Absolute Approximation of Tukey Depth: Theory and Experiments

Dan Chen

School of Computer Science, Carleton University

Pat Morin

School of Computer Science, Carleton University

Uli Wagner

Institut für Theoretische Informatik

Abstract

A Monte Carlo approximation algorithm for the Tukey depth problem in high dimensions is introduced. The algorithm is a generalization of an algorithm presented by Rousseeuw and Struyf (1998). The performance of this algorithm is studied both analytically and experimentally.

Keywords: Tukey depth, computational geometry

1. Introduction

Tukey depth is also known as *location depth* or *halfspace depth*. Given a finite set S of n points and a point p in $\mathbb{R}^d$, the Tukey depth of p is defined as the minimum number of points of S contained in any closed halfspace with p on its boundary [15, 22]. An equivalent definition is the minimum number of points of S contained in any halfspace which also contains p [4]. This problem is NP-hard if both n and d are parts of the input [16], and it is even hard to approximate [2]. Many different algorithms have been developed to compute the Tukey depth of a point [5, 4, 20]. This problem is equivalent to the *maximum feasible subsystem (MAX FS)* problem [9] which

Email addresses: dchen4@connect.carleton.ca (Dan Chen), morin@scs.carleton.ca (Pat Morin), uli@inf.ethz.ch (Uli Wagner)

Preprint submitted to Elsevier *February 29, 2012*

is a long-standing problem and has been extensively studied [10, Chapter 7]. There are also algorithms for finding a point that maximizes the Tukey depth [6, 17, 18], and Teng [21] showed that testing whether a point does so is coNP-complete in general dimension.

Suppose points in S are in general position (no $d+1$ points of $S \cup \{p\}$ lie on a common hyperplane), an upper bound on the Tukey depth of p can be obtained by selecting any non-trivial vector $v \in \mathbb{R}^d$ and computing the Tukey depth of $p \cdot v$ in the one-dimensional point set

$$S \cdot v = \{x \cdot v : x \in S\}. \tag{$*$}$$

If v is the inner-normal of the boundary of the halfspace $\hbar$ that defines the depth value of p, then

$$\operatorname{depth}(p, S) = \operatorname{depth}(p \cdot v, S \cdot v). \tag{1}$$

In $\mathbb{R}^1$, we rank the points $S \cup \{p\}$ starting with 0 from both ends to the median, then the depth of p is its rank. More generally, given any k-flat f orthogonal to the boundary of $\hbar$, we have

$$\operatorname{depth}(p, S) = \operatorname{depth}(p \cdot f, S \cdot f), \tag{2}$$

where $p \cdot f$ is the orthogonal projection of p onto f, and $S \cdot f$ is the orthogonal projection of S onto f.

1.1. Related Work

Due to the hardness of the Tukey depth problem, algorithms for approximating Tukey depth in low dimensions are of interest. Rousseeuw and Struyf [20] proposed four approximation algorithms. The basic idea is to randomly choose m of four categories of lines: (1) all lines connecting p and a point in S, (2) all lines connecting two points in S, (3) all lines perpendicular to the hyperplanes determined by p and $d-1$ points in S, (4) all lines perpendicular to the hyperplanes determined by d points in S, then project all points onto the lines to solve one-dimensional Tukey depth problems, and take the best result as the approximation. They claimed that the fourth idea worked best.

Wilcox [24] proposed two approximation algorithms. The strategies are similar to those of Rousseux and Struyf. The difference is that, in the first approximation, the points are orthogonally projected onto the lines connecting an affine equivariant measure of location e and a point in S; in the second approximation the points are projected onto all the lines determined by two points in S.

Cuesta-Albertos and Nieto-Reyes [13] proposed a notion of *random Tukey depth* where they project all points onto m randomly chosen vectors, and take the best one-dimensional Tukey depth. They claim this depth is a reasonable approximation of Tukey depth. All the above approximations have no theoretical guarantee of the performance.

Afshani and Chan [1] gave a data structure for Tukey depth queries in 3D. For any constant $\epsilon > 0$, their data structure can preprocess a 3D point set in $O(n \log n)$ expected time into a data structure of size $O(n)$ such that the Tukey depth of any query point q can be approximated in $O(\log n \log\log n)$ time. Here, the approximation is relative; their data structure returns a value y such that

$$(1-\epsilon)\operatorname{depth}(q,S) \leq y \leq (1-\epsilon)^{-1}\operatorname{depth}(q,S) \ .$$

1.2. New Results

In this paper, we analyze the following 2 heuristics for this problem:

1 Randomly select a set Q of $d-1$ points from S. Let π be the unique hyperplane containing $Q \cup \{p\}$, and let v be a vector orthogonal to π. Apply (1) to get an upper bound on depth (p, S).

2 Randomly select a set Q of $d-k$ points from S. Let π be the unique $(d-k)$-flat containing $Q \cup \{p\}$, and let f be a k-flat orthogonal to π. Apply (2) to get an upper bound on depth (p, S).

The first algorithm described above is the third method proposed by Rousseeuw and Struyf. The second algorithm is a generalization of this method. Notice that when we project the points in S to the vector or k-flat, those points in Q do not contribute to the depth of p.

The first algorithm reduces the original problem to a one-dimensional Tukey depth problem, but the second reduces to a k-dimensional Tukey depth problem. The projection of S to a vector takes $O(dn)$ time. Then the first heuristic requires $O(dn)$ time. In the second heuristic, the projection of S to a k-flat takes $O(kdn)$ time, and the k-dimensional Tukey depth problem has the following time complexity:

For $k = 1$, the Tukey depth is easily computed in $O(n)$ time by counting the number of points less than p and the number of points greater than p, and taking the minimum of those 2 quantities.

For $k = 2$, the Tukey depth of p can be computed in $O(n \log n)$ time by sorting the points of S radially about p and scanning this sorted list using two pointers [20].

For $k \geq 3$, the algorithms already become significantly more complicated. When $k = 3$, a brute-force algorithm runs in $O(n^3)$ time, and an algorithm of Chan [7] runs in $O((n + t^2) \log n)$ time, where t is the depth of p.

In the remainder of this paper we analyze how good these upper bounds can be with the following two theorems, which bound the probability that the approximated depth exceeds the true depth by more than σ.

Theorem 1. *Let S be a set of n points in general position in $\mathbb{R}^d$, S' be a subset of $d - 1$ elements chosen at random and without replacement from S, v be the vector perpendicular to the plane containing S' and another point p, σ be an integer such that $0 \leq \sigma \leq \lfloor \frac{n}{d} \rfloor - 1$. Then*

$$\Pr\{\text{depth}\,(p \cdot v, S \cdot v) \leq \text{depth}\,(p, S) + \sigma\} \geq \frac{\binom{\sigma+d-1}{d-1}}{\binom{n}{d-1}}.$$

Theorem 2. *Let S be a set of n points in general position in $\mathbb{R}^d$, S' be a subset of $d - k$ elements chosen at random and without replacement from S, f be the k-flat orthogonal to the $(d - k)$-flat containing S' and another point p, σ be an integer such that $0 \leq \sigma \leq \lfloor \frac{n}{2} \rfloor - 1$. Then*

$$\Pr\{\text{depth}\,(p \cdot f, S \cdot f) \leq \text{depth}\,(p, S) + \sigma\} \geq \frac{2^{d-k}\binom{\sigma+d-k}{d-k}}{(d-k)!\binom{n}{d-k}}.$$

Here is a sketch of the proof of Theorem 1: Under point/hyperplane duality, the selection of v is equivalent to selecting a random vertex in an arrangement of hyperplanes in $d - 1$ dimensions. This selection of v approximates depth (p, S) to within σ provided that the vertex is contained in a particular pseudo-ball of radius σ. Therefore the proof boils down to showing that the number of vertices of an arrangement in a pseudo-ball of radius σ is sufficiently large. In particular, we show that the number of vertices in such a pseudo-ball is at least $\binom{\sigma+d-1}{d-1}$.

The proof of Theorem 2 is similar, except that we lower-bound the number of k flats that intersect a pseudo-ball of radius σ.

The remainder of the paper is organized as follows: In Section 2 we prove lower-bounds on the number of vertices in pseudo-balls and the number of

k-flats that intersect pseudo-balls. In Section 3 we show how these results apply to the analysis of the algorithms for approximating Tukey depth. In Section 4 we give some experimental results for the two algorithms.

2. Arrangements of Hyperplanes

Let H be a set of ℓ 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_1-axis. Without loss of generality, we assume that no hyperplane in H is vertical.

Arrangements.. 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.

Pseudo-distance.. Following Welzl [23], we use δ_H to denote the *pseudo-distance* for pairs of points (relative to H), where $\delta_H(p,q)$ is defined as the number of hyperplanes in H which have p and q on opposite sides. For a point p and an integer σ, we define the pseudo-ball $D_H(p,\sigma)$ as the set of vertices q in $\mathcal{A}(H)$ with $\delta_H(p,q) \leq \sigma$. Our goal in this section is to show that arrangements have big pseudo-balls. In particular, we will prove

Lemma 1. *If H is a set of ℓ hyperplanes in general position in $\mathbb{R}^d$, and σ is an integer, $0 \leq \sigma \leq \ell - d$, then $|D_H(p,\sigma)| \geq \binom{\sigma+d}{d}$ for all points p disjoint from H.*

To prove this lemma, we need to use a result, due to Clarkson [11], on the number of *i-bases* in an arrangement. With this result we can prove a lower bound on the size of $D_H(p,\sigma)$. The following is a review of Clarkson's theorem (with some modifications) on the number of i-bases, which is the main tool used to prove Theorem 1. The difference between this proof and the original is in the definition of i-basis.

Let $\mathcal{P}(H)$ be the convex polytope given by $\mathcal{P}(H) = \cap_{h \in H}(h \cup h^+)$, where h^+ is the open halfspace bounded by h and containing point $(\infty, 0, \ldots, 0)$. Let $G \subset H$, $|G| \geq d$. Then we define $x^*(G)$ as the vertex of $\mathcal{P}(G)$ with lexicographically smallest coordinates. Note that $x^*(G)$ is well defined since $|G| \geq d$ and the hyperplanes in H are in general position. Also note that there exists one subset $B \subset G$ with $|B| = d$ and such that $x^*(B) = x^*(G)$. We call B the *basis* $b(G)$ of G. For any $B \in \binom{H}{d}$, let

$$I_B \equiv \{h \in H \mid b(B \cup \{h\}) \neq B\}$$

be the set of hyperplanes that *violate* $b(B)$. If $|I_B| = i$, B is called an i-basis.

Since any random sample $R \in \binom{H}{r}$, where $d \leq r \leq \ell$, has exactly one basis, we have

$$1 = \sum_{B \in \binom{H}{d}} \Pr\{B = b(R)\} \quad \forall d \leq r \leq \ell. \tag{3}$$

Any $B \in \binom{H}{d}$ is the basis of R if and only if $B \subseteq R$ and R does not contain any element of I_B. If B is an i-basis, the probability that B is the basis of R is $\frac{\binom{\ell-i-d}{r-d}}{\binom{\ell}{r}}$. Let $g'_i(H)$ denote the number of i-bases in the arrangement. Equation (3) can be rewritten as

$$1 = \sum_{0 \leq i \leq \ell-d} \frac{\binom{\ell-i-d}{r-d}}{\binom{\ell}{r}} g'_i(H) \quad \forall d \leq r \leq \ell. \tag{4}$$

Equation (4) gives a system of $l-d+1$ linear equations in $l-d+1$ variables. Solving this system gives

$$g'_i(H) = \binom{i+d-1}{d-1}. \tag{5}$$

For more details see Clarkson [11]. Mulmuley also proved this result with a different method [19].

Proof (of Lemma 1). By a standard projective transformation, we can assume that all hyperplanes in H are below p. An i-basis defines a vertex with distance to p no more than i. We know that the number of i-bases is $\binom{i+d-1}{d-1}$ in $\mathcal{A}(H)$. The number of vertices with distance to p no more than σ is therefore at least

$$\sum_{i=0}^{\sigma} \binom{i+d-1}{d-1} = \binom{\sigma+d}{d}.$$

□

The bound in Lemma 1 is a generalization of the second result of Welzl [23] for the case $d = 2$. It also strengthens the bounds of Chazelle and Welzl [8] for $d \geq 3$. This bound is a lower bound on the number of *$\leq$k-sets*.

Now we develop the tools needed to prove Theorem 2. We define the distance from p to a k-flat f as

$$\delta^k_H(p, f) = \min_{q \in f} \delta_H(p, q).$$

For a point p and an integer σ, we let $D^k_H(p, \sigma)$ denote the set of k-flats f in the arrangement of H with $\delta^k_H(p, f) \leq \sigma$. Notice that $D_H(p, \sigma) = D^0_H(p, \sigma)$.

Proposition 2.1. *For any point p disjoint from H in $\mathbb{R}^d$,*

$$|D^{d-1}_H(p, \sigma)| \geq 2(\sigma + 1) \quad \forall \sigma \in \left\{0, 1, \ldots, \left\lfloor \frac{\ell}{2} \right\rfloor - 1\right\}.$$

Proof. Welzl's proof [23] for $\mathbb{R}^2$ is also valid for $\mathbb{R}^d$. We can always find a line through p that intersects $\lfloor \frac{\ell}{2} \rfloor$ hyperplanes of H on each side of p. □

Lemma 2. *If H is a set of ℓ hyperplanes in general position in $\mathbb{R}^d$, and σ is an integer, $0 \leq \sigma \leq \lfloor \frac{\ell}{2} \rfloor - 1$, then $|D^k_H(p, \sigma)| \geq \frac{2^{d-k}}{(d-k)!}\binom{\sigma+d-k}{d-k}$ for all vertices p disjoint from H.*

Proof. We are going to prove this theorem by induction on d. The proof is inspired by the proof by Welzl in [23]. In $\mathbb{R}^{k+1}$, we have, by Proposition 2.1,

$$|D^k_H(p, \sigma)| \geq 2(\sigma + 1) = \frac{2^{k+1-k}}{(k + 1 - k)!}\binom{\sigma + k + 1 - k}{k + 1 - k}.$$

Assume that $|D^k_H(p, \sigma)| \geq \frac{2^{t-k}}{(t-k)!}\binom{\sigma+t-k}{t-k}$ in $\mathbb{R}^t$, where $t \geq k + 1$. In $\mathbb{R}^{t+1}$, we have at least $2(\sigma + 1)$ t-flats with distance to p no more than σ according to Proposition 2.1. Let h_j be a t-flat with distance of j to p. We know that there are at least two such t-flats according to Proposition 2.1. We also know that there is a point q_j in h_j with $\delta_H(p, q_j) \leq j$. Then any vertices in h_j with distance to q_j no more than $\sigma - j$ have distance to p no more than σ. Since h_j is a space of dimension t, there are at least $\frac{2^{t-k}}{(t-k)!}\binom{\sigma-j+t-k}{t-k}$ such vertices. Since a k-flat is the intersection of $t + 1 - k$ hyperplanes, a vertex can be counted at most $t + 1 - k$ times. Therefore, in $\mathbb{R}^{t+1}$, we have

$$\begin{aligned} |D^k_H(p, \sigma)| &\geq \frac{2}{t + 1 - k} \sum_{j=0}^{\sigma} \frac{2^{t-k}}{(t - k)!}\binom{\sigma - j + t - k}{t - k} \\ &= \frac{2^{t+1-k}}{(t + 1 - k)!}\binom{\sigma + t + 1 - k}{t + 1 - k}. \end{aligned}$$

Hence, in $\mathbb{R}^d$,

$$|D_H^k(p, \sigma)| \geq \frac{2^{d-k}}{(d-k)!} \binom{\sigma + d - k}{d - k}.$$

□

With these two lemmas, we then suggest two approximation algorithms using the two heuristics in Section 1 for the Tukey depth. Our analysis of these algorithms is done by showing that the vector v that minimizes $(*)$ corresponds to a point h_v^* in an arrangement of n hyperplanes in $\mathbb{R}^{d-1}$. Any vertex or k-flat in the arrangement that is "close" to h_v^* will provide a good approximation. Thus, the analysis boils down to showing that there are many vertices or k-flats that are close to h_v^* so that we have a good chance of picking one of them.

3. Approximations for Tukey Depth

In order to relate the hyperplane arrangements studied in Section 2 to the approximation algorithms for Tukey depth, we need to introduce duality [14].

Point/hyperplane duality.. For a point $a = (a_1, a_2, \ldots, a_d)$ in S, its dual image, denoted by a^*, is a hyperplane in T with equation $x_d = a_1x_1 + a_2x_2 + \ldots + a_{d-1}x_{d-1} - a_d$, and for a hyperplane b with equation $x_d = b_1x_1 + b_2x_2 + \ldots + b_{d-1}x_{d-1} - b_d$, its dual image, denoted by b^*, is the point $(b_1, b_2, \ldots, b_d)$. Duality preserves incidences between points and hyperplanes and reverses the above/below relationship. The point a lies on the hyperplane b if and only if b^* lies on a^*; a lies above b if and only if a^* is below b^*. All the hyperplanes through point p in the primal dualize to all the points on the hyperplane p^* in the dual.

The dual arrangement.. Given a set S of n points in $\mathbb{R}^d$, we define the *dual arrangement* $\mathcal{A}(T)$ of S as a set of n hyperplanes, T, that are the duals of the points in S. In the dual arrangement, we say a hyperplane is *vertical* if it contains a line parallel to the x_d-axis.

Duality and Tukey depth.. Finding the Tukey depth of p is equivalent to finding a hyperplane h (with inner-normal v) through p with the fewest points either above or below. In the dual, this is the same as finding a point h_v^* on they hyperplane p^* with the fewest hyperplanes of T either below or above.

The hyperplanes in T divide p^* into cells. Within a cell, the number of hyperplanes above or below any two points is the same. Suppose cell c in T contains the optimal points (h_v^* is a point inside c). For any vertex b^* in $\mathcal{A}(T)$ with $\delta_T(h_v^*, b^*) = \sigma$, the normal vector v_b of its primal image b gives a depth value at most σ more than the optimal depth value (Heuristic 1 in page 3). Similarly, for any k-flat y^* in $\mathcal{A}(T)$ with $\delta_T^k(h_v^*, y^*) = \sigma$, the $(k+1)$-flat f_y orthogonal to its primal image y gives a depth value at most σ more than optimal depth value (Heuristic 2 in page 3).

3.1. Analysis of First Heuristic

Now let us analyze how well the first heuristic works. Sampling $d-1$ points from S is the same as sampling $d-1$ hyperplanes in T which will define a vertex on p^*. Then we only need to consider the $d-1$ dimensional arrangement $\mathcal{A}(T_{p^*})$ restricted to p^*. According to Lemma 1, $|D_{T_{p^*}}(h_v^*, \sigma)| \geq \binom{\sigma+d-1}{d-1}$. Since there are $\binom{n}{d-1}$ vertices on p^*, by one sampling, the probability that we get a depth value with an error no more than σ is at least

$$\frac{\binom{\sigma+d-1}{d-1}}{\binom{n}{d-1}} = \frac{(\sigma+d-1)!(n-d+1)!}{\sigma!n!}. \tag{6}$$

Let $P_\sigma = \frac{(\sigma+d-1)!(n-d+1)!}{\sigma!n!}$. We can repeat this heuristic s times and use the best result as an approximation. The probability that the best depth value with an error more than σ is at most $(1-P_\sigma)^s$. Hence, the probability that we get a depth value with an error no more than σ is at least

$$1-(1-P_\sigma)^s \geq 1-\frac{1}{e} \text{ for } s = \frac{\sigma!n!}{(\sigma+d-1)!(n-d+1)!} \leq \left(\frac{n}{\sigma}\right)^{d-1}. \tag{7}$$

If we set σ to ϵn, where ϵ is a fixed constant, this approximation runs in $O(\epsilon^{1-d}dn)$ time.

3.2. Analysis of Second Heuristic

In the second heuristic, sampling $d-k$ points from S is the same as sampling $d-k$ hyperplanes in T which will define a $(k-1)$-flat on p^*. According to Lemma 2, we have $|D_{T_{p^*}}^{k-1}(h_v^*, \sigma)| \geq \frac{2^{d-k}}{(d-k)!}\binom{\sigma+d-k}{d-k}$. Since there are $\binom{n}{d-k}$ $(k-1)$-flats on p^*, by one sampling, the probability that we get a depth value with an error no more than σ is at least

$$\frac{\frac{2^{d-k}}{(d-k)!}\binom{\sigma+d-k}{d-k}}{\binom{n}{d-k}} = \frac{2^{d-k}(\sigma+d-k)!(n-d+k)!}{(d-k)!\sigma!n!}. \tag{8}$$

Similar to the above analysis, we let $P'_\sigma = \frac{2^{d-k}(\sigma+d-k)!(n-d+k)!}{(d-k)!\sigma!n!}$. Running this heuristic s times, the probability that we get a depth value with an error no more than σ is at least

$$1-(1-P'_\sigma)^s \geq 1-\frac{1}{e} \text{ for } s = \frac{(d-k)!\sigma!n!}{2^{d-k}(\sigma+d-k)!(n-d+k)!} \leq \left(\frac{d-k}{2}\right)^{d-k} \cdot \left(\frac{n}{\sigma}\right)^{d-k}. \tag{9}$$

This approximation needs less samples when d is small, but we need to solve s Tukey depth problems in $\mathbb{R}^k$. For $k = 2$, if we set σ to ϵn, this approximation runs in $O\left(\left(\frac{d-2}{2}\right)^{d-2} \epsilon^{2-d} n \log n\right)$ time.

Our approximation algorithms are comparable to the following simple approximation. For a fixed constant ϵ and a large enough constant c, we make a random sample R of S, where each element of S is selected with probability $\frac{c \log n}{\epsilon n} < 1$. We then compute depth$(p, R)$ with brute-force. With high probability, depth$(p, R) \cdot \frac{\epsilon n}{c \log n}$ is an approximation of depth(p, S) with error no more than ϵn. This approximation runs in $O\left(\left(\epsilon^{-1} c \log n\right)^d\right)$ time. While this is asymptotically faster than our algorithms, $\log^d n$ can be significantly larger than n in many cases. This is the case, for example, with all the data sets used in the next section.

4. Experimental Results

We tested the two approximation algorithms on a Dell Precision 490 workstation with a 2.80 GHz Intel Xeon CPU. For the second approximation, we tested the case of $k = 2$, and the 2-dimensional problems are solved with a scan and sort algorithm [20]. The two algorithms are run s times (as indicated in (7) and (9)) and tested with the 9 data sets listed in Table 1:

The Rand4d data set is randomly generated, and the data items are uniformly distributed in a unit ball. All other data sets are extracted from some data sets in the University of California, Irvine (UCI) Machine Learning Repository (MLR) [3]. The data points in the data sets extracted from UCI MLR are not in general position. Even worse, there are duplicate data points in some data sets. There are no duplicates in Wine4d, Wine5d, Pima4d, Pima5d, and Rand4d. There are a few duplicates in Iris, Auto4d, and Auto5d. There are many duplicates in Yeast, Forest4d, and Forest5d.

The running time of the algorithms on different data sets is given in Table 2. The second approximation runs faster, but it is more sensitive to

Name	Item # (n)	Attrib # (d)	Source
Iris	150	4	UCI MLR.
Wine4d	178	4	UCI MLR. 4 attributes of the Wine data set
Wine5d	178	5	UCI MLR. 5 attributes of the Wine data set
Auto4d	392	4	UCI MLR. 4 attributes of the Auto MPG data set
Auto5d	392	5	UCI MLR. 5 attributes of the Auto MPG data set
Rand4d	500	4	Randomly generated
Forest4d	517	4	UCI MLR. 4 attributes of the Forest Fires data set [12]
Forest5d	517	5	UCI MLR. 5 attributes of the Forest Fires data set [12]
Pima4d	768	4	UCI MLR. 4 attributes of the Pima Indians Diabetes data set
Pima5d	768	5	UCI MLR. 5 attributes of the Pima Indians Diabetes data set
Yeast4d	1484	4	UCI MLR. 4 attributes of the Yeast data set

Table 1: The data sets

rounding error. In order to generate a 2d problem in the second approximation, we first find 2 perpendicular vectors in the 2-flat orthogonal to the $(d-2)$-flat containing the $d-2$ sampling points and p, then project all points in the data set onto the 2 vectors. The values are used as the coordinates of points in the 2-dimensional space. This projection and the sorting of the 2-dimensional points bring rounding errors. To overcome this problem, exact arithmetic is applied on the Iris data set with GMP (GNU Multiple Precision Library). GMP slows down the algorithm dramatically, hence it is not practical to use it on larger data sets.

The true depth values are computed with the binary search idea in [9] which requires solving a series of mixed integer program. It takes a long time and a large amount of memory to solve the integer programs. Many instances can not be solved due to time and memory limitations. The time required to solve integer programs is output-sensitive, so that problems with larger depth values take longer. For example, we need a few hours to solve a problem with depth 10 in Pima5d. On the other hand, the approximation algorithms do not have this sensitivity. They take roughly the same time to

Data Set	σ value	Algorithm	Running time	Max error[1]	Average error[1]
Iris	2	approx 1	50s(GMP)	2	0.309
		approx 2	7s(GMP)	2	0.258
Wine4d	2	approx 1	2s	2	0.372
		approx 2	1s	2	0.326
Wine5d	2	approx 1	70s	3	0.223
		approx 2	8s	3	0.086
Auto4d	2	approx 1	31s	1	0.213
		approx 2	2s	2	0.213
Auto5d	2	approx 1	2400s	1	0.164
		approx 2	187s	1	0.071
Rand4d	2	approx 1	77s	1	0.186
		approx 2	3s	2	0.169
Forest4d	2	approx 1	87s	2	0.309
		approx 2	4s	2	0.136
Forest5d	3	approx 1	3880s	2	0.319[2]
		approx 2	287s	1	0.088[3]
Pima4d	2	approx 1	387s	2	0.299
		approx 2	12s	1	-1.031[4]
Pima5d	4	approx 1	12350s	2	0.708
		approx 2	815s	2	-0.333[5]
Yeast4d	3	approx 1	2400s	1	0.571
		approx 2	56s	1	0.429

[1] Some points are not tested because we do not know the real depth of them.
[2] 2 depth values are smaller than the real ones (due to rounding errors).
[3] 7 depth values are smaller than the real ones (due to rounding errors).
[4] 23 depth values are smaller than the real ones (due to rounding errors).
[5] 14 depth values are smaller than the real ones (due to rounding errors).

Table 2: The performance of the algorithms

solve all the problems in the same data set.

For smaller data sets, the tests were run with the absolute error σ set to 2. However, in the vast majority of cases (at least those in which the true depth can be computed exactly), both approximation algorithms computed the depth correctly with no error. In a small number of cases the error is 1 or 2. The second approximation gave less average error.

5. Concluding Remarks

In this paper, we have

1. given a rigorous theoretical analysis of the algorithm of Rousseeuw and Struyf [20] that explains their experimental results;
2. generalized the algorithm of Rousseeuw and Struyf to solve k-dimensional subproblems. Using value $k = 2$ gives a substantial improvement in running time while providing the same approximation; and
3. done extensive testing of these algorithms on real and synthetic data sets. This testing shows that the algorithms are indeed fast and that, in most cases, they compute the exact Tukey depth, and make an error of 1 or 2 (when σ is set to 2) rather infrequently.

These algorithms are simple, easy to implement, and our results show that, as well as having theoretical guarantees, they work well in practice.

[1] P. Afshani and T. M. Chan. On approximate range counting and depth. In *SCG '07: Proceedings of the Twenty-Third Annual Symposium on Computational Geometry*, pages 337–343, New York, NY, USA, 2007. ACM.

[2] E. Amaldi and V. Kann. The complexity and approximability of finding maximum feasible subsystems of linear relations. *Theoretical Computer Science*, 147(1–2):181–210, 1995.

[3] A. Asuncion and D. J. Newman. UCI machine learning repository, 2007.

[4] D. Bremner, D. Chen, J. Iacono, S. Langerman, and P. Morin. Output-sensitive algorithms for Tukey depth and related problems. *Statistics and Computing*, 18:259–266, 2008.

[5] D. Bremner, K. Fukuda, and V. Rosta. Primal dual algorithms for data depth. In *Data Depth: Robust Multivariate Analysis, Computational Geometry, and Applications*, AMS DIMACS Book Series, 2006.

[6] T. M. Chan. An optimal randomized algorithm for maximum tukey depth. In *SODA '04: Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 430–436, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

[7] T. M. Chan. Low-dimensional linear programming with violations. *SIAM Journal on Computing*, 34(4):879–893, 2005.

[8] B. Chazelle and E. Welzl. Quasi-optimal range searching in spaces of finite VC-dimension. *Discrete Comput. Geom.*, 4(5):467–489, 1989.

[9] D. Chen. A branch and cut algorithm for the halfspace depth problem. Master's thesis, Faculty of Computer Science, University of New Brunswick, Fredericton, Canada, 2007.

[10] J. W. Chinneck. *Feasibility and Infeasibility in Optimization: Algorithms and Computational Methods*, volume 118 of *International Series in Operations Research and Management Sciences*. Springer, New York, USA, 2008.

[11] K. L. Clarkson. A bound on local minima of arrangements that implies the upper bound theorem. *Discrete and Computational Geometry*, 10:427–233, 1993.

[12] P. Cortez and A. Morais. A data mining approach to predict forest fires using meteorological data. In J. Neves, M. F. Santos, and Machado J., editors, *New Trends in Artificial Intelligence, Proceedings of the 13th EPIA 2007 - Portuguese Conference on Artificial Intelligence*, pages 512–523. Springer, 2007.

[13] J. A. Cuesta-Albertos and A. Nieto-Reyes. The random Tukey depth. *Computational Statistics and Data Analysis*, 52(11):4979–4988, 2008.

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

[15] J. L. Hodges. A bivariate sign test. *The Annals of Mathematical Statistics*, 26(3):523–527, 1955.

[16] D. S. Johnson and F. P. Preparata. The densest hemisphere problem. *Theoretical Computer Science*, 6(1):93–107, 1978.

[17] S. Langerman and W. Steiger. Optimization in arrangements. In *Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science*, volume 2607, pages 50–61, London, UK, 2003. Springer-Verlag.

[18] J. Matoušek. Computing the center of planar point sets. In J.E. Goodman, R. Pollack, and W. Steiger, editors, *Computational Geometry: Papers from the Special Year*, pages 221–230. AMS, Providence, 1991.

[19] K. Mulmuley. Dehn-Sommerville relations, upper bound theorem, and levels in arrangements. In *SCG '93: Proceedings of the Ninth Annual Symposium on Computational Geometry*, pages 240–246, New York, NY, USA, 1993. ACM.

[20] P. J. Rousseeuw and A. Struyf. Computing location depth and regression depth in higher dimensions. *Statistics and Computing*, 8(3):193–203, 1998.

[21] S. Teng. *Points, Spheres, and Separators: A Unified Geometric Approach to Graph Partitioning*. PhD thesis, School of Computer Science, Carnegie Mellon University, 1991.

[22] 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.

[23] E. Welzl. On spanning trees with low crossing numbers. In B. Monien and T. Ottmann, editors, *Data Structures and Efficient Algorithms: Final Report on the DFG Special Joint Initiative, B. Monien and Th. Ottmann (Eds.), LNCS 594*, Lecture Notes in Computer Science, pages 233–249. Springer, London, 1992.

[24] R. Wilcox. Approximating Tukey's depth. *Communications in Statistics - Simulation and Computation*, 32(4):977–985, 2003.