An Optimal Randomized Algorithm for Maximum Tukey Depth

Timothy M. Chan*

Abstract

We present the first optimal algorithm to compute the maximum *Tukey depth* (also known as *location* or *halfspace depth*) for a non-degenerate point set in the plane. The algorithm is randomized and requires $O(n \log n)$ expected time for n data points. In a higher fixed dimension $d \geq 3$, the expected time bound is $O(n^{d-1})$, which is probably optimal as well. The result is obtained using an interesting variant of the author's randomized optimization technique, capable of solving "implicit" linear-programming-type problems; some other applications of this technique are briefly mentioned.

1 Introduction

1.1 Problem statement. Given a set P of n points in $\mathbb{R}^d$, the *Tukey depth* of a point $q \in \mathbb{R}^d$ is defined as:

$$\min\{|P \cap \gamma| \ : \ \text{over all halfspaces } \gamma \text{ containing } q\}.$$

We are interested in finding a *Tukey median*, that is, a point $q \in \mathbb{R}^d$ with the largest Tukey depth.

1.2 Motivation. Notions of depths for point data sets are important in statistical analysis. The above definition (also called *location depth*, *data depth*, and *halfspace depth*) is among the most well-known and was popularized by John Tukey [47], who suggested using the corresponding depth *contours* (boundaries of regions of all points with equal depth) to visualize data. A Tukey median can serve as a point estimator for the data set (a "center") which is robust against outliers, does not rely on distances, and is invariant under affine transformations [41, 42, 45].

Because of the applications to statistics, the issue of designing efficient algorithms to find Tukey medians and their relatives—for example, a point with maximum *Liu/simplicial depth*, minimum *Oja depth*, or maximum *convex-layers/peeling depth*, and a line or flat with maximum *regression depth*—has attracted a great deal of attention from researchers in computational geometry [2, 3, 5, 25, 28, 29, 30, 31, 37]. The topic has even led to a DIMACS workshop recently. The goal of the present paper is to determine the computational complexity for the most basic such problem.

1.3 Previous results. For dimension $d = 1$, the Tukey median problem obviously corresponds to the standard median and can therefore be solved in $O(n)$ time [19]; the maximum depth here is exactly $\lceil n/2 \rceil$.

For higher d, the maximum Tukey depth is between $\lceil n/(d+1) \rceil$ and $\lceil n/2 \rceil$; the lower bound here can be proved by Helly's theorem. In the geometry literature, a point with depth at least $\lceil n/(d+1) \rceil$ is referred to as a *centerpoint*. The first nontrivial algorithmic result for $d = 2$, by Cole *et al.* [18], was stated in terms of centerpoints: they showed that a centerpoint can be found in $O(n \log^5 n)$ time, using a two-level application of *parametric search* [35]. Cole's refined parametric-search technique [17] subsequently reduced the time bound to $O(n \log^3 n)$. Later, an $O(n)$-time algorithm for centerpoints in the plane was discovered by Jadhav and Mukhopadhyay [27], using a clever *prune-and-search* approach. This does not solve the Tukey median problem, however.

In 1991, Matoušek [32] described an algorithm that can decide whether the maximum Tukey depth is at least a given value k in $O(n \log^4 n)$ time, using also a two-level parametric search as a subroutine. His algorithm actually constructs the entire depth-k contour (the region of all points at depth at least k). Consequently, a binary search in k yields the maximum Tukey depth and a Tukey median in $O(n \log^5 n)$ time.

Matoušek's result has remained unsurpassed, until recently. In 2000, Langerman and Steiger [29] obtained a faster decision algorithm with an $O(n \log^3 n)$ running time by using an alternative to parametric search; this algorithm does not construct the depth contour. The additional binary search then leads to an $O(n \log^4 n)$ time bound for Tukey median. Subsequently, Langerman and Steiger [31] showed that the Tukey median problem itself can be solved in $O(n \log^3 n)$ time. Some extra logarithmic factors seem inherent in their approach, because of its binary-search-like behavior. There is an $\Omega(n \log n)$ lower bound on the time complexity for computing the maximum Tukey depth (or deciding whether the maximum depth is at least k, or testing the depth value of just a single point q, or find-

*School of Computer Science, Univ. of Waterloo, Waterloo, Ontario N2L 3G1, Canada, tmchan@uwaterloo.ca. This work has been supported in part by an NSERC Research Grant.

ing a Tukey median that is extreme along a given direction) [3, 29]. We conjecture that the $\Omega(n \log n)$ lower bound holds for finding an arbitrary Tukey median as well.

Extensions of the algorithms to $d = 3$ were also discussed in some of the previous papers, with more logarithmic factors in the running time. For example, with a three-level parametric search, Cole *et al.*'s centerpoint algorithm [17] now takes time $O(n^2 \log^7 n)$, or by Cole's refinement, $O(n^2 \log^4 n)$ [17]. Another $O(n^2 \operatorname{polylog} n)$ algorithm was apparently given by Naor and Sharir [38].

Note that the problem is difficult because of our insistence on using exact depth values. Approximate versions of the problem can be solved considerably more quickly; for example, see [14, 32].

1.4 New results. For $d = 2$, we show that theoretically a faster decision algorithm is possible by randomization: we can decide whether the maximum depth is at least a given value k in $O(n \log n)$ expected time (assuming non-degeneracy of the input). Our algorithm does not construct the entire depth-k contour, but it can find a point of depth at least k that is extreme along any given direction. The algorithm is based on a generalization of the author's randomized optimization technique [7]. This generalized technique is interesting in its own right, as it can deal with certain LP-type problems where the constraints are too numerous to write down and are instead specified "implicitly".

By binary search in k, the maximum Tukey depth can now be computed in $O(n \log^2 n)$ expected time. We show that the binary search can also be avoided, by the generalized randomized optimization technique again. We thus have a randomized $O(n \log n)$-time algorithm for computing the maximum Tukey depth and a Tukey median. In view of the aforementioned lower bound (which holds for randomized algorithms), the result is optimal, at least if the maximum Tukey depth value is desired.

Our algorithm in fact works for any fixed dimension $d \geq 3$ and requires $O(n^{d-1})$ expected time. As the problem of detecting affine degeneracies (the existence of d points on a common hyperplane) among n points in $\mathbb{R}^{d-1}$ is believed to require $\Omega(n^{d-1})$ time [24] and can be reduced to computing the maximum Tukey depth in $\mathbb{R}^d$, our result is likely to be optimal for $d \geq 3$ as well. Note that as a byproduct, we get an improved $O(n^2)$-time randomized algorithm for centerpoints in $\mathbb{R}^3$.

2 A Randomized Optimization Technique for Implicit LPs

In this section, we present general tools that are of interest not just to the Tukey depth problem.

2.1 The original technique. Several years ago, the author [7] identified a simple lemma thatcan surprisingly be used to solve many geometric optimization problems. As in the popular *parametric-search* technique [1, 35], the strategy is to solve the decision problem first (requirement 1): decide whether the optimal value is at least a given value. Once a decision algorithm is found, an algorithm for the optimization problem can usually be obtained by following a general "recipe", even if the values are real numbers (where an ordinary binary search is not applicable).

In the parametric-search recipe, an efficient parallel version of the decision algorithm (or another algorithm with appropriate characteristics) is usually required; the transformed algorithm is not only slower by some polylogarithmic factor but also quite complicated (sometimes un-implementable!). The novelty of the lemma below is requirement 2: if this particular condition is met for the problem at hand, not only can these complications be bypassed by a simpler randomized algorithm (which uses the decision algorithm only as a black box), but the resulting algorithm is also faster and has *no* logarithmic-factor slow-down. Requirement 2 is similar to the design of *prune-and-search* algorithms (forming subproblems of size a fraction less), but unlike traditional prune-and-search, we do not need to know *a priori* which subproblem defines the optimal solution.

LEMMA 2.1. **[7]** *Let $\alpha < 1$ and r be fixed constants. Suppose $f : \mathcal{P} \to \mathbb{R}$ is a function that maps* inputs *to real values, with the following properties:*

0. *For any input $P \in \mathcal{P}$ of constant size, $f(P)$ can be computed in constant time;*

1. *For any input $P \in \mathcal{P}$ of size n and any $t \in \mathbb{R}$, we can decide whether $f(P) \geq t$ in time $D(n)$;*

2. *For any input $P \in \mathcal{P}$ of size n, we can construct inputs $P_1, \ldots, P_r \in \mathcal{P}$ each of size at most $\lceil \alpha n \rceil$, in time no more than $D(n)$, such that*

$$f(P) = \min\{f(P_1), \ldots, f(P_r)\}.$$

Then for any input $P \in \mathcal{P}$ of size n, we can compute $f(P)$ in $O(D(n))$ expected time, assuming that $D(n)/n^\varepsilon$ is monotone increasing.

The proof [7] follows from the well-known fact [19] that the standard way to compute the minimum of r numbers, if randomized, requires $O(r)$ comparisons but only $O(\log r)$ expected number of "evaluations". The algorithm uses this fact recursively.

2.2 LP-type problems. Randomized techniques had been discovered earlier for a special class of geometric optimization problems that share properties (as defined below) enjoyed by linear and convex programming. Problems in this class can be solved in linear time for any fixed dimension, by simple algorithms [43, 44].

DEFINITION 2.2. Let $w : 2^{\mathcal{H}} \to \mathcal{W}$ be a function that maps sets of *constraints* (members of $\mathcal{H}$) to values in a totally ordered set $\mathcal{W}$. We say that w is *LP-type* of dimension at most d if the following properties hold for all sets $H \subseteq \mathcal{H}$ and all constraints $h \in \mathcal{H}$:

(i) $w(H) = w(B)$ for some $B \subseteq H$ of size at most d.

(ii) $w(H \cup \{h\}) \geq w(H)$.

(iii) Suppose $w(H) = w(B)$ with $B \subseteq H$. Then $w(H \cup \{h\}) = w(H) \iff w(B \cup \{h\}) = w(B)$.

We call a set B of size at most d a *basis*, and if (i) is obeyed, a *basis for* H. If $w(B \cup \{h\}) = w(B)$, we say that B *satisfies* h. More generally, if $w(B \cup H) = w(B)$, B *satisfies* H. Primitive operations that algorithms may use include *basis evaluation* (computing $w(B)$ for a basis B) and *satisfaction/violation test* (determining if a basis B satisfies or violates a constraint h).

It can be shown from the definition that if w is LP-type, then the modified function $\overline{w} : 2^{2^{\mathcal{H}}} \to \mathcal{W}$ with $\overline{w}(\{H_1, \ldots, H_n\}) := w(H_1 \cup \cdots \cup H_n)$ is LP-type as well, of the same dimension. If w is the standard linear programming problem (the minimum of a fixed linear function over the intersection of a given set of halfspaces H), then $\overline{w}$ corresponds to a convex programming problem.

2.3 A generalized technique. In requirement 2 of the lemma, the subproblems are combined via the min operator, essentially 1-dimensional linear programming. The main observation of this section is that the min operator can be replaced more generally by d-dimensional linear programming. Parametric search can also reduce the d-dimensional optimization problem to a satisfaction/violation problem (requirement 1 without requirement 2), but in a more complicated *multi-level* or *multi-dimensional* form [16, 33, 39]. This technique incurs further polylogarithmic slow-down and does not carry over to abstract LP-type problems, unlike the new lemma below:

LEMMA 2.3. *Let $w : 2^{\mathcal{H}} \to \mathcal{W}$ be an LP-type function of constant dimension d and let $\alpha < 1$ and r be fixed constants. Suppose $f : \mathcal{P} \to 2^{\mathcal{H}}$ is a function that maps inputs to sets of constraints, with the following properties:*

0. *For inputs $P_1, \ldots, P_d \in \mathcal{P}$ of constant size, a basis for $f(P_1) \cup \cdots \cup f(P_d)$ can be computed in constant time;*

1. *For any input $P \in \mathcal{P}$ and any basis B, we can decide whether B satisfies $f(P)$ in time $D(n)$;*

2. *For any input $P \in \mathcal{P}$, we can construct inputs $P_1, \ldots, P_r \in \mathcal{P}$ each of size at most $\lceil \alpha n \rceil$, in time no more than $D(n)$, such that*

$$f(P) = f(P_1) \cup \cdots \cup f(P_r).$$

Then we can compute a basis for $f(P)$ in $O(D(n))$ expected time, assuming that $D(n)/n^{\varepsilon}$ is monotone increasing.

Proof. We describe a recursive algorithm that, given d inputs $P_1, \ldots, P_d$ each of size at most n, compute a basis for $f(P_1) \cup \cdots \cup f(P_d)$. The base case can be taken care of by requirement 0. By requirement 2, we can form inputs $Q_1, \ldots, Q_{dr}$ each of size at most $\lceil \alpha n \rceil$, and reduce the problem to computing a basis for $f(Q_1) \cup \cdots \cup f(Q_{dr})$, or in terms of the modified LP-type function $\overline{w}$, a basis for the set of dr elements $\{f(Q_1), \ldots, f(Q_{dr})\}$. The standard randomized incremental algorithm for LP-type problems [44] finds the solution using a linear ($O(r)$) expected number of satisfaction/violation tests and a polylogarithmic ($O(\log^d r)$) expected number of basis evaluations for $\overline{w}$. (In fact, a weighted sampling algorithm by Clarkson [12] uses only $O(\log r)$ basis evaluations.) The satisfaction tests can be handled by requirement 1 and the basis evaluations can be performed by recursive calls. As a result, we have the following recurrence for the expected running time:

$$T(n) = c \log^d r \, T(\lceil \alpha n \rceil) + O(rD(n)),$$

for some constant c that depends only on d. This recurrence is in the standard "master" form [19] and solves to $T(n) = O(D(n))$, provided that

$$\log(c \log^d r) / \log(1/\alpha) < \varepsilon.$$

If this inequality is not true, it can be made true by replacing α with α^{ℓ} and r with r^{ℓ} for a sufficiently large constant ℓ, since we can repeat the division procedure in requirement 2 a total of ℓ times before applying the recursive algorithm. □

2.4 Some examples. To illustrate the versatility of the new lemma, we briefly sketch a few applications where some known results can be re-derived:

- The problem of answering *linear programming queries* for a preprocessed set of n halfspaces in $\mathbb{R}^d$

was considered by Matoušek [33], who used multi-level parametric search to reduce the problem to designing data structures for satisfaction queries (usually called membership queries).

We can obtain a simpler reduction by the generalized lemma (similar to a reduction obtained by the original lemma for ray shooting queries [7]): just build a binary tree for the halfspaces and store a membership structure at each node (preprocessing time and space increases by at most a logarithmic factor); then requirement 2 trivially holds (with $r = 2$ and $\alpha = 1/2$) and we get an expected query time $Q(n)$ which coincides with the time for membership queries, with no extra logarithmic factor, if $Q(n)/n^{\varepsilon}$ is monotone increasing.

Similar results was obtained earlier by a different randomized method of the author [6]. The method here uses randomization only in the query algorithm, not the preprocessing, although the previous method can be derandomized more effectively, as shown by Ramos [40].

- *Minimum diameter of moving points.* Given n points $\{p_i(t)\}_{i=1,\ldots,n}$, each moving linearly in $\mathbb{R}^d$, for what the time value t is the diameter $\max_{ij} \|p_i(t) - p_j(t)\|$ the smallest?

  Gupta *et al.* [26] applied parametric search to get an $O(n \log^3 n)$-time algorithm for the two-dimensional problem. Clarkson [13] later described a randomized $O(n \log n)$-time algorithm in dimension $d \leq 3$, but this result can also be obtained as a corollary of the generalized lemma: because $\|p_i(t) - p_j(t)\|^2$ is a convex quadratic function in t, the problem is equivalent to a convex program with $O(n^2)$ constraints, one for each pair of points; requirement 2 can be met by partitioning the point set P into three subsets P_1, P_2, P_3 of equal size and expressing the constraint set for P as the union of the constraint sets for $P_1 \cup P_2$, $P_2 \cup P_3$, and $P_1 \cup P_3$ (with $r = 3$ and $\alpha = 2/3$). The satisfaction test reduces to computing the diameter of the point set at a fixed time, which can be accomplished in $O(n \log n)$ time for $d \leq 3$ [15].

- *Inverse parametric minimum spanning trees.* Eppstein [23] considered the following graph problem: we have an undirected graph $G(t_1, \ldots, t_d)$ with n vertices and m edges, where the weight of each edge is a linear function in $t_1, \ldots, t_d$; we are also given a tree T; the goal is to find parameters $t_1, \ldots, t_d$ (if exist) such that the minimum spanning tree of $G(t_1, \ldots, t_d)$ coincides with T.

  This problem can also be viewed as an implicit LP with $O(mn)$ constraints, one for each pair of non-tree edge e and tree edge e', where $T \cup \{e\} \setminus \{e'\}$ is a tree (the constraint is that the weight of e must be at least the weight of e'). In the journal version of his paper [23], Eppstein has already applied our randomized optimization technique to solve the problem in linear expected time, although he used the original lemma instead of the generalized lemma and the solution to the decision problem was not entirely clear.

3 Tukey Depth as an Implicit LP

We now detail the application of the new lemma to the maximum Tukey depth problem.

3.1 Finding a point of a given depth k. Let P be the given non-degenerate set of n points in $\mathbb{R}^d$. We first consider the problem of finding a point with Tukey depth at least k, minimizing a linear function, if such a point exists. That this is related to linear programming is not surprising, because the problem asks for an extreme point inside a halfspace intersection:

$$\bigcap \{\gamma \ : \text{ over all halfspaces } \gamma \text{ with } |P \setminus \gamma| < k\}.$$

In the subsequent discussion, it is best to switch to *dual* space [22]. Here, the linear programming problem becomes the following: given a set S of points colored black or white in $\mathbb{R}^d$, compute

$$\begin{array}{lcll} w(S) & := & \min & \varphi(h) \\ & & \text{s.t.} & \text{all black points of } S \text{ are below } h \\ & & & \text{all white points of } S \text{ are above } h \end{array}$$

where $\varphi(h)$ can be any linear function over the coefficients of h's hyperplane equation.

For our problem, the dualized input becomes a non-degenerate set H of n hyperplanes. Given any point $q \in \mathbb{R}^d$ colored black or white, its *level* $\ell_H(q)$ refers to the number of hyperplanes below q if black, above q if white. Let $L_k(H)$ denote the set of all black points of level $< k$ and let $U_k(H)$ denote the set of all white points of level $< k$. Our problem corresponds to computing $w(L_k(H) \cup U_k(H))$.

Although $L_k(H)$ and $U_k(H)$ are infinite sets, it suffices to take only the vertices along their boundaries (called the *k-level vertices*). Unfortunately, even for dimension $d = 2$, the number of such vertices can be superlinear ($n2^{\Omega(\sqrt{\log k})}$) [46], and currently the best upper bound is $O(nk^{1/3})$ [20]. In practice, the number is probably smaller than this upper bound, and the obvious approach of constructing the k-level [8] and running a linear programming algorithm may not be as ineffective as one thinks (at least for $d = 2$). To design algorithms that are guaranteed to do well on any input,

however, we need to adopt the implicit LP approach from the previous section.

We now state a known geometric result that enables us to divide a problem into subproblems of size a fraction less (requirement 2). We then demonstrate how the problem can be solved by our randomized technique.

LEMMA 3.1. **(Cutting Lemma)** *Given n hyperplanes in a fixed dimension d, we can cut $\mathbb{R}^d$ into a constant number of simplices such that each simplex intersects at most $\lceil \alpha n \rceil$ hyperplanes for some constant $\alpha < 1$. The construction takes linear time.*

Proof. This was first proved by Megiddo and Dyer [21, 36] (though the result was stated differently). For instance, in dimension 2, this construction gives 4 cells with $\alpha = 7/8$ (which can be refined to $\alpha = 3/4$ with more work). A simple random sampling algorithm was suggested by Clarkson [11]. The theoretically fastest deterministic algorithm ($O(r^d)$ cells with $\alpha = 1/r$) was obtained by Chazelle [10]. □

THEOREM 3.1. *Given a number k and a non-degenerate set of n points in a fixed dimension d, we can decide whether there exists a point of Tukey depth at least k (and return such a point) in $O(n \log n + n^{d-1})$ expected time.*

Proof. We use Lemma 2.3 to solve a slight extension of the dual problem: given a simplex Δ and numbers a and b, compute $w(f(H, \Delta, a, b))$, where

$$f(H, \Delta, a, b) := (L_{k-a}(H) \cup U_{k-b}(H)) \cap \Delta.$$

We check that the requirements of the lemma can indeed be fulfilled (ignoring the trivial base cases):

1. Given input (H, Δ, a, b) of size n and a basis B, we can decide whether B satisfies $f(H, \Delta, a, b)$, i.e., whether for the hyperplane h defined by B, $L_{k-a}(H) \cap \Delta$ is (completely) below h and $U_{k-b}(H) \cap \Delta$ above h, in the following manner.

   Without loss of generality, we concentrate on the first condition. Let Δ' be the portion of Δ above h. The condition is equivalent to $L_{k-a}(H) \cap \Delta' = \emptyset$, i.e., $L_{k-a}(H) \cap \partial\Delta' = \emptyset$. We take each $(d-1)$-dimensional simplex σ of $\partial\Delta'$. Testing whether $L_{k-a}(H) \cap \sigma$ is empty reduces to finding a point inside all but $k-a$ halfspaces in $(d-1)$-dimensional space (the affine hull of σ)—the problem of "linear programming with violations" [9, 34]. In our case (where $k-a$ is possibly large), a naive approach of constructing the entire $(d-1)$-dimensional arrangement [22] is best and enough to implement the entire satisfaction test in $D(n) := O(n \log n + n^{d-1})$ time.

2. Given input (H, Δ, a, b) of size n, we can form the simplices from the cutting lemma, intersect them with Δ, and retriangulate to partition Δ into simplices $\Delta_1, \ldots, \Delta_r$. Then

$$\begin{aligned} f(H, \Delta, a, b) &= \bigcup_{i=1}^{r} f(H, \Delta_i, a, b) \\ &= \bigcup_{i=1}^{r} f(H_i, \Delta_i, a + a_i, b + b_i), \end{aligned}$$

   where H_i denotes the set of hyperplanes of H intersecting Δ_i (of size at most $\lceil \alpha n \rceil$), and a_i and b_i respectively denote the number of hyperplanes strictly below Δ_i and above Δ_i.

The theorem thus follows. □

Remark: As Stefan Langerman (personal communication) pointed out, our technique can speed up a subroutine in Matoušek's two-dimensional depth algorithm (his Lemma 3.3) [32] and result in an improved $O(n \log^2 n)$ expected time bound for constructing the entire depth-k contour. We leave open the question of whether $O(n \log n)$ time is possible for the two-dimensional depth contour problem.

3.2 Finding a point of maximum depth.

Having solved the problem of deciding whether the maximum Tukey depth is at least k, we consider the problem of computing the maximum Tukey depth. One way is to apply the randomized optimization technique again, this time in its original form (Lemma 2.1)—the application is not entirely trivial (but is doable with the help of Helly's theorem) and results in a "two-level" algorithm. We describe another way that directly modifies our previous algorithm and applies the optimization technique just once.

To this end, we consider a more general linear programming problem, where each of the constraints comes with a *label* value: maximize k such that there exists a point inside all given halfspaces with label $< k$; or in dual form, compute

$$\begin{aligned} w'(S) &:= \min(-k, \varphi(h)) \\ \text{s.t.} &\quad \text{all black points with label } < k \text{ are below } h \\ &\quad \text{all white points of } S \text{ with label } < k \text{ are above } h \end{aligned}$$

where the minimum is taken lexicographically. For example, the following is a special case (where labels are positions in a sequence): given a sequence S of linear constraints, find the longest prefix of S whose linear program is feasible. Curiously, this problem is LP-type and can thus be solved in $O(|S|)$ expected time, without any extra logarithmic factor caused by binary search:

OBSERVATION 3.2. *w' is LP-type of dimension at most $2d+1$.*

Proof. We can check properties (i)–(iii) directly, or we can just recognize that w' is an instance of *quasiconvex programming* (or "quasilinear programming"?), as defined (and shown LP-type) by Amenta *et al.* [4]: Roughly, the goal is to find a point that satisfies a set of constraints, maximizing a variable t, where each level set (points with the same t value) of each constraint is convex, and the level sets of each constraint are nested. In our case, for each point q with label k, the level set at all $t > k$ is the dual halfspace of q, and the level set at $t \leq k$ is all of $\mathbb{R}^d$. □

The maximum Tukey depth problem dualizes to finding $w'(f'(H, \Delta, a, b))$ for $\Delta = \mathbb{R}^d$ and $a = b = 0$, where

$$\begin{aligned} f'(H, \Delta, a, b) = {} & \{q \in \Delta \text{ in black with label } \ell_H(q) + a\} \\ & \cup \{q \in \Delta \text{ in white with label } \ell_H(q) + b\}. \end{aligned}$$

THEOREM 3.2. *Given a non-degenerate set of n points in a fixed dimension d, we can compute the maximum Tukey depth (and return a Tukey median) in $O(n \log n + n^{d-1})$ expected time.*

Proof.

1. Deciding whether a basis satisfies $f'(H, \Delta, a, b)$ with respect to w' again reduces to testing, for a given hyperplane h and a given number k, whether $L_{k-a}(H) \cap \Delta$ is below h and $U_{k-b}(H) \cap \Delta$ is above h. As in the proof of the previous theorem, this requires $D(n) = O(n \log n + n^{d-1})$ time.

2. Forming the simplices Δ_i, sets H_i, and indices a_i and b_i in the same fashion as in the previous proof, we similarly have

$$\begin{aligned} f'(H, \Delta, a, b) &= \bigcup_{i=1}^{r} f'(H, \Delta_i, a, b) \\ &= \bigcup_{i=1}^{r} f'(H_i, \Delta_i, a + a_i, b + b_i). \end{aligned}$$

The requirements of Lemma 2.3 are thus met, and the expected time bound is asymptotically the same. □

Remark: It would be interesting to see whether our technique can be made practical, considering that initial estimates for the hidden constants seem huge.

Acknowledgements

I thank Stefan Langerman for re-posing the problem at the 2002 Fall Workshop on Computational Geometry problem session, and for subsequent discussions.

References

[1] P. K. Agarwal and M. Sharir. Efficient algorithms for geometric optimization. *ACM Comput. Surveys*, 30:412–458, 1998.

[2] G. Aloupis, C. Cortes, F. Gomez, M. Soss, and G. T. Toussaint. Lower bounds for computing statistical depth. *Computational Statistics and Data Analysis*, 40:223–229, 2002.

[3] G. Aloupis, S. Langerman, M. Soss, and G. Toussaint. Algorithms for bivariate medians and a Fermat-Torricelli problem for lines. *Comput. Geom. Theory Appl.*, 26:69–79, 2003.

[4] N. Amenta, M. Bern, and D. Eppstein. Optimal point placement for mesh smoothing. *J. Algorithms*, 30:302–322, 1999.

[5] M. Bern and D. Eppstein. Multivariate regression depth. *Discrete Comput. Geom.*, 28:1–17, 2002.

[6] T. M. Chan. Fixed-dimensional linear programming queries made easy. In *Proc. 12th ACM Sympos. Comput. Geom.*, pages 284–290, 1996.

[7] T. M. Chan. Geometric applications of a randomized optimization technique. *Discrete Comput. Geom.*, 22:547–567, 1999.

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

[9] T. M. Chan. Low-dimensional linear programming with violations. In *Proc. 43rd IEEE Sympos. Found. Comput. Sci.*, pages 570–579, 2002.

[10] B. Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete Comput. Geom.*, 9:145–158, 1993.

[11] K. L. Clarkson. New applications of random sampling in computational geometry. *Discrete Comput. Geom.*, 2:195–222, 1987.

[12] K. L. Clarkson. Las Vegas algorithms for linear and integer programming when the dimension is small. *J. ACM*, 42:488–499, 1995.

[13] K. L. Clarkson. Algorithms for the minimum diameter of moving points and for the discrete 1-center problem. Manuscript, `http://cm.bell-labs.com/who/clarkson/moving_diam.html`, 1997.

[14] K. L. Clarkson, D. Eppstein, G. L. Miller, C. Sturtivant, and S.-H. Teng. Approximating center points with iterated Radon points. *Int. J. Comput. Geom. Appl.*, 6:357–377, 1996.

[15] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4:387–421, 1989.

[16] E. Cohen and N. Megiddo. Strongly polynomial-time and NC algorithms for detecting cycles in dynamic graphs. *J. ACM*, 40:791–830, 1993.

[17] R. Cole. Slowing down sorting networks to obtain faster sorting algorithms. *J. ACM*, 34:200–208, 1987.

[18] R. Cole, M. Sharir, and C. K. Yap. On k-hulls and related problems. *SIAM J. Comput.*, 16:61–77, 1987.

[19] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. McGraw-Hill, 2nd ed., 2001.

[20] T. K. Dey. Improved bounds on planar k-sets and related problems. *Discrete Comput. Geom.*, 19:373–382, 1998.

[21] M. E. Dyer. Linear time algorithms for two- and three-variable linear programs. *SIAM J. Comput.*, 13:31–45, 1984.

[22] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag, Berlin, 1987.

[23] D. Eppstein. Setting parameters by example. *SIAM J. Comput.*, 82:638–653, 2003.

[24] J. Erickson. New lower bounds for convex hull problems in odd dimensions. *SIAM J. Comput.*, 28:1198–1214, 1999.

[25] J. Gill, W. Steiger, and A. Wigderson. Geometric medians. *Discrete Math.*, 108:37–51, 1992.

[26] P. Gupta, R. Janardan, and M. Smid. Fast algorithms for collision and proximity problems involving moving geometric objects. *Comput. Geom. Theory Appl.*, 6:371–391, 1996.

[27] S. Jadhav and A. Mukhopadhyay. Computing a centerpoint of a finite planar set of points in linear time. *Discrete Comput. Geom.*, 12:291–312, 1994.

[28] M. van Kreveld, J. S. B. Mitchell, P. Rousseeuw, M. Sharir, J. Snoeyink, and B. Speckmann. Efficient algorithms for maximum regression depth. In *Proc. 15th ACM Sympos. Comput. Geom.*, pages 31–40, 1999.

[29] S. Langerman and W. Steiger. Computing a maximal depth point in the plane. *Japan Conf. Discrete Comput. Geom.*, 2000.

[30] S. Langerman and W. Steiger. The complexity for hyperplane depth in the plane. In *Proc. 11th ACM-SIAM Sympos. Discrete Algorithms*, pages 54–59, 2000. *Discrete Comput. Geom.*, to appear.

[31] S. Langerman and W. Steiger. Optimization in arrangements. In *Proc. 20th Sympos. Theoret. Aspects Comput. Sci.*, Lect. Notes in Comput. Sci., vol. 2607, Springer-Verlag, pages 50–61, 2003.

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

[33] J. Matoušek. Linear optimization queries. *J. Algorithms*, 14:432–448, 1993. Also with O. Schwarzkopf in *Proc. 8th ACM Sympos. Comput. Geom.*, pages 16–25, 1992.

[34] J. Matoušek. On geometric optimization with few violated constraints. *Discrete Comput. Geom.*, 14:365–384, 1995.

[35] N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *J. ACM*, 30:852–865, 1983.

[36] N. Megiddo. Linear programming in linear time when the dimension is fixed. *J. ACM*, 31:114–127, 1984.

[37] K. Miller, S. Ramaswami, P. Rousseeuw, D. Souvaine, T. Sellares, I. Streinu and A. Struyf. Efficient computation of location depth contours by methods of computational geometry. *Statistics and Computing*, 13:153–162, 2003. Preliminary version in *Proc. 11th ACM-SIAM Sympos. Discrete Algorithms*, pages 153–162, 2001.

[38] N. Naor and M. Sharir. Computing a point in the center of a point set in three dimensions. In *Proc. 2nd Canad. Conf. Comput. Geom.*, pages 10–13, 1990.

[39] C. H. Norton, S. A. Plotkin and E. Tardos. Using separation algorithms in fixed dimension. *J. Algorithms*, 13:79–98, 1992.

[40] E. A. Ramos. Linear programming queries revisited. In *Proc. 16th ACM Sympos. Comput. Geom.*, pages 176–181, 2000.

[41] P. J. Rousseeuw and I. Ruts. Constructing the bivariate Tukey median. *Statistica Sinica*, 8:827–839, 1998.

[42] I. Ruts and P. J. Rousseeuw. Computing depth contours of bivariate point clouds. *Computational Statistics and Data Analysis*, 23:153–168, 1996.

[43] R. Seidel. Small-dimensional linear programming and convex hulls made easy. *Discrete Comput. Geom.*, 6:423–434, 1991.

[44] M. Sharir and E. Welzl. A combinatorial bound for linear programming and related problems. In *Proc. 9th Sympos. Theoret. Aspects Comput. Sci.*, Lect. Notes in Comput. Sci., vol. 577, Springer-Verlag, pages 569–579, 1992.

[45] C. G. Small. A survey on multidimensional medians. *Internat. Statist. Rev.*, 58:263–277, 1990.

[46] G. Toth. Point sets with many k-sets. *Discrete Comput. Geom.*, 26:187–194, 2001.

[47] J. Tukey. Mathematics and the picturing of data. In *Proc. Int. Congress of Mathematicians*, pages 2:523–531, 1975.