Yugoslav Journal of Operations Research
28 (2018), Number 4, 475–499
DOI: https://doi.org/10.2298/YJOR180520022B

APPROXIMATE CALCULATION OF TUKEY'S DEPTH AND MEDIAN WITH HIGH-DIMENSIONAL DATA

Milica BOGIĆEVIĆ
School of Electrical Engineering, University of Belgrade, Belgrade, Serbia
antomripmuk@yahoo.com

Milan MERKLE
School of Electrical Engineering, University of Belgrade, Belgrade, Serbia
emerkle@etf.rs

Received: May 2018 / Accepted: August 2018

Abstract: We present a new fast approximate algorithm for Tukey (halfspace) depth level sets and its implementation-ABCDepth. Given a d-dimensional data set for any $d \geq 1$, the algorithm is based on a representation of level sets as intersections of balls in $\mathbb{R}^d$. Our approach does not need calculations of projections of sample points to directions. This novel idea enables calculations of approximate level sets in very high dimensions with complexity that is linear in d, which provides a great advantage over all other approximate algorithms. Using different versions of this algorithm, we demonstrate approximate calculations of the deepest set of points ("Tukey median") and Tukey's depth of a sample point or out-of-sample point, all with a linear in d complexity. An additional theoretical advantage of this approach is that the data points are not assumed to be in "general position". Examples with real and synthetic data show that the executing time of the algorithm in all mentioned versions in high dimensions is much smaller than the time of other implemented algorithms. Also, our algorithms can be used with thousands of multidimensional observations.

Keywords: Big Data, Multivariate Medians, Depth Functions, Computing Tukey's Depth.

MSC: 62G15, 68U05.

1. INTRODUCTION

Although this paper is about multivariate medians and related notions, for completeness and understanding rationale of multivariate setup, we start from the univariate case. In terms of probability distributions, let X be a random variable and let $\mu = \mu_X$ be the corresponding distribution, i.e., a probability measure on $(\mathbb{R}, \mathcal{B})$ so that $P(X \leq x) = \mu\{(-\infty, x]\}$. For univariate case, a median of X (or a median of μ_X) is any number m such that $P(X \leq m) \geq 1/2$ and $P(X \geq m) \geq 1/2$. In terms of data sets, this property means that to reach any median point from the outside of the data set, we have to pass at least $1/2$ of data points, so this is the deepest point within the data set. With respect to this definition, we can define the depth of any point $x \in \mathbb{R}$ as

$$D(x, \mu) = \min\{P(X \leq x), P(X \geq x)\} = \min\{\mu((-\infty, x]), \mu([x, +\infty))\}. \quad (1)$$

The set of all median points $\{\text{Med}\mu\}$ is a non-empty compact interval (can be a singleton). As shown in [18],

$$\{\text{Med}\mu\} = \bigcap_{\text{J=[a,b]: } \mu(\text{J})>1/2} \text{J}, \quad (2)$$

and (2) can be taken for an alternative (equivalent) definition of univariate median set. In $\mathbb{R}^d$ with $d > 1$, there are quite a few different concepts of depth and medians (see for example [21], [28], [33]). In this paper we propose an algorithm for halfspace depth (Tukey's depth, [31]), which is based on extension and generalization of (2) to $\mathbb{R}^d$ with balls in place of intervals as in [19].

The rest of the paper is organized as follows. Section 2 deals with a theoretical background of the algorithm in a broad sense. In Section 3 we present an approximate algorithm for finding Tukey median as well as versions of the same algorithm for finding Tukey depth of a sample point, the depth of out-of-sample point, and for data contours. We also provide a derivation of complexity for each version of the algorithm and present examples. Section 4 provides a comparison with several other algorithms in terms of performances.

2. THEORETICAL BACKGROUND: DEPTH FUNCTIONS BASED ON FAMILIES OF CONVEX SETS

Definition 2.1. *Let $\mathcal{V}$ be a family of convex sets in $\mathbb{R}^d$, $d \geq 1$, such that: (i) $\mathcal{V}$ is closed under translations and (ii) for every ball $B \in \mathbb{R}^d$ there exists a set $V \in \mathcal{V}$ such that $B \subset V$. Let $\mathcal{U}$ be the collection of complements of sets in $\mathcal{V}$. For a given probability measure μ on $\mathbb{R}^d$, let us define*

$$D(x; \mu, \mathcal{V}) = \inf\{\mu(U) \mid x \in U \in \mathcal{U}\} = 1 - \sup\{\mu(V) \mid V \in \mathcal{V},\ x \in V'\} \quad (3)$$

The function $x \mapsto D(x; \mu, \mathcal{V})$ will be called a depth function based on the family $\mathcal{V}$.

Remark 2.1. Definition 2.1 is a special case of Type D depth functions as defined in [33] which can be obtained by generalizations of (1) to higher dimensions. The conditions stated in [19] that provide a desirable behavior of the depth function are satisfied in this special case, with additional requirements that sets in $\mathcal{V}$ are closed or compact. For instance, taking $\mathcal{V}$ to be the family of all "boxes" with sides parallel to coordinate hyper-planes yields the deepest points which coincide with the coordinate-wise median. It is easy to see that in this case, regardless of the measure μ, there exists at least one point $x \in \mathbb{R}^d$ with $D(x; \mu, \mathcal{V}) \geq \frac{1}{2}$. According to the next theorem [19, Theorem 4.1], in general case of arbitrary $\mathcal{V}$, the maximal depth can't be smaller than $1/(d+1)$.

Theorem 2.1. *Let $\mathcal{V}$ be any non-empty family of compact convex subsets of $\mathbb{R}^d$ satisfying the conditions as in Definition 2.1. Then for any probability measure μ on $\mathbb{R}^d$ there exists a point $x \in \mathbb{R}^d$ such that $D(x; \mu, \mathcal{V}) \geq \frac{1}{d+1}$.*

The set of points with maximal depth is called *the center of distribution*, denoted as $C(\mu, \mathcal{V})$. In general, one can observe *level sets* (or *depth regions* or *depth-trimmed regions*) of level α defined as

$$S_\alpha = S_\alpha(\mu, \mathcal{V}) := \{x \in \mathbb{R}^d \mid D(x; \mu, \mathcal{V}) \geq \alpha\}. \tag{4}$$

Clearly, if $\alpha_1 < \alpha_2$, then $S_{\alpha_1} \supseteq S_{\alpha_2}$ and $S_\alpha = \emptyset$ for $\alpha > \alpha_m$, where α_m is the maximal depth for a given probability measure μ.

The borders of depth level sets are called *depth contours* (in two dimensions) or *depth surfaces* in general. Depth surfaces are of interest in multivariate statistical inference (see [11, 12, 32, 25]). Although (4) suggests that in order to describe level sets we need to calculate depth functions, there is another way, as the next result shows ([9], [33] and Theorem 2.2. in [19]).

Theorem 2.2. *Let $D(x; \mu, \mathcal{V})$ be defined for $x \in \mathbb{R}^d$ as in Definition 2.1. Then for any $\alpha \in (0, 1]$*

$$S_\alpha(\mu, \mathcal{V}) = \bigcap_{V \in \mathcal{V}, \mu(V) > 1-\alpha} V. \tag{5}$$

From (5) it follows that the center of a distribution is the smallest non-empty level set, or equivalently,

$$C(\mu, \mathcal{V}) = \bigcap_{\alpha: S_\alpha \neq \emptyset} S_\alpha(\mu, \mathcal{V}) \tag{6}$$

Since sets in $\mathcal{V}$ are convex, the level sets are also convex. From (4) and (5) we can see that the depth function can be uniquely reconstructed starting from level sets as follows.

For given μ and $\mathcal{V}$, let S_α, $\alpha \geq 0$ be defined as in (5), with $S_\alpha = \emptyset$ for $\alpha > 1$. Then the function $D : \mathbb{R}^d \mapsto [0, 1]$ defined by

$$D(x) = h \quad \Longleftrightarrow \quad x \in S_\alpha, \alpha \leq h \quad \text{and} \quad x \notin S_\alpha, \alpha > h \tag{7}$$

is the unique depth function such that (4) holds.

The algorithm that we propose primary finds approximations to level sets based on formula (5), and then finds (approximate) depth via Corollary 2. The algorithm will be demonstrated in the case of half-space depth, which is described in the next section.

The scientific interest in algorithms for Tukey's depth is shared between Statistics (robust estimation of location parameters, clustering, classification, outlier detection, especially with big data in high dimensions) and Computational geometry (as a challenge).

3. ABCDEPTH ALGORITHM FOR TUKEY DEPTH: IMPLEMENTATION AND THE OUTPUT

The most prominent representative of type D depth functions is Tukey's (or halfspace) depth, which is defined by (3) with $\mathcal{U}$ being a family of all open half-spaces, and $\mathcal{V}$ a family of closed half-spaces, as in the following definition.

$$D(x;\mu) = \inf\{\mu(H) \mid x \in H \in \mathcal{H}\}, \tag{8}$$

where $\mathcal{H}$ is the family of all open half-spaces. In this section, we consider only half-space depth, so we use the notation $D(x,\mu)$ instead of $D(x;\mu,\mathcal{V})$.

The idea traces back to J. W. Tukey's lecture notes [30] and the conference paper [31]. It was first formalized in D. L. Donoho's Ph. D. qualifying paper [9] of 1982, a technical report [10] and the 1992's paper [11].

One depth function can be defined based on different families $\mathcal{V}$. We say that families $\mathcal{V}_1$ and $\mathcal{V}_2$ are depth-equivalent if $D(x;\mu,\mathcal{V}_1) = D(x;\mu,\mathcal{V}_2)$ for all $x \in \mathbb{R}^d$ and all probability measures μ. Sufficient conditions for depth-equivalence are given in [19, Theorem 2.1], where it was shown that in the case of half-space depth the following families are depth-equivalent: a) Family of all open half-spaces; b) all closed half-spaces; c) all convex sets; d) all compact convex sets; e) all closed or open balls.

For determining level sets we choose closed balls, and so we can define $\mathcal{V}$ as a set of all closed balls (hyper-spheres) and level sets can be described as

$$S_\alpha(\mu,\mathcal{V}) = \bigcap_{B\in\mathcal{V},\mu(B)>1-\alpha} B, \tag{9}$$

instead of using the classical approach based on half-spaces. The advantage of the formula (9) over the intersections of half-spaces is obvious if we recall that a ball B is already the intersection of all tangent spaces that contain B.

From now on, we consider only the case where the underlying probability measure μ is derived from a given data set.

3.1. The sample version

In the data setup with a sample $x_1, \ldots, x_n$ (allowing repetitions) and the counting measure

$$\mu(A) = \frac{\#\{x_i : \ x_i \in A\}}{n}, \tag{10}$$

it is a common practice to express the depth as an integer defined as

$$D(x) = \min\{\#\{x_i : \ x_i \in H\} \mid x \in H \in \mathcal{H}\}, \qquad x \in \mathbb{R}^d, \tag{11}$$

whereas the depth in terms of probability is $D(x)/n$. As it was first noticed by Donoho [9], the depth can be expressed via one-dimensional projections to directions determined by normal vectors of hyperplanes that are borders of half-spaces (with $d = 2$ the borders are straight lines).

$$D(x) = \min_{\|u\|=1} \#\{x_i : \langle u, x_i\rangle \leq \langle u, x\rangle\}, \tag{12}$$

where $\langle \cdot, \cdot \rangle$ is the inner product of vectors in $\mathbb{R}^d$.

The level set in the sample version with $\alpha = k/n$ is defined by

$$S_{\frac{k}{n}} = \{x \in \mathbb{R}^d \mid D(x) \geq \lfloor n(1-\alpha)+1 \rfloor\} = \{x \in \mathbb{R}^d \mid D(x) \geq n-k+1\}, \tag{13}$$

and (9) becomes

$$S_{\frac{k}{n}} = \bigcap_{B \in \mathcal{V}, \#\{x_i : \ x_i \in B\} \geq n-k+1} B, \tag{14}$$

where $\mathcal{V}$ is a family of all closed balls in $\mathbb{R}^d$.

All so far implemented algorithms (both exact and approximate) are based on calculation of the depth from the formula (12) or its variations. The approximate algorithm that we propose uses a completely new approach: we start with a discrete approximation to level sets using formula (9), and then we calculate the depth in conjunction with the formula (7). In the next subsections we present details of the approximations.

3.2. First approximation: finite intersection.

For a fixed sample size n and a fixed k, for simplicity we write S instead of $S_{\frac{k}{n}}$. So, for fixed n and k, S is an exact (unknown) level set as in (13) and (14). Let us choose M points to be centers of balls. In most of examples in this paper, we choose the points from the sample ($M = n$), and then add new points at random if needed. The radius of each ball B_i with the center at c_i is equal to the $(n-k+1)$th

smallest distance between c_i and the points in the sample set $\{x_1, \ldots, x_n\}$. In this way, we end up with the first approximation of the level set $S = S_{\frac{k}{n}}$:

$$\hat{S}_M := \bigcap_{i=1}^{M} B_i \tag{15}$$

It is natural to assume that if we want to intersect more than M balls, then we just add new balls to the existing intersection, hence the sets $\hat{S}_M$ are nested and

$$\hat{S}_M \supseteq \hat{S}_{M+1} \supset S, \qquad M = 1, 2, \ldots \tag{16}$$

As shown in [1, Lemma 2], to decide whether or not the intersection in (15) is empty, it takes $M \cdot 2^{O(d^2)}$ time, so the exact approach is not feasible. On the other hand, for a given point $s \in \mathbb{R}^d$ it is easy to decide whether or not it belongs to $\hat{S}$ as defined in (15). This observation leads to the second approximation as follows.

3.3. Second approximation: a discrete set of points.

Let $A_n = \{x_1, \ldots, x_n\}$ (set of sample points), and let $D \supset A_n$ be a convex domain in $\mathbb{R}^d$. For $N = n+1, n+2, \ldots$, let A_N be a set obtained from A_{N-1} by randomly adding one point from D. Then we have

$$A_{N+1} \supset A_N, \quad N = n, n+1, n+2, \ldots, \tag{17}$$

and every set A_N contains all sample points. We call sets A_N $(N \geq n+1)$ *augmented data sets*. Points in $A_N \setminus A_n$ will be called *artificial points* and their role is to "shed light" on a depth region via discrete approximation as follows:

$$\hat{\hat{S}}_{M,N} := A_N \cap \hat{S}_M = \{a \in A_N \mid a \in \hat{S}_M, \} \tag{18}$$

where $\hat{S}_M$ is defined in (15). Let us note that A_N does not depend on k, so the same A_N can be used in approximation of all depth regions.

From (17) and (18), it follows that

$$\hat{\hat{S}}_{M,N} \subseteq \hat{\hat{S}}_{M,N+1} \subset \hat{S}_M \tag{19}$$

Therefore, for a fixed M, the sets $\hat{\hat{S}}_{M,N}$ approximate $\hat{S}_M$ from inside, obviously with accuracy which increases with N. In order to make a contour, we can construct a convex hull of $\hat{\hat{S}}_\alpha$ using *QuickHull* algorithm, for example. The relations in (19) remain true with Conv $(\hat{\hat{S}})$ in place of $\hat{\hat{S}}$. Due to convexity of $\hat{S}_M$, the approximation with Conv $(\hat{\hat{S}})$ would be better, but then the complexity would be too high for really high dimensions. As an alternative, we use $\hat{\hat{S}}_{M,N}$ as the final approximation to the true level set $S = S_{\frac{k}{n}}$.

3.4. More about artificial points

The simplest way to implement the procedure from 3.2 and 3.3 is to take $M = N = n$, and to use sample points for centers of balls and also for finding $\hat{\hat{S}}$ in (18). The basic Algorithm 1 of the subsection 3.5 is presented in that setup. However, in some cases this approach does not work, regardless of the sample size. As an example, consider a uniform distribution in the region bounded by circles $x^2 + y^2 = r_i^2$, $r_1 = 1$ and $r_2 = 2$. It is easy to prove (see also [12]) that the depth monotonically increases from 0 outside of the larger circle, to $1/2$ at the origin, which is the true and unique median. With a sample from this distribution, we will not have data points inside the inner circle, and we can not identify the median in the way proposed above.

In similar cases and whenever we have sparse data or small sample size n, we can still visually identify depth regions and center simply by adding artificial points to the data set. Let the data set contain points $x_1, \ldots, x_n$ and let $x_{n+1}, \ldots, x_N$ be points chosen from uniform distribution in some convex domain that contains the whole data set. Then we use augmented data set (all N points) in (18), but n in formulas (10) and (14) remains to be the cardinality of the original data set.

Figure 1 shows the output of ABCDepth algorithm in the ring example above. By adding artificial data points, we are able to obtain an approximate position of the Tukey's median.

As another example, let us consider a triangle with vertices $A(0, 1)$, $B(-1, 0)$ and $C(1, 0)$. Assuming that A, B, C are sample points, all points in the interior and on the border of ABC triangle have depth $1/3$, so the depth reaches its maximum value at $1/3$. Since the original data set contains only 3 points, by adding artificial data and applying ABCDepth algorithm we can visualize the Tukey's median set as shown in Figure 2.

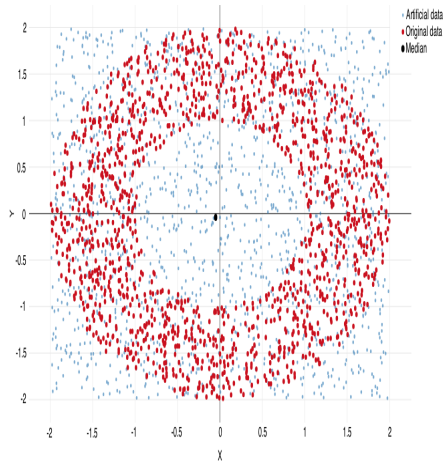

Figure 1: A sample from uniform distribution in a ring (red): Tukey's median (black) found with the aid of artificial points (blue).

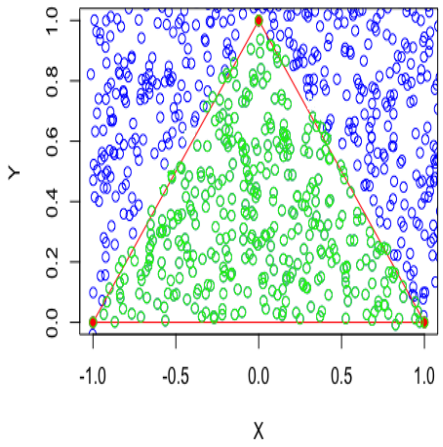

Figure 2: Tukey's median set of red triangle represented as triangle itself and green points inside of the triangle. Blue points are artificial data.

In the rest of this section, we describe the details of implementation of the

approximate algorithm for finding Tukey's median, as well as versions of the same algorithm for finding Tukey depth of a sample point, the depth of out-of-sample point, and for data contours. For simplicity, in the rest of the paper we use notation S_α instead of $\hat{\hat{S}}_\alpha$ unless explicitly noted otherwise.

3.5. Implementation: Algorithm 1 for finding deepest points (Tukey's median)

Phase 1 In order to execute the calculation in (14), constructing balls for intersection is the first step. Each ball is defined by $\lfloor n(1-\alpha)+1 \rfloor$ nearest points to its center, so at the beginning of this phase, we calculate Euclidean inter-distances. Distances are stored as a triangular matrix in a *list of lists* structure, where i-th list $(i = 1, ..., n-1)$ contains distances $d_{i+1,j}, j = 1, \ldots, i$. This part of the implementation is described in lines $1-6$ of Algorithm 1.

Phase 2 In this phase, ABCDepth sorts distances for each point (center) and populates a *hashmap* structure, where the *key* is a center of a ball, and *value* is a *list* with $\lfloor n(1-\alpha)+1 \rfloor$ nearest points. This part of algorithm is presented in lines $7-10$ of Algorithm 1.

Phase 3 Now, we intersect balls iteratively and in each iteration, α is increased by $\frac{1}{n}$. Since this algorithm is meant to find the deepest location, there is no need to start with the minimal value of $\alpha = \frac{1}{n}$; due to Theorem 2.1, we set the initial value of α to be $\frac{1}{d+1}$. Balls intersections are shown on Algorithm 1, lines $11-18$.

If the input set is sparse, ABCDepth optionally creates an augmented data set of total size N as explained on page 480 and demonstrated on Figures 1 and 2. The rest of the algorithm takes three phases which we described above.

The initial version of ABCDepth algorithm was presented in [3].

Data: Original data, $X_n = (\boldsymbol{x_1}, \boldsymbol{x_1}, ..., \boldsymbol{x_n}) \in \mathbb{R}^{d \times n}$
Result: List of level sets, $S = \{S_{\alpha_1}, S_{\alpha_2}, ..., S_{\alpha_m}\}$, where S_{α_m} represents a Tukey median
`/* Note:` S_α `here means` $\hat{S}_\alpha$ `*/`

1 **for** $i \leftarrow 2$ ***to*** n **do**
2 **for** $j \leftarrow 1$ ***to*** $i-1$ **do**
3 Calculate Euclidean distance between point $\boldsymbol{x_i}$ and point $\boldsymbol{x_j}$;
4 Add distance to the list of lists ;
5 **end**
6 **end**
7 **for** $i \leftarrow 1$ ***to*** n **do**
8 Sort distances for point $\boldsymbol{x_i}$;
9 Populate structure with balls ;
10 **end**
`/* Iteration Phase */`
11 $size = n$, $\alpha_1 = \frac{1}{d+1}$, $k = 1$;
12 **while** $size > 1$ **do**
13 $S_{\alpha_k} = \{\bigcap_j^n B_j, |B_j| = \lfloor n(1-\alpha_k)+1 \rfloor\}$;
14 $size = |S_{\alpha_k}|$;
15 $\alpha_{k+1} = \alpha_k + \frac{1}{n}$;
16 Add S_{α_k} to S ;
17 $k = k + 1$;
18 **end**

Algorithm 1: Calculating Tukey median.

3.5.1. Complexity

Theorem 3.1. *ABCDepth algorithm for finding approximate Tukey median has order of* $O((d+k)n^2+n^2 \log n)$ *time complexity, where* k *is the number of iterations in the iteration phase.*

Proof. To prove this theorem we use the pseudocode of Algorithm 1. Lines 1-6 calculate Euclidean inter-distances of points. The first *for* loop (line 1) takes all n points, so its complexity is $O(n)$. Since there is no need to calculate $d(x_i, x_i)$ or $d(x_j, x_i)$ if it is already calculated, the second for-loop (line 2) runs in $O(\frac{n-1}{2})$ time. Finally, calculation of Euclidean distance takes $O(d)$ time. The overall complexity for lines 1-6 is:

$$O(\frac{nd(n-1)}{2}) \sim O(dn^2) \tag{20}$$

Iterating through the *list of lists* obtained in lines 1-6, the first *for* loop (line 7) runs in $O(n)$ time. For sorting the distances per each point, we use *quicksort* algorithm that takes $O(n \log n)$ comparisons to sort n points [16]. Structure

populating takes $O(1)$ time. Hence, this part of the algorithm has complexity of:

$$O(n^2 \log n). \tag{21}$$

In the last phase (lines 11-18), algorithm calculates level sets by intersecting balls constructed in the previous steps. In every iteration (line 12), all n balls that contain $\lfloor n(1-\alpha_k)+1 \rfloor$ are intersected (line 13). The parameter k can be considered as a number of iterations, i.e. it counts how many times the algorithm enters in *while* loop. Each intersection has the complexity of $O(\lfloor n(1-\alpha_k)+1 \rfloor) \sim O(n)$ due to the property of the hash-based data structure we use (see for example [8]). We can conclude that the iteration phase has complexity of:

$$O(kn^2). \tag{22}$$

From (20), (21), and (22),

$$O(dn^2) + O(n^2 \log n) + O(kn^2) \sim O((d+k)n^2 + n^2 \log n), \tag{23}$$

which ends the proof. □

Remark 3.1. 1° From the relations between S_α, $\hat{S}_\alpha$, and $\hat{\hat{S}}_\alpha$ (in notations as in 3.1, page 479), it follows that the maximal approximative depth for a given point can not be greater than its exact depth.

2° Under the assumption that data points are in the general position, the exact sample maximal depth is $\alpha_m = \frac{m}{n}$, where m is not greater than $\lceil \frac{n}{2} \rceil$ (see [11, Proposition 2.3]), and so by remark 1°, the number k of steps satisfies the inequality

$$\frac{k-1}{n} \leq \frac{n+1}{2n} - \frac{1}{d+1}, \tag{24}$$

and the asymptotical upper bound for k is $\frac{n}{2}$.

Remark 3.2. In the case when we add artificial data points to the original data set, n in (23) should be replaced with N, where N is the cardinality of the augmented data set. The upper bound for k in (23) remains the same.

The rates of complexity with respect to n and d of Theorem 3.1 are confirmed by simulation results presented in Figures 3 and 4. Measurements are taken on simulated samples of size n from d-dimensional $\mathcal{N}(0, I)$ distribution, where I is the unit $d \times d$ matrix, with $d = 2, ..., 10$ and $n = 40$, 80, 160, 320, 640, 1280, 2560, 3000, 3500, 4000, 4500, 5000, 5500, 6000, 6500, 7000. The results are averaged on 10 repetitions for each fixed pair (d, n).

For assessing the accuracy of the median approximation, we also use simulated data sets from $\mathcal{N}(0, I)$. Let $\hat{m}$ be an approximate median obtained as the output of the algorithm. Knowing that the median of $\mathcal{N}(0, I)$ is at origin, and that

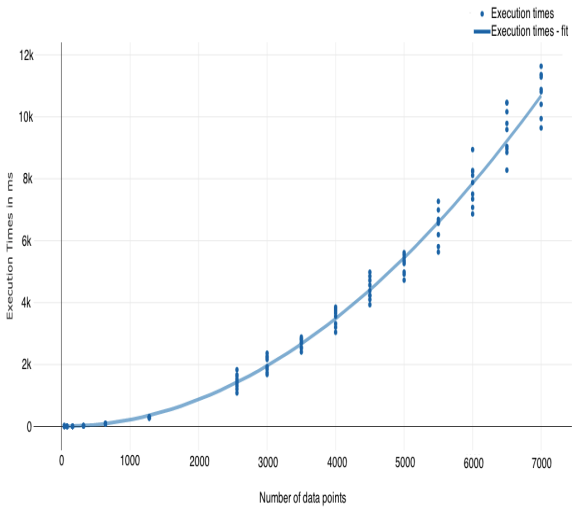

Figure 3: When number of points increases, the execution time grows with the order of $n^2 \log n$.

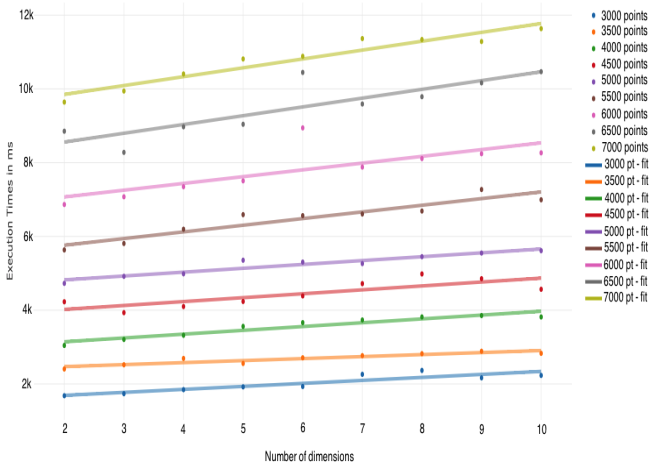

Figure 4: The execution time grows linearly with dimensionality.

the squared distance to origin has $\chi^2(d)$ distribution, it is convenient to take p-values $P(\chi^2(d) \leq \|\hat{m}\|^2)$ as a measure of error. For samples of size $n = 1000$ in

ten selected dimensions, the measurements are performed using ABCDEPTH and DEEPLOC on each sample. The results summarized in Table 1, and graphically presented in figure 5, show the superiority of our algorithm and advantage that increases with dimensionality.

Table 1: Error size of approximate median points in terms of p-values $P(\chi^2(d) \leq \|\hat{m}\|^2)$

n	Algorithm	d									
		4	5	6	7	8	9	10	20	100	200
1000	DEEPLOC	0.0021	0.0027	0.0039	0.0041	0.0045	0.0051	0.0056	0.0056	0.0612	0.1234
	ABCDEPTH	0.0011	0.0016	0.0012	0.0011	0.0015	0.0015	0.0012	0.0012	0.0022	0.002

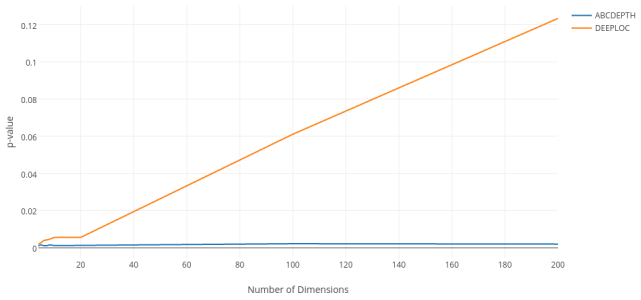

Figure 5: Plotted values from Table 1

3.5.2. Examples

The first simple example considers n points in dimension 1 generated from normal $\mathcal{N}(0,1)$ distribution. By running ABCDepth in this case with $n = 1000$, we got two points (as expected) in the median level set, $S_{0.5} = \{-0.00314, 0.00034\}$. With another sample with $n = 1001$ (odd number) from the same distribution, the median set was a singleton, $S_{0.5} = \{0.0043\}$

Now, we demonstrate data sets generated from bivariate and multivariate normal distribution. Figure 6 and Figure 7 show the median calculated from 1000 points in dimension 2 and 3, respectively from normal $\mathcal{N}(0, I)$ distribution. Starting from $\alpha = \frac{1}{d+1}$ (see Theorem 2.1) the algorithm produces ~ 200 levels sets for $d = 2$, and ~ 300 level sets for $d = 3$, so not all of them are plotted. On both figures the median is represented as a black point with depth $\frac{499}{1000}$ on Figure 6, i.e. $\frac{493}{1000}$ on Figure 7.

All data generators that we use in this paper in order to verify and plot the algorithm output were presented at [2], and they are available within an open source project at `https://bitbucket.org/antomripmuk/generators`.

As a real data example, we take a data set which is rather sparse. The data set is taken from [29], and it has been used in several other papers as a bench-

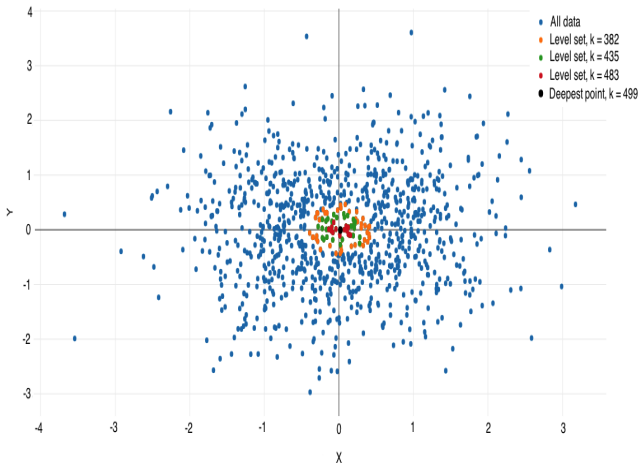

Figure 6: Bivariate normal distribution - four level sets, where the black point at the center is the deepest point

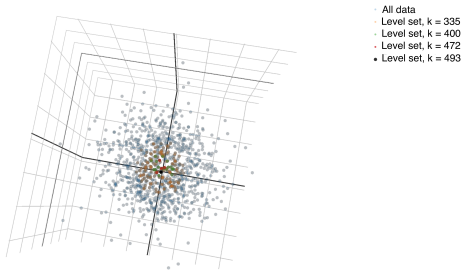

Figure 7: 3D normal distribution - four level sets, where the black point at the center is the deepest point

mark. It contains 23 four-dimensional observations in period from 1966 to 1967 that represent seasonally adjusted changes in auto thefts in New York city. For the sake of clarity, we take only two dimensions: percent changes in manpower, and seasonally adjusted changes in auto thefts. The data is downloaded from `http://lib.stat.cmu.edu/DASL/Datafiles/nycrimedat.html`. Figure 8 shows the output of ABCDepth algorithm if we consider only points from the sample

(orange point). Obviously, the approximate median belongs to the original data set. Then, we run ABCDepth algorithm with 1000 artificial data points from the uniform distribution as explained in Section 2 and earlier in this section. The approximate median obtained by this run (green point) has the same depth of $\frac{9}{23}$ as the median calculated using DEEPLOC algorithm [29] by running their Fortran code (red point). We check depths of those two points (green and red) applying *depth* function based on [26] and implemented in R "depth" package [14]. Evidently, the median, in this case, is not a singleton, i.e. there is more than one point with depth $\frac{9}{23}$. By adding more than 1000 artificial points, we can get more than one median point. We will discuss this example again in subsection 3.6.2.

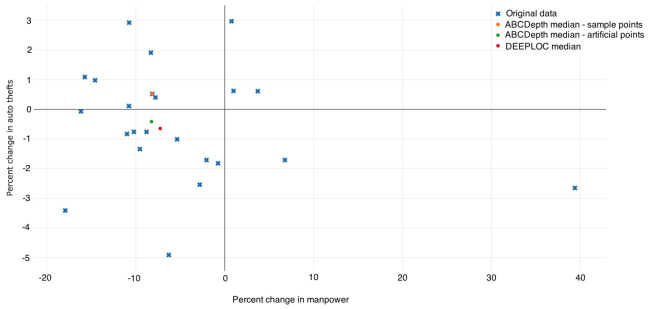

Figure 8: NY crime data set, comparison of Tukey medians using ABCDepth and DEEPLOC

Another two examples are chosen from [27]. Figure 9 shows 27 two-dimensional observations that represent animals brain weight (in g) and the body weight (in kg) taken from [22]. In order to represent the same data values, we plotted the logarithms of those measurements as done in [27].

Figure 10 considers the weight and the cost of 23 single-engine aircraft built between $1947 - 1979$. This data set is taken from [15].

As in Figure 8, in those two figures the orange point is the median obtained by running ABCDepth algorithm using only sample data. Green and red points represent outputs of ABCDepth algorithm applied by adding 1000 artificial data points from the uniform distribution and DEEPLOC median, respectively. These two examples show the importance of out-of-sample points in finding the depth levels and Tukey's median.

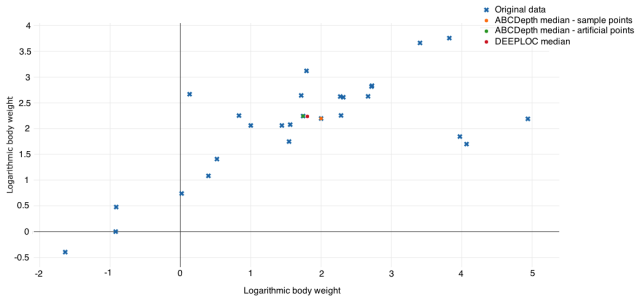

Figure 9: Animals data set, comparison of Tukey medians using ABCDepth and DEEPLOC

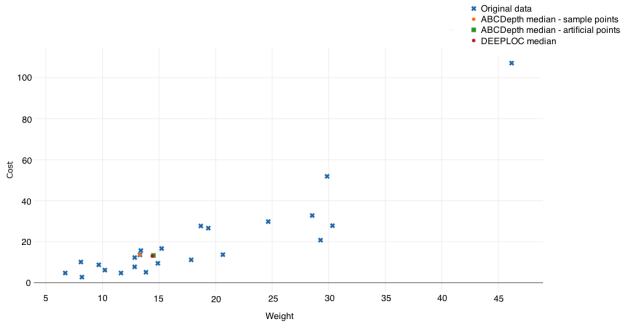

Figure 10: Aircraft data set, comparison of Tukey medians using ABCDepth and DEEPLOC

3.6. Adapted Implementation: Algorithm 2 for finding the Tukey's depth of a sample point and out-of-sample point

Let us recall that by Corollary 2, a point $\boldsymbol{x}$ has depth h if and only if $\boldsymbol{x} \in S_\alpha$ for $\alpha \leq h$ and $\boldsymbol{x} \notin S_\alpha$ for $\alpha > h$. With a sample of size n, we can consider only $\alpha = \frac{k}{n}$, $k = 1, \ldots, n$, because for $\frac{k-1}{n} < \alpha < \frac{k}{n}$, we have that $D(\boldsymbol{x}) \geq \alpha \iff D(\boldsymbol{x}) \geq \frac{k}{n}$. Therefore, the statement of Corollary 2 adapted to the sample distribution can be formulated as (using the fact that $S_\beta \subset S_\alpha$ for $\alpha < \beta$):

$$D(\boldsymbol{x}) = \frac{k}{n} \iff \boldsymbol{x} \in S_{\frac{k}{n}} \quad \text{and} \quad \boldsymbol{x} \notin S_{\frac{k+1}{n}}. \tag{25}$$

From (25) we derive the algorithm for Tukey's depth of a sample point $\boldsymbol{x}$ as follows. Let $\alpha_k = \frac{k}{n}$. The level set S_{α_1} contains all points in the sample. Then we construct S_{α_2} as an intersection of n balls that contain $n-1$ sample points. If $\boldsymbol{x} \notin S_{\alpha_2}$, we conclude that $D(\boldsymbol{x}) = 1/n$, and stop. Otherwise, we iterate this procedure till we get the situation as in right side of (25), when we conclude that the depth is $\frac{k}{n}$. The output of the algorithm is k.

Remark 3.3. As in Remark 3.1, it can be shown that the approximate depth k/n is never greater than the true depth.

Implementation-wise, in order to improve the algorithm complexity, we do not need to construct the level sets. It is enough to count balls that contain point $\boldsymbol{x}$. The algorithm stops when for some k, there exists at least one ball (among the candidates for the intersection) that does not contain $\boldsymbol{x}$. Thus, the depth of the point $\boldsymbol{x}$ is $k-1$.

With a very small modification, the same algorithm can be applied to a point $\boldsymbol{x}$ out of the sample. We can just treat $\boldsymbol{x}$ as an artificial point, in the same way as in previous sections. That is, the size of the required balls has to be $n-k+1$ points from the sample, not counting $\boldsymbol{x}$. The rest of the algorithm is the same as in the case of a sample point $\boldsymbol{x}$.

In both versions (sample or out-of-sample), we can use additional artificial points to increase the precision. The sample version of the algorithm is detailed below.

Data: Original data, $X_n = (\boldsymbol{x_1}, \boldsymbol{x_1}, ..., \boldsymbol{x_n}) \in \mathbb{R}^{d\times n}$, $\boldsymbol{x} = \boldsymbol{x}_i$ for a fixed i - the data point whose depth is calculated.

Result: Tukey depth at $\boldsymbol{x}$.

```
   /* Iteration Phase                                          */
 1 S_{α_1} = {x_1, ..., x_n};
 2 for k ← 2 to n do
 3 |   p = 0 - Number of balls that contain x. Its initial value is 0 ;
   |   /* Find balls that contain point x                      */
 4 |   for i ← 1 to n do
 5 |   |   if x ∈ B_i, where B_i contains n − k + 1 original data points then
 6 |   |   |   p = p + 1;
 7 |   |   end
 8 |   end
 9 |   if p ≠ n then
10 |   |   return k − 1
11 |   end
12 end
```

Algorithm 2: Calculating Tukey depth of a sample point.

3.6.1. Complexity

Theorem 3.2. *Adapted ABCDepth algorithm for finding approximate Tukey depth of a sample point has order of $O(dn^2 + n^2 \log n)$ time complexity.*

Proof. Balls construction for Algorithm 2 is the same as in Algorithm 1 (lines 1-10), so by Theorem 3.1 this part runs in $O(dn^2+n^2\log n)$ time. For the point with the depth α_k, algorithm enters in iteration loop k times and it iterates through all n points to find the balls that contain point $\boldsymbol{x}$, so the whole iteration phase runs in $O(kn)$ time.

Overall complexity of the Algorithm 2 is:

$$O(dn^2 + n^2 \log n) + O(kn) \sim O(dn^2 + n^2 \log n), \tag{26}$$

which had to be proved. □

Remark 3.4. When the input data set is sparse or when the sample set is small, we add artificial data to the original data set in order to improve the algorithm accuracy. In that case, n in (26) should be replaced with N.

3.6.2. Examples

To illustrate the output for the Algorithm 2, we use the same real data sets as we used in Figures 8-10. For all data sets we applied Algorithm 2 in two runs; first time with sample points only, and second time with additional 1000 artificial points generated from uniform distribution. Points' depths are verified using *depth* function from [26] implemented in [14]. For each data set, we calculate the accuracy as $\frac{100k}{n}\%$, where n is the sample size and k is the number of points that has the correct depth compared with algorithm presented in [26].

In Figure 11 we showed NY crime points depths with accuracy of 26%, but if we add more points to the original data set as we showed on Figure 12, the accuracy is greatly improved, to 87%.

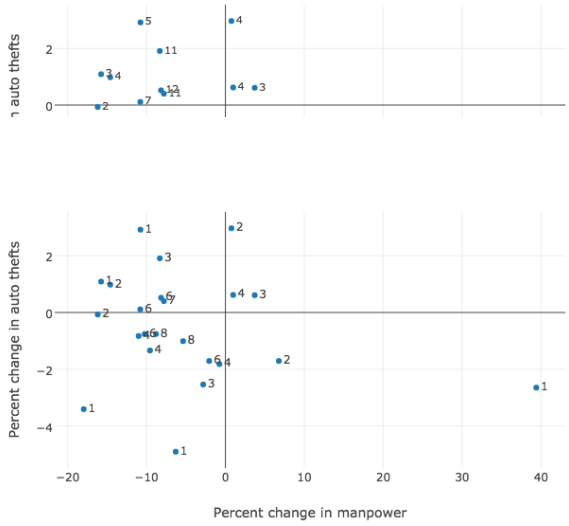

Figure 12: NY crime data - point depths using original and artificial data

Figure 13 shows the same accuracy of 26% for animals data set, in the case when Algorithm 2 is run with sample points only. By adding more points as in Figure 14, the accuracy is improved to 100%.

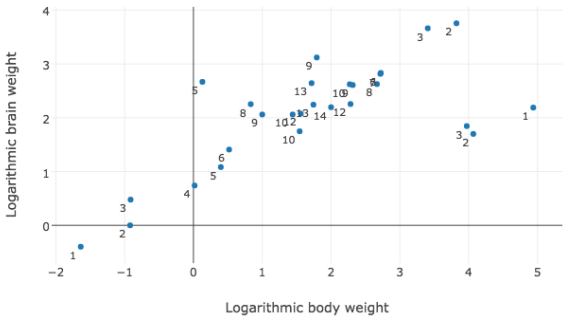

Figure 13: Animals data - point depths using only original data.

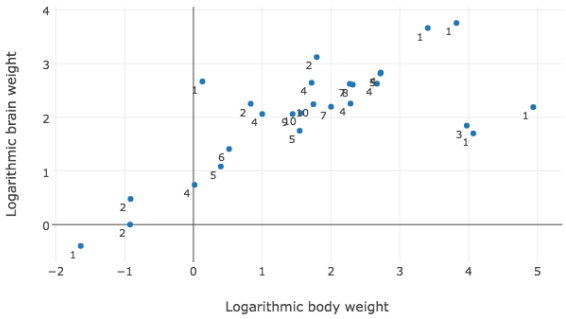

Figure 14: Animals data - point depths using original and artificial data

The third example is aircraft data set presented in Figure 15 and Figure 16. The accuracy with artificial points is 95%, otherwise it is 18%.

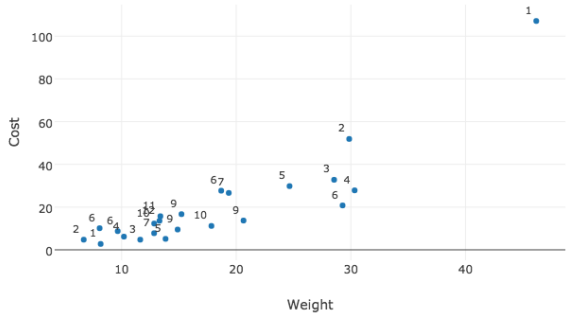

Figure 15: Aircraft data - point depths using only original data.

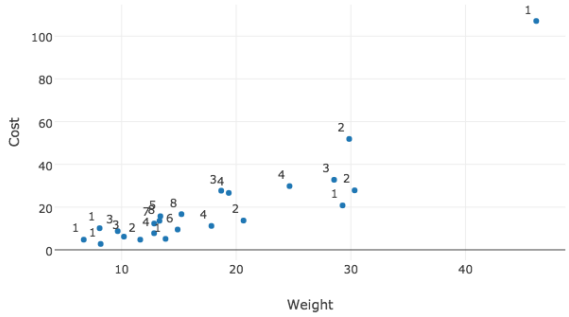

Figure 16: Aircraft data - point depths using original and artificial data

As the last example of this section, we would like to calculate depths of the points plotted on Figure 8 using ABCDepth Algorithm 2. In Figure 8 we plotted Tukey median for NY crime data set using Algorithm 1 with artificial data points (green point) and compared the result with the median obtained by DEEPLOC (red point). Both points are out of the sample. In Figure 17, we show depths of all sample points including the depths of two median points, all attained by ABCDepth Algorithm 2. Algorithm presented in [26] and ABCDepth Algorithm 2 calculate the same depth value for both median points.

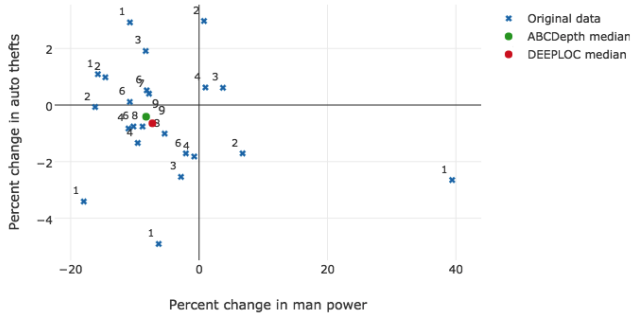

Figure 17: NY crime data - point depths using original and artificial data

4. PERFORMANCE AND COMPARISONS

According to Theorem 3.1, the complexity of calculating Tukey median grows linearly with the dimension and in terms of a number of data points, it grows with the order of $n^2 \log n$. Rousseeuw and Ruts in [24] pioneered with an exact algorithm called HALFMED for Tukey median in two dimensions that runs in $O(n^2 log^2 n)$ time. This algorithm is better than ABCDepth for $d = 2$, but it processes only bivariate data sets. Struyf and Rousseeuw in [29] implemented the first approximate algorithm called DEEPLOC for finding the deepest location in higher dimensions. Its complexity is $O(kmn \log n + kdn + md^3 + mdn)$ time, where k is the number of steps taken by the program and m is the number of directions, i.e. vectors constructed by the program. This algorithm is very efficient for low-dimensional data sets, but for high-dimensional data sets ABCDepth algorithm outperforms DEEPLOC. Chan in [6] presents an approximate randomized algorithm for maximum Tukey depth. It runs in $O(n^{d-1})$ time but it has not been implemented yet.

In Table 2 execution times of DEEPLOC algorithm and ABCDepth algorithm for finding Tukey median are reported. The measurements are performed using synthetic data generated from the multivariate normal $\mathcal{N}(0, 1)$ distribution. In this table, we demonstrate how ABCDepth algorithm behaves with thousands of high-dimensional data points. It takes ~ 13 minutes for $n = 7000$ and $d = 2000$. Since DEEPLOC algorithm does not support data sets with $d > n$ and returns the error message: "the dimension should be at most the number of objects", we denoted those examples with $-$ sign in the table. The sign $*$ means that the median is not computable at least once in 12 hours.

Table 2: Comparison between DEEPLOC and ABCDepth execution times in seconds

d	Algorithm	n												
		320	640	1280	2560	3000	3500	4000	4500	5000	5500	6000	6500	7000
50	Deeploc	4.43	7.15	12.65	23.87	30.93	31.79	37.66	45.35	50.72	63.13	63.75	84.13	69.61
	ABCDepth	0.15	0.63	2.86	4.95	7.27	8.65	12.51	14.18	17.51	22.18	25.86	29.24	37.34
100	Deeploc	19.42	22.85	33.81	77.45	69.04	105.56	97.39	120.05	140.04	131.85	127.36	212.42	183.27
	ABCDepth	0.22	0.92	2.03	7.83	9.78	13.14	17.89	23.52	30.6	39.18	49.03	68.46	82.02
500	Deeploc	-	1616.53	*	*	*	*	*	*	*	*	*	*	*
	ABCDepth	0.693	3.181	8.4	27.9	41.61	53.73	71.95	89.36	109.22	140.18	151.45	180.5	213.01
1000	Deeploc	-	-	*	*	*	*	*	*	*	*	*	*	*
	ABCDepth	1.165	3.99	14.389	54.18	74.38	98.73	129.85	164.96	203.37	246.54	286.17	344.94	39.16
2000	Deeploc	-	-	-	*	*	*	*	*	*	*	*	*	*
	ABCDepth	2.21	7.86	27.25	107.46	132.77	180.02	243.1	297.6	386.75	475.87	554.23	666.4	764.74

ABCDepth algorithm for finding Tukey depth of a point runs in $O(dn^2 + n^2 \log n)$ as we showed in Theorem 3.2. Most of the algorithms for finding Tukey depth are exact and at the same time computationally expensive. One of the first exact algorithms for bivariate data sets, called LDEPTH, is proposed by Rousseeuw and Ruts in [23]. It has complexity of $O(n \log n)$ and like HALFMED, it outperforms ABCDepth for $d = 2$. Rousseeuw and Struyf in [26] implemented an exact algorithm for $d = 3$ that runs in $O(n^2 \log n)$ time, and an approximate algorithm for $d > 3$ that runs in $O(md^3 + mdn)$, where m is the number of directions perpendicular to hyperplanes through d data points.

The later work of Chen et al. in [7], presented approximate algorithms, based on the third approximation method of Rousseeuw and Struyf, in [26], reducing the problem from d to k dimensions.

The first one, for $k = 1$, runs in $O(\epsilon^{1-d}dn)$ time and the second one, for $k \geq 2$, runs in $O((\epsilon^{-1}c \log n)^d)$, where ϵ and c are empirically chosen constants. Another exact algorithm for finding Tukey depth in $\mathbb{R}^d$ is proposed by Liu and Zuo in [17], which proves to be extremely time-consuming (see Table 5.1 of Section 5.3 in [20]) and the algorithm involves heavy computations, but can serve as a benchmark. Recently, Dyckerhoff and Mozharovskyi in [13] proposed two exact algorithms for finding halfspace depth that run in $O(n^d)$ and $O(n^{d-1} \log n)$ time.

Table 3 shows execution times of ABCDepth algorithm for finding a depth of a sample point. Measurements are derived from synthetics data from the multivariate standard normal distribution. Execution time for each data set represents averaged time consumed per data point. Most of the execution time ($\sim 95\%$) is spent on balls construction (see lines 1-10 of the Algorithm 1), while finding a point depth itself (iteration phase of the Algorithm 2) is really fast since it runs in $O(kn)$ time.

The ABCDepth algorithms have been implemented in Java. Tests for all algorithms are run using one kernel of Intel Core i7 (2.2 GHz) processor. Computational codes are available from the authors upon request.

Table 3: Average time per data point.

d	n												
	320	640	1280	2560	3000	3500	4000	4500	5000	5500	6000	6500	7000
50	0.07	0.21	1.21	8.23	12.64	19.22	28.56	42.04	64.33	77.45	98.79	121.86	150.73
100	0.08	0.25	1.23	8.18	13.91	20.48	28.51	44.31	65.55	81.91	99.96	123.84	154.65
500	0.13	0.42	1.84	11.42	17.93	21.42	35.41	52.07	73.21	95.18	119.82	141.88	176.12
1000	0.17	0.53	2.52	13.53	20.13	32.35	41.71	58.72	82.92	103.84	138.69	155.32	200.55
2000	0.26	0.94	4.12	18.32	28.12	38.79	56.04	73.79	102.98	124.48	156.54	186.59	232.45

5. CONCLUDING REMARKS

There is no doubt that exact algorithms are needed, whether it is about calculating the depth of a point, or a multivariate median. Those algorithms are precise and serve as an benchmark measurement for all approximate algorithms. Nowadays, the real life applications such as clustering, classification, outlier detection or, in general, any kind of data processing, contain at least thousands of multidimensional observations. In those cases, available exact algorithms are not the best choice - the complexity of the exact algorithms grows exponentially with dimension due to projections of sample points to a large number of directions. Hence, the exact algorithms are time consuming and often restricted by number of observations and its dimensionality. Therefore, for large data sets, approximate algorithms correspond to a good solution.

In this paper we presented approximate ABCDepth algorithms based on a novel *balls intersection* idea explained in Sections 2 and 3 that brings a lots of advantages. With a small modification, the main idea from [19] is used for implementing two algorithms: one is for calculating Tukey median and the other one is for calculating Tukey depth of a sample and out-of-sample point. Using synthetic and real data sets and comparing our performances with those of previous approximate algorithms, we showed that our algorithms fulfill the following:

i) high accuracy (see examples in Sections 3.5.2 and 3.6.2), ii) they are much faster especially for large n and d (see Tables 2 and 3 as well as Table 5.1 of Section 5.3 in [20]), iii) they can handle a larger number of multidimensional observations; those algorithms are the only algorithms tested with data sets that contain up to $n = 7000$ and $d = 2000$,, iv) Algorithm 1 computes multidimensional median with the complexity of $O((d + k)n^2 + n^2 \log n)$, v) Algorithm 2 computes the depth of a single point with complexity of $O(dn^2 + n^2 \log n)$, vi) both complexities have linear growth in d and quadratic growth in n (see Figures 3 and 4), vii) an additional theoretical advantage of ABCDepth approach is that the data points are not assumed to be in "general position".

Disclaimer. A previous version of this work was presented in a poster session of CMStatistics2016 and the abstract is posted and available in [4]. Otherwise, this paper has not been published in conference proceedings or elsewhere.

Acknowledgements: We would like to express our gratitude to Anja Struyf and coauthors for sharing the code and the data that were used in their papers of immense importance in the area. Answering to Yijun Zuo's doubts about the first arXiv version [5] of this paper and solving difficult queries that he was proposing, helped us to improve the presentation and the algorithms. The second author acknowledges the support by grants III 44006 and 174024 from Ministry of Education, Science and Technological Development of Republic of Serbia.

REFERENCES

[1] Ahn, H.-K., Knauer, C., Scherfenberg, M., Schlipf, L., and Vigneron, A., "Computing the discrete Fréchet distance with Imprecise input", in: *Algorithms and Computation, ISAAC 2010, Lecture Notes in Computer Science*, Ieju Island, Korea, (6507)(2016) 422-433.

[2] Bogićević, M., and Merkle, M., "Multivariate Medians and Halfspace Depth: Algorithms and Implementation", in: *Proc. 1st International Conference on Electrical, Electronic and Computing Engineering (IcETRAN 2014), Vrnjačka Banja, Serbia*, (1)(2014) 27-33. `http://milanmerkle.etf.rs/wp-content/uploads/2016/11/Bogicevic-Merkle-2014.pdf`.

[3] Bogićević, M., and Merkle, M., "Data Centrality Computation: Implementation and Complexity Calculation", in: *Proc. 2nd International Conference on Electrical, Electronic and Computing Engineering (IcETRAN 2015), Srebrno Jezero, Serbia*, (1)(2015) 23-29. `http://milanmerkle.etf.rs/wp-content/uploads/2016/11/Bogicevic-Merkle-2015.pdf`.

[4] Bogićević, M., and Merkle, M., "ABCDepth: Efficient algorithm for Tukey depth", in: *9th International Conference of the ERCIM WG on Computational and Methodological Statistics (CMStatistics 2016)*, 2016. `http://cmstatistics.org/RegistrationsV2/CMStatistics2016/viewSubmission.php?in=1774&token=p2q8o69709n312568rs4p4n4spn97n57`.

[5] Bogićević, M., and Merkle, M., "ABCDepth: efficient algorithm for Tukey depth", *arXiv:1603.05609*, 2016.

[6] Chan, T. M., "An Optimal Randomized Algorithm for Maximum Tukey Depth", in: *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ACM, New York, (2004) 430–436.

[7] Chen, D., Morin, P., and Wagner, U., "Absolute approximation of Tukey depth: Theory and experiments", *Comput. Geom.*, (46) (2013) 566–573.

[8] Ding, B., and König, A. C., "A Fast set intersection in memory", *Proceedings of the VLDB Endowment*, (4)(2011) 255–266.

[9] Donoho, D. L., "Breakdown properties of multivariate location estimators, Harvard University, Cambridge, Massachusetts, US, 1982.

[10] Donoho, D. L., and Gasko, M., "Multivariate Generalizations of the Median and Trimmed Mean, I", Technical report 133, Department of Statistics, University of California, Berkeley, December, 1987.

[11] Donoho, D. L., and Gasko, M., "Breakdown properties of location estimates based on halfspace depth and projected outlyingness", *Ann. Statist.*, (20) (1992) 1803–1827.

[12] Dutta, S., Ghosh, A. K., and Chaudhuri, P., "Some intriguing properties of Tukey's halfspace depth", *Bernoulli*, (17) (2011) 1420–1434.

[13] Dyckerhoff, R., and Mozharovskyi, P., "Exact computation of the halfspace depth", *Computational Statistics and Data Analysis*, (98) (2016) 19–30.

[14] Genest, M., Jean-Claude, and Plante, J.-F., Package depth, 2012. `https://cran.r-project.org/web/packages/depth/index.html`.

[15] Gray, J., "Graphics for regression diagnostics", *ASA Proc. Statistical Computing Section*, (1985) 102–107.

[16] Hoare, C. A. R., Algorithm 64: Quicksort, *Comm. Acm.*, (4) (1961) 321.

[17] Liu, X., and Zuo, Y., "Computing halfspace depth and regression depth", *Communications in Statistics Simulation and Computation,* (43) (2014) 969–985.
[18] Merkle, M., "Jensen's inequality for medians", *Stat. Prob. Letters*, (71) (2005) 277–281.
[19] Merkle, M., "Jensen's inequality for multivariate medians", *J. Math. Anal. Appl.,*, (370) (2010) 258–269.
[20] Mozharovskyi, P., *Contributions to depth-based classification and computation of the Tukey depth*, PhD thesis, Faculty of Economics and Social Sciences, University of Cologne, France, 2014.
[21] Rousseeuw, P. J., and Hubert, M., Statistical depth meets computational geometry: a short survey, *arXiv*: 1508.03828, 2015.
[22] Rousseeuw, P. J., and Leroy, A. M., Robust Regression and Outlier Detection, *Wiley*, (1997) 57.
[23] Rousseeuw, P. J., and Ruts, I., "Bivariate Location Depth", *Journal of the Royal Statistical Society, Series C (Applied Statistics)*, (45) (1996) 516–526.
[24] Rousseeuw, P. J., and Ruts, I., "Constructing the Bivariate Tukey Median", *Statistica Sinica*, (8) (1998) 827–839.
[25] Rousseeuw, P. J., and Ruts, I., "The depth function of a population distribution", *Metrika,* (49) (1999) 213–244.
[26] Rousseeuw, P. J., and Struyf, A., "Computing location depth and regression depth in higher dimension", *Statistics and Computing,* (8) (1998) 193–203.
[27] Ruts, I., and Rousseeuw, P. J., "Computing depth contours of bivariate point clouds", *Computational Statistics and Data Analysis,* (23) (1996) 153–168.
[28] Small, C. G., "A survey of multidimensional medians", *Internat. Statist. Inst. Rev.,* (58) (1990) 263–277.
[29] Struyf, A., and Rousseeuw, P. J., "High-dimensional computation of the deepest location", *Comp. Statist. & Data Anal.,* (34)(2000) 415–426.
[30] Tukey, J., *Order Statistics*, In Mimeographed notes for Statistics 411, Princeton University., 1974.
[31] Tukey, J., Mathematics and Picturing Data, in: *Proc. International Congress of Mathematicians, Vancouver, 1974*, (2) (1975) 523–531.
[32] Zhou, Y., and Serfling, R., "Multivariate spatial U-quantiles: A Bahadur-Kiefer representation, a Theil-Sen estimator for multiple regression, and a robust dispersion estimator", *J. Statist. Plann. Inference,* (138)(2008) 1660–1678.
[33] Zuo, Y., and Serfling, R., "General notions of statistical depth function", *Ann. Stat.,* (28) (2000) 461–482.