the median level whose major side does not contain p^*.

If n is even we need to compute both the $\frac{n}{2}$-level and $(\frac{n}{2}-1)$-level. The polygons should be formed by part of p^* and the one of the two median levels which is further away from p^* (see the schematic example in Figure 8). In Figure 9 is an example where all regions are bounded. Then we need to count all the vertices in the polygons, and count the vertices that on both those levels since they should be counted twice for the depth of p.

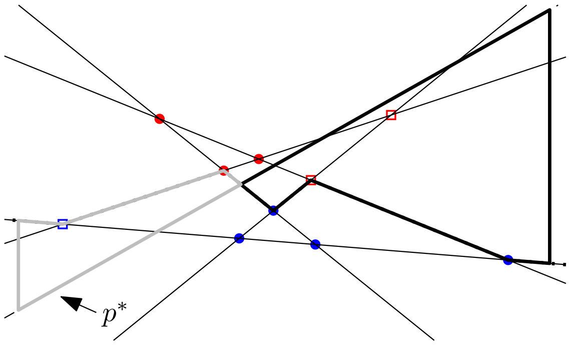

Figure 7: The polygons when n is odd

The number of lines that intersect with p^* is n, and the number of lines that intersect with the two vertical lines is no more than $2n$. Since, in the polygons, each line segment that intersects with the median level has a unique extension on the median level, the total number of line segments that intersect with the median level is no more than m. Each line segment in the polygons has two ends on the boundaries, therefore, the total number of line segments in all the polygons is no more than $3n + m$.

We obtain two different algorithms for computing majority depth depending on which algorithm we use for computing the median level and counting the vertices in a polygon.

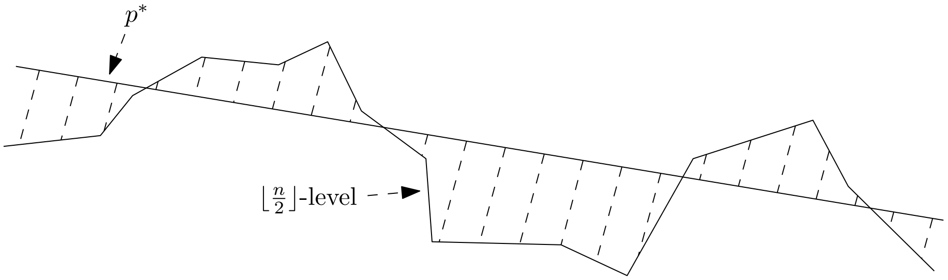

Figure 6: The regions when n is odd

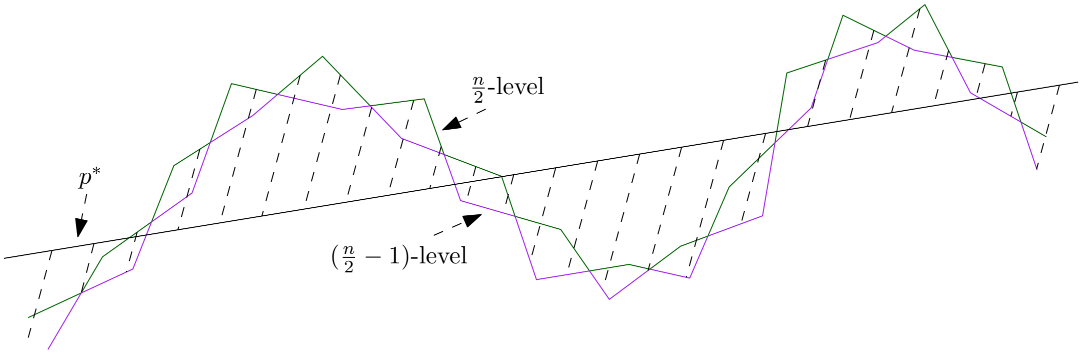

Figure 8: The regions when n is even

Theorem 1 *The majority depth in $\mathbb{R}^2$ can be computed in*

1. *$O((n+m)\log n)$ time with Brodal and Jacob's data structure.*
2. *$O\left((n+m)\frac{\log n}{\log\log n}\right)$ time in the word RAM model.*

The complexity of these algorithms is determined by the value of m, which is the number of vertices of the median level.

5 Conclusion

We have given an algorithm for computing the majority depth of a point p with respect to a set S of n points in $\mathbb{R}^2$. The algorithm's running time is dependent on the size of the median level of the dual arrangement of S. Even without leaving 2 dimensions, this work leaves several open questions:

1. (Depth of a point) Is there an $O(n\log^{O(1)} n)$ time algorithm for computing the majority depth of a point p with respect to a set S of n points in $\mathbb{R}^2$?
2. (Deepest point) Given a set S of n points in $\mathbb{R}^2$, how quickly can we compute a point p whose majority depth (with respect to S) is maximum?
3. (Centerpoint) Determine the maximum value $k = f(n)$ for which the following statement is true: For any set S of n points in $\mathbb{R}^2$, there exists a point $p \in \mathbb{R}^2$ whose majority depth, with respect to S, is at least $\binom{n}{2}/2 + k$.
4. (Faster algorithm in the word RAM model) The related problem of counting inversions has recently been solved in $O(n\sqrt{\log n})$ running time [5]. This unfortunately does not improve our algorithm. Can the factor $\frac{\log n}{\log\log n}$ in the running time of our algorithm be replaced by $\sqrt{\log n}$?

An algorithm for the first problem would have to avoid computing the median level. The second problem is easily solved in $O(n^4)$ time and $O(n^2)$ space by traversing the arrangement of lines through all $\binom{n}{2}$ pairs of points in S using the topological sweep algorithm [12].

References

[1] P. Agarwal and M. Sharir. Arrangements and their applications. In *Handbook of Computational Geometry*, pages 49–119. Elsevier Science Publishers North-Holland, 1998.

[2] G. Aloupis. Geometric measures of data depth. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 2006.

[3] G. Brodal and R. Jacob. Dynamic planar convex hull. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, pages 617–626, 2002.

[4] T. Chan. Remarks on k-level algorithms in the plane. Manuscript, 1999.

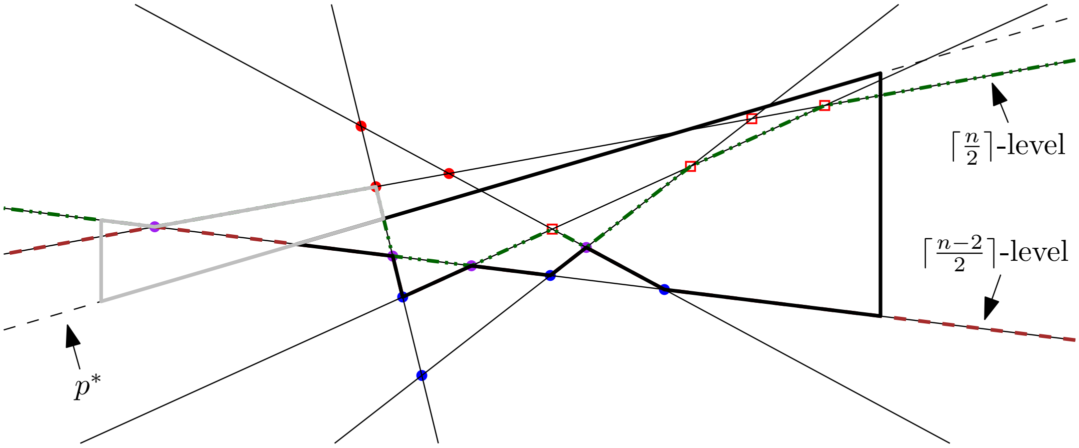
$\lceil \frac{n}{2} \rceil$-level
$\lceil \frac{n-2}{2} \rceil$-level
p^*

Figure 9: The polygons when n is even

[5] T. Chan and M. Pătraşcu. Counting inversions, offline orthogonal range counting, and related problems. In *Proceedings of the 21st ACM/SIAM Symposium on Discrete Algorithms (SODA)*, pages 161–173, 2010.

[6] Y. Ching and D. Lee. Finding the diameter of a set of lines. *Pattern Recognition*, 18(3-4):249–255, 1985.

[7] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, MA USA, 2nd edition, 2001.

[8] E. Demaine and M. Pătraşcu. Tight bounds for dynamic convex hull queries (again). In *Proceedings of the 23rd annual ACM symposium on Computational geometry*, SoCG '07, pages 354–363, New York, NY, USA, 2007. ACM.

[9] T. Dey. Improved bounds for planar k-sets and related problems. *Discrete & Computational Geometry*, 19(3):373–382, 1998.

[10] P. Dietz. Optimal algorithms for list indexing and subset rank. In F. Dehne, J. Sack, and N. Santoro, editors, *Algorithms and Data Structures*, volume 382 of *Lecture Notes in Computer Science*, pages 39–46. Springer Berlin, 1989.

[11] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag, Heidelberg, Germany, 1987.

[12] H. Edelsbrunner and L. Guibas. Topologically sweeping an arrangement. *Journal of Computer and System Sciences*, 38(1):165–194, 1989.

[13] H. Edelsbrunner and E. Welzl. Constructing belts in two-dimensional arrangements with applications. *SIAM Journal on Computing*, 15(1):271–284, 1986.

[14] D. Halperin. Handbook of discrete and computational geometry. chapter 24, pages 529–562. Chapman and Hall / CRC, Boca Raton, FL, USA, 2nd edition, 2004.

[15] R. Liu. On a notion of data depth based on random simplices. *Annals of Statistics*, 18(1):405–414, 1990.

[16] R. Liu and K. Singh. A quality index based on data depth and multivariate rank tests. *Journal of the American Statistical Association*, 88(421):252–260, 1993.

[17] H. Oja. Descriptive statistics for multivariate distributions. *Statistics and Probability Letters*, 1(6):327–332, 1983.

[18] K. Singh. A notion of majority depth. Technical report, Department of Statistics, Rutgers University, 1991.

[19] C. Small. A survey of multidimensional medians. *International Statistical Review*, 58(3):263–277, 1990.

[20] G. Tóth. Point sets with many k-sets. *Discrete & Computational Geometry*, 26(2):187–194, 2001.

[21] J. W. Tukey. Mathematics and the picturing of data. In *Proceedings of the International Congress of Mathematicians: Vancouver*, volume 2, pages 523–531, Montreal, 1975. Canadian Mathematical Congress.