Computational Geometry 43 (2010) 647–654

Contents lists available at ScienceDirect

Computational Geometry: Theory and Applications

www.elsevier.com/locate/comgeo

Approximate centerpoints with proofs ☆

Gary L. Miller, Donald R. Sheehy *

Computer Science Department, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, United States

ARTICLE INFO

Article history:
Received 7 July 2009
Received in revised form 16 January 2010
Accepted 28 April 2010
Available online 4 May 2010
Communicated by A. Zomorodian

ABSTRACT

We present the IteratedTverberg algorithm, the first deterministic algorithm for computing an approximate centerpoint of a set $S \subset \mathbb{R}^d$ with running time sub-exponential in d. The algorithm is a derandomization of the IteratedRadon algorithm of Clarkson et al. (International Journal of Computational Geometry and Applications 6 (3) (1996) 357–377) and is guaranteed to terminate with an $\Omega(1/d^2)$-center. Moreover, it returns a polynomial-time checkable proof of the approximation guarantee, despite the coNP-completeness of testing centerpoints in general. We also explore the use of higher order Tverberg partitions to improve the running time of the deterministic algorithm and improve the approximation guarantee for the randomized algorithm. In particular, we show how to improve the $\Omega(1/d^2)$-center of the IteratedRadon algorithm to $\Omega(1/d^{\frac{r}{r-1}})$ for a cost of $O((rd)^d)$ in time for any integer $r > 1$.

1. Introduction

A centerpoint of a set S of n points in $\mathbb{R}^d$ is a point c such that every closed half-space containing c also contains at least $\frac{n}{d+1}$ points of S. Intuitively, every hyperplane through c divides S into roughly equal parts. The Centerpoint Theorem asserts that every set of n points in $\mathbb{R}^d$ has a centerpoint. This was first established by a theorem of Rado [14] for general measures and by Danzer et al. [4] for the special case of point sets.

A centerpoint is a natural generalization of the median to higher dimensions. In fact, a centerpoint is an approximate median of the set $f(S)$ for every linear projection $f : S \to \mathbb{R}$. They are used as robust estimators in statistics, because they are invariant under affine transformation and robust to outliers [5]. They are also used in mesh partitioning [7].

The *Tukey depth* of a point x is the minimum number of points of S lying in a half-space that contains x. Thus, a centerpoint is a point with Tukey depth at least $\frac{n}{d+1}$. More generally, c is a *β-center* if it has depth at least βn. We consider the problem of finding an approximate centerpoint, giving a deterministic algorithm for computing a $\frac{1}{2(d+1)^2}$-center.

The existence of centerpoints can be proven directly from either of two classic theorems of convexity theory, Helly's Theorem and Tverberg's Theorem. In Section 3, we discuss how these two proofs of the centerpoint theorem lead to different perspectives for designing algorithms for computing approximate centerpoints.

The exact complexity of computing centerpoints in higher dimensions is not known. The decision problem of testing if a given point is a centerpoint is coNP-complete [16]. However, a simple corollary of Tverberg's Theorem guarantees the existence of a subset of centerpoints, call them Tverberg points, that admit polynomial-time checkable proofs. Moreover, testing if a point is a Tverberg point is NP-complete [16]. In this case, the decision problem is well understood but sheds little light on the hardness of the search problem of actually finding a centerpoint.

☆ This work was supported in part by the National Science Foundation under Grant CCF-0635257.
* Corresponding author.
E-mail address: dsheehy@cs.cmu.edu (D.R. Sheehy).

0925-7721/$ – see front matter
doi:10.1016/j.comgeo.2010.04.006

The fastest known algorithm for computing a centerpoint of $S \subset \mathbb{R}^d$ is due to Chan [1] and computes a β-center in time $O(n^{d-1})$ in expectation, where β is the maximum achievable for the set S. Such a β-center is also known as a Tukey median.

The IteratedRadon algorithm of Clarkson et al. was the first algorithm that computes an approximate centerpoint in time sub-exponential in d [2]. The algorithm computes an $\Omega(1/d^2)$-center with high probability. Section 3 describes how this algorithm resembles the proof of the Centerpoint Theorem via Helly's Theorem.

The main operation in the IteratedRadon algorithm is to replace sets of points by their Radon point, a point in the common intersection of the convex hull of two disjoint subsets. Radon's Theorem guarantees the existence of such a point. Tverberg's Theorem is a generalization of Radon's Theorem that guarantees a common intersection for a larger collection of subsets.

In this paper, we use the intuition from Tverberg's Theorem to iteratively construct a proof of depth for an approximate centerpoint. The result is a new approximation algorithm, IteratedTverberg, that derandomizes the IteratedRadon algorithm of Clarkson et al. In Section 4, we prove that the IteratedTverberg algorithm produces an $\Omega(1/d^2)$-center in time sub-exponential in d. Moreover, the center comes with a polynomial-time checkable proof of its depth.

We elaborate on this intuition in Section 5, showing how solving larger sub-problems can be used to speed up the running time of the deterministic algorithm and to improve the approximation ratio of the randomized version.

2. Related work

Centerpoints are the most well known definition of a geometric median [6]. Like many such medians, it can be computed via linear programming and the problem of finding a "best" centerpoint can be written as a maximum feasible subsystem problem (see [5] for a survey of computational aspects of data depth). The linear constraints are just the set of hyperplanes intersecting $d+1$ affinely independent points of S. There are $O(n^{d+1})$ of these. Consequently, any linear programming method will require time $n^{O(d)}$, limiting their usefulness to low-dimensional instances.

In the plane, centerpoints can be computed in linear time [9]. Several algorithms are known to compute centerpoints in $\mathbb{R}^3$ in $O(n^2 \operatorname{polylog} n)$ time [3,13]. The best known algorithm for $d \geqslant 3$ is due to Chan and runs in $O(n^{d-1})$ expected time [1]. Chan's algorithm computes the deepest possible centerpoint, also known as a Tukey median. He conjectures that the $O(n^{d-1})$ running time is optimal for this problem in the algebraic decision tree model. However, the exact complexity of computing centerpoints is not known. In particular, it is not known if it is possible to compute a centerpoint in time polynomial in n and d.

Several approximation algorithms for centerpoints exist in the literature. Many approaches using random sampling are known [11,16,2]. Verbarg showed that for dense point sets, the mean is a good approximate centerpoint [18]; it is a β-center where β depends on the density.

The only previously known algorithm to compute an approximate centerpoint in time sub-exponential in d is the IteratedRadon algorithm of Clarkson et al. [2]. The IteratedRadon algorithm returns an $\Omega(1/d^2)$-center with high probability in time polynomial in n and d. IteratedRadon is a Monte Carlo algorithm and there is no way known to verify that the point returned by the algorithm is indeed an $\Omega(1/d^2)$-center. The inner loop of IteratedRadon depends on the following classic theorem [15].

Theorem 2.1 *(Radon's Theorem, 1921). Given $n > d+1$ points $S \subset \mathbb{R}^d$, there exists a partition $(U, \overline{U})$ of S such that* $\operatorname{conv}(U) \cap \operatorname{conv}(\overline{U}) \neq \emptyset$.

We call a partition of $d+2$ points as described in the theorem, a *Radon partition*, and we call a point in the intersection, a *Radon point*.

The simplest version of the IteratedRadon algorithm works as follows. Build a balanced $(d+2)$-ary tree of height h. Fill in the leaves with points from the input set S by sampling them uniformly at random. Each interior node of the tree is filled in with the Radon point of its children. Clarkson et al. show that a height of $h = \lg n$ is needed to compute an $\Omega(1/d^2)$-center with high probability, resulting in a running time that is $O((d \lg n)^{\lg d})$ [2]. Thus it is sub-exponential in d but not polynomial.

Several different modifications to the basic IteratedRadon algorithm are presented in [2]. In one version, the running time is reduced to $O(\mathit{poly}(n,d))$ by reusing the Radon points the same way the input points are reused, but this bound hides the dependence on δ, the probability guarantee that the output is correct. Other variations of the algorithm incorporate linear programming to solve larger subproblems. These variants are able to produce $\frac{1+\varepsilon}{d+1}$-centers with some constant probability.

It is not known how to derandomize the other versions of IteratedRadon. We do, however, explore a variation on the linear programming based solution. We present a new way to leverage these larger subproblems and analyze the impact on both the randomized and the deterministic algorithms (see Section 5).

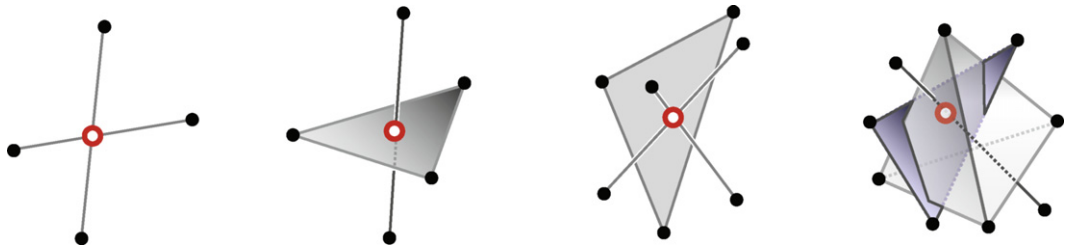

Fig. 1. Radon points for $\mathbb{R}^2$ and $\mathbb{R}^3$. Tverberg points for $r = 3$ in $\mathbb{R}^2$ and $\mathbb{R}^3$.

3. Two proofs of the Centerpoint Theorem

Theorem 3.1 *(The Centerpoint Theorem (Rado [14], Danzer et al. [4])). Given a set of n points $S \subset \mathbb{R}^d$, there exists a centerpoint $c \in \mathbb{R}^d$ such that every closed half-space containing c also contains at least $\frac{n}{d+1}$ points of S.*

The Centerpoint Theorem is most often presented as an easy consequence of Helly's Theorem. It is also possible to prove the existence of centerpoints via Tverberg's Theorem (see [12] for a comprehensive treatment). The relationship between these two proofs gives insight into the relationship between the IteratedRadon algorithm and its derandomization presented in this paper.

Theorem 3.2 *(Helly's Theorem [8], 1913). Given a collection of compact, convex sets $X_1, \ldots, X_n \subset \mathbb{R}^d$. If every $d+1$ of these sets have a common intersection, then the whole collection has a common intersection.*

The Centerpoint Theorem follows from Helly's Theorem as follows. Consider the set H of all open half-spaces that contain more than $\frac{dn}{d+1}$ points of S. For each such half-space $h \in H$, let P_h denote $\operatorname{conv}(S \cap h)$, a compact, convex set containing the same points of S as h. Any $d+1$ of the half-spaces have a common intersection at one of the points of S, and thus every $d+1$ of the P_h's also have a common intersection. We apply Helly's Theorem to the family of sets $P_H = \{P_h \mid h \in H\}$. The common intersection guaranteed by Helly's Theorem is exactly the set of all centerpoints.

The most common elementary proof of Helly's Theorem makes extensive use of Radon's Theorem, despite that Helly's Theorem technically came first (though published second). The proof first considers the case where there are only $d+2$ sets. The hypothesis of the theorem implies the existence of $d+2$ points, each taken from the common intersection of $d+1$ of the sets. The Radon point of these $d+2$ points satisfies the conclusion of the theorem. The proof for $n > d+2$ sets uses induction. The inductive step again considers a set of points taken from each of the common intersections of $n-1$ sets, and shows the Radon point of this set satisfies the theorem. Unraveling this induction into an algorithm, we arrive at something very much like the IteratedRadon algorithm except with far too many Radon point computations to be computationally feasible. This is why IteratedRadon uses random sampling used to avoid paying for the combinatorial blowup in the number of sets. The IteratedTverberg algorithm we present shows how to make this sampling deterministic (Section 4).

The Centerpoint Theorem can also be proven via Tverberg's generalization of Radon's Theorem.

Theorem 3.3 *(Tverberg's Theorem [17], 1966). Given $(d+1)(r-1)+1$ points $S \subset \mathbb{R}^d$, there exists a partition of S into $S_1, \ldots, S_r$, such that $\bigcap_{i=1}^{r} \operatorname{conv}(S_i) \neq \emptyset$.*

Observe that Radon's Theorem is a special case of Tverberg's Theorem when $r = 2$.

Say that a point c is a Tverberg point if it is in the common intersection of the convex hulls in the Tverberg partition. Then, setting $r = \lceil \frac{n}{d+1} \rceil$ yields a Tverberg point contained in the convex hull of $\lceil \frac{n}{d+1} \rceil$ pairwise disjoint subsets of S. Any half-space containing c must also contain at least one point from each of the subsets and therefore, c is a centerpoint.

Observe that the centerpoints guaranteed by Tverberg's Theorem come equipped with a polynomial-time checkable proof. Given the partition, we need only verify that the point is in the convex hull of each part. If any part in the partition has more than $d+1$ points then by Carathéodory's Theorem, there is a subset of size $d+1$ that contains the Tverberg point is its convex hull. We may therefore assume the convex hulls are simplices of dimension at most d, so checking can be done quickly. The key insight in derandomizing the IteratedRadon algorithm is to actively construct these Tverberg partitions for the intermediate points used in the algorithm.

4. Derandomizing the IteratedRadon algorithm

4.1. Short proofs of depth

Given a point $p \in \mathbb{R}^d$, a *proof* that p has Tukey depth (at least) r with respect to S is a collection of r disjoint subsets $U_1, \ldots, U_r \subset S$ such that $p \in \bigcap_{j=1}^{r} \operatorname{conv}(U_j)$. If there is such a proof, we say p has *depth* r. For any input point $s \in S$, the

set $\{s\}$ is a proof of depth 1. A point and its proof of depth is denoted by $\langle point, proof\rangle$. We say that a collection of proofs are disjoint if their sets are pairwise disjoint.

The IteratedTverberg algorithm is very similar to the IteratedRadon algorithm. The key difference is that each successive approximation carries with it a proof of its depth. When we combine $d+2$ points of depth r into a Radon point c, we can rearrange the proofs to get a new proof that c has depth $2r$ as shown in the following lemma.

Lemma 4.1. *Given a set P of $d+2$ points with disjoint proofs of depth r, the Radon point of P has depth at least $2r$.*

Proof. Let (P_1, P_2) be the Radon partition for P, and let c be the Radon point. For each $p_i \in P$, order the parts in the proof partition of p_i and call the jth part $U_{i,j}$. We build a proof that c has depth at least $2r$ with parts of the form $\bigcup_{p_i \in P_k} U_{i,j}$ for $k \in \{1,2\}$ and $j \in \{1,\ldots,r\}$. These sets are disjoint by our hypothesis. Since each $p_i \in P_k$ is contained in $\operatorname{conv}(U_{i,j})$, it follows that $P_k \subset \operatorname{conv}(\bigcup_{p_i \in P_k} U_{i,j})$ and also, by standard convexity arguments, $\operatorname{conv}(P_k) \subseteq \operatorname{conv}(\bigcup_{p_i \in P_k} U_{i,j})$. By the definition of Radon points, $c \in \operatorname{conv}(P_k)$. Therefore, $c \in \operatorname{conv}(P_k) \subseteq \operatorname{conv}(\bigcup_{p_i \in P_k} U_{i,j})$ for all $2r$ choices of $k \in \{1,2\}$ and $j \in \{1,\ldots,r\}$. □

The preceding lemma implies a simple deterministic algorithm for computing an approximate centerpoint. Construct a $(d+2)$-ary tree with n leaves. Store the points of S in the leaves. In each interior node of the tree, store the Radon point of its children. The height of the tree is $\log_{d+2} n$, so Lemma 4.1 implies that the depth of the root is $2^{\log_{d+2} n} = O(n^{1/\lg(d+2)})$. Not too shabby for such a simple algorithm, but the depth guarantee of the output is only sub-linear in n. To get an $\Omega(1/d^2)$-center, we need to find a way to build this tree higher, and in order to do that, we need more leaves. The following lemma gives a hint as to where we can look for more points to put in the leaves.

Lemma 4.2. *If there is a proof that a point p has depth r, there exists such a proof that contains at most $r(d+1)$ points of S.*

Proof. Let $P_1, \ldots, P_r$ be the sets in the proof for p. By Carathéodory's Theorem, there exists a subset $P_i' \subset P_i$ of at most $d+1$ points such that $p \in \operatorname{conv}(P_i')$. So, the sets $P_1', \ldots, P_r'$ is the desired proof of the correct size. □

We refer to this economizing of proofs as *pruning*. A standard proof of Carathéodory's Theorem implies a reasonable algorithm for pruning a proof. Starting with the point of interest c as a convex combination of k points (where $k > d+1$), find a new convex combination that uses one less point by subtracting a multiple of an affine dependence of the points. The choice of the affine dependences does not matter so we may find one by solving a linear system on any subset of $d+2$ points in $O(d^3)$ time. We then iterate this procedure until we have c as a convex combination of just $d+1$ points.

As in Lemma 4.1, a set to be pruned is the union of one set from each proof of each point in one part of a Radon partition. If the proofs to be combined were already pruned, and thus contained only sets of size $d+1$, then the total number of points in the combined sets is at most $(d+1)^2$. Consequently, the pruned set can be found by after $O(d^2)$ iterations. The entire pruning operation takes $O(d^5)$ time.

4.2. The deterministic algorithm

We are now ready to describe the IteratedTverberg algorithm. We make buckets B_ℓ to store $\langle point, proof\rangle$ pairs for points of depth 2^ℓ. At the start of the algorithm, the input points $s \in S$ are placed in bucket B_0 with the trivial proof of depth 1. The main loop of the algorithm takes any bucket B_ℓ with a least $d+2$ points $q_1, \ldots, q_{d+2}$ and combines them as in Lemma 4.1 to form a proof of depth $2^{\ell+1}$ for the Radon point of $q_1, \ldots, q_{d+2}$. We then prune the proof as in Lemma 4.2 and push it into bucket $B_{\ell+1}$. The loop terminates when we get a point in bucket B_z, where $z = \lg\lceil \frac{n}{2(d+1)^2} \rceil$. A single iteration of the main loop is illustrated in Fig. 2 and pseudocode is provided in Algorithm 1.

4.3. Theoretical guarantees

Theorem 4.3. *The* IteratedTverberg *algorithm returns a point of depth at least $\lceil \frac{n}{2(d+1)^2} \rceil$.*

Proof. Clearly, if the algorithm terminates then it returns a point of the desired depth and its corresponding proof. Suppose for contradiction that the algorithm does not terminate. Then, at some time t, it is impossible to pick i because every bucket has fewer than $d+2$ points. Let h be the maximum such that B_h is nonempty at time t. For any point p in a bucket B_i, the proof for p contains at most $2^i(d+1)$ input points. Since all of the proofs in all of the buckets are disjoint, we have

$$\sum_{i=0}^{h} 2^i (d+1)^2 \geqslant n. \tag{1}$$

Therefore,

$$2^{h+1} - 1 \geqslant \frac{n}{(d+1)^2}, \tag{2}$$

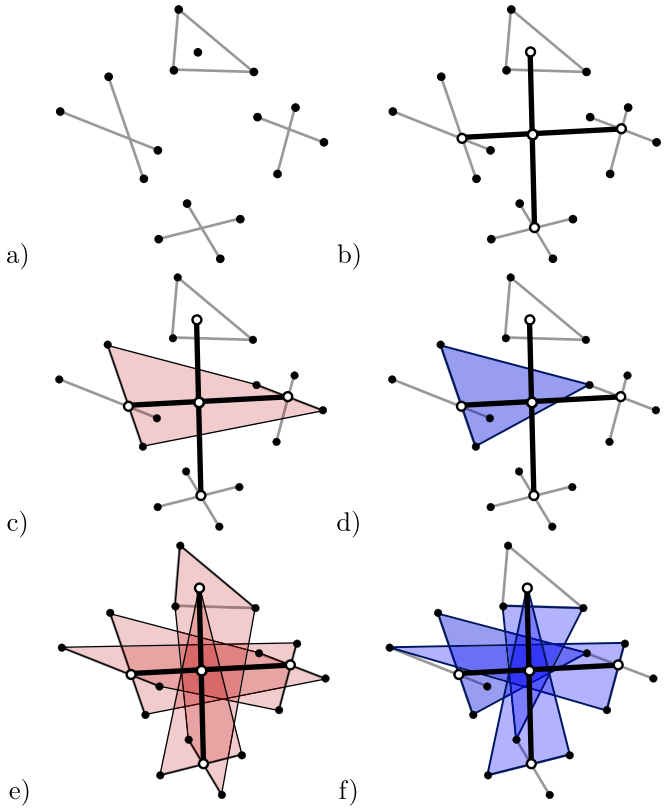

Fig. 2. The IteratedTverberg algorithm: (a) Sets of $d+2$ points are divided into Radon partitions. (b) $d+2$ Radon points are combined into a second-order Radon partition. (c) A proof polygon is formed by taking the convex hull of two subsets, one from each of the Radon points in the second order partition. (d) The proof polygon is pruned to a simplex. (e) All of the proof polygons before the pruning phase. (f) The proof simplices after the pruning.

Algorithm 1 IteratedTverberg

Input: $S \in \mathbb{R}^d : |S| = n$
Initialize empty stacks $B_0, \ldots, B_{\lg\lceil \frac{n}{2(d+1)^2} \rceil}$.
Push $\langle s, \{s\}\rangle$ to B_0 for each $s \in S$
while $B_{\lg\lceil \frac{n}{2(d+1)^2} \rceil}$ is empty **do**
  Initialize *proof* to be an empty stack
  Let ℓ be the max such that $B_{\ell-1}$ has at least $d+2$ points
  Pop $d+2$ points $q_1, \ldots, q_{d+2}$ from $B_{\ell-1}$
  Let $\langle c, \{U_1, U_2\}\rangle = \textsc{Radon}(q_1, \ldots, q_{d+2})$
  for $k = 1, 2$ **do**
    for $i = 1, \ldots, 2^{\ell-1}$ **do**
      Let S_{ij} be the ith part of the proof for q_j
      Let $X = \bigcup_{q_j \in U_k} S_{ij}$
      Let $X' = \textsc{Prune}(c, X)$
      Push X' to *proof*
      Push $(\langle s, \{s\}\rangle)$ to B_0 for each $x \in X \setminus X'$
    end for
  end for
  Push $\langle c, \textit{proof}\rangle$ to B_ℓ
end while
return $\langle c, \textit{proof}\rangle$ from $B_{\lg\lceil \frac{n}{2(d+1)^2} \rceil}$

and thus,

$$2^h > \frac{n}{2(d+1)^2}. \tag{3}$$

However, this is impossible because it implies that $h > \lg \frac{n}{2(d+1)^2}$, and therefore $B_{\lg\lceil \frac{n}{2(d+1)^2} \rceil}$ is nonempty. This would have caused the algorithm to break out of the while loop and terminate, contrary to our assumption. □

Theorem 4.4. *The* IteratedTverberg *algorithm requires* $n^{O(\lg d)}$ *time.*

Proof. The main loop of the algorithm increases the size of some B_ℓ by one and decreases the size of $B_{\ell-1}$ by $d+2$. It requires a Radon partition computation and 2^ℓ pruning operations that dominate the running time of the inner loop.

As mentioned previously, the Radon partition can be found in $O(d^3)$ time, and the pruning takes $O(d^5)$ time. Since $\ell \leqslant \lg \frac{n}{2(d+1)^2}$, these two operations require $O(d^3) + 2^{\lg \frac{n}{2(d+1)^2}} O(d^5) = O(nd^3)$ time.

We will count the number of iterations of the main loop, charging the iterations for a point that was popped from a bucket to the point created in that iteration. Let the *cost* of point in bucket B_ℓ be defined inductively as one more than the sum of the costs of the $d+2$ points that have been popped from $B_{\ell-1}$. Since the cost of a point in B_ℓ is determined by ℓ, it can be written as a simple recurrence,

$$t_\ell = (d+2)t_{\ell-1} + 1. \tag{4}$$

In the base case $t_0 = 0$, because input points have trivial proofs of depth 1. It follows that

$$t_\ell < (d+2)^\ell. \tag{5}$$

Let $T_{\mathcal{B}}$ be the total number of iterations of the main loop given buckets of sizes $\mathcal{B} = \{|B_0|, \ldots, |B_m|\}$. The value of $T_{\mathcal{B}}$ can be computed exactly by summing the costs of the points remaining in buckets.

$$T_{\mathcal{B}} = \sum_{i=1}^{m} |B_i| t_i.$$

At termination, no bucket has size greater then $d+1$. So, if $\mathcal{B}$ is the set of final bucket sizes, then

$$T_{\mathcal{B}} \leqslant \sum_{i=1}^{m} (d+1)t_i < t_{m+1}. \tag{6}$$

Eqs. (5) and (6) imply the $T_{\mathcal{B}} < (d+2)^{m+1}$. By Theorem 4.3, the final depth is $\frac{n}{2(d+1)^2}$, so we are interested in $m = \lfloor \lg \frac{n}{2(d+1)^2} \rfloor$. The total number of iterations of the main loop can now be bounded as follows.

$$T_{\mathcal{B}} < (d+2)^{(\lg \frac{n}{2(d+1)^2})+1} \tag{7}$$

$$= \left(\frac{n}{(d+1)^2}\right)^{\lg(d+2)} \tag{8}$$

$$= n^{O(\lg d)} \tag{9}$$

Since each iteration of the main loop requires $O(nd^3)$ time and the number of iterations is $n^{O(\log d)}$, the total running time is $n^{O(\log d)}$. □

5. Leveraging larger subproblems

The Radon point of a set of $d+2$ points has Tukey depth at least 2 and, thus, is a centerpoint for that set. Both the IteratedRadon algorithm and the IteratedTverberg algorithm replace sets of $d+2$ points with their centerpoint using Radon's Theorem. In this section we address the effect on these algorithms if we instead solve larger subproblems. Rather than combining points in sets of $d+2$, we look at sets of size $(d+1)(r-1)+1$ for some fixed r. The centerpoint of such a set has Tukey depth at least r. It is not known how to solve these larger problems in time sub-exponential in d. However, if n is large and d is not too large, it may be feasible to solve subproblems in $O(d^d)$ time even though $O(n^d)$ is prohibitive. In fact, we will see that this can even lead to an asymptotic speedup for the IteratedTverberg algorithm.

5.1. Improving the approximation for the IteratedRadon algorithm

The Radon point of $d+2$ points is a centerpoint of the subset. Consider the following modified version of the IteratedRadon algorithm.

We can run the same iterative algorithm as before except using r-partitions instead of 2-partitions. In fact, it is not necessary to keep around the partition, it actually suffices just to find any centerpoint of $(d+1)(r-1)+1$ points at each round. We can go through the analysis from [2] and see the impact of r in the depth guarantee of the approximate centerpoint.

The analysis works by looking at any projection of the point set to the line. We compute the probability $f_h(x)$ that the tree of iterations with height h returns a center of depth at most nx. Without loss of generality, the projections of the points of S land on $\frac{1}{n}, \frac{2}{n}, \ldots, 1$. It follows that $f_0(x) \leqslant x$. The center of the center will be x where x is such that $f_h(x)$ is very small.

At each iteration, the centerpoint is at least r deep in the projection. There are $\binom{(d+1)(r-1)+1}{r}$ choices for the r points less than the centerpoint in the projection. The probability of the union over all these choices is bounded from above by the sum of the individual probabilities, and thus,

$$f_h(x) \leqslant \binom{(d+1)(r-1)+1}{r} f_{h-1}(x)^r. \tag{10}$$

Let $\beta = \binom{(d+1)(r-1)+1}{r}^{-1}$.

$$\begin{aligned} f_h(x) &\leqslant \beta^{-1} f_{h-1}(x)^r && (11)\\ &\leqslant \beta^{-1}\left(\beta^{-r} f_{h-2}(x)^{r^2}\right) && (12)\\ &\leqslant \beta^{-1}\beta^{-r}\cdots\beta^{-r^{h-1}} f_0(x)^{r^h} && (13)\\ &\leqslant \beta^{\frac{1-r^h}{r-1}} x^{r^h} && (14)\\ &\leqslant \beta^{\frac{1}{r-1}}\left(\frac{x}{\beta^{\frac{1}{r-1}}}\right)^{r^h}. && (15) \end{aligned}$$

Now, since $\beta = O((rd)^{-r})$, we can choose x smaller than $O((rd)^{\frac{-r}{r-1}})$ and the probability $f_h(x)$ vanishes as desired. So, even for a choice of $r = 3$, we can improve the depth guarantee of the resulting centerpoint by an $O(\sqrt{d})$ factor.

5.2. Speeding up the IteratedTverberg algorithm

The same trick of solving larger subproblems can speed up the running time of the deterministic algorithm. Tverberg's Theorem guarantees the existence of a partition of S into r sets whose convex hulls have a common intersection as long as $|S| > (d+1)(r-1)+1$. Say $T(r)$ is the time required to compute a Tverberg partition into r parts. To the best of our knowledge, nothing better than brute force is known for computing Tverberg partitions for $r > 2$.

A slight modification to the IteratedTverberg algorithm to use Tverberg r-partitions instead of Radon partitions results in an $n^{\lg r}/T(r)$ speedup. Thus, for n large enough, we get an asymptotic speedup.

The modified algorithm simply makes recursive calls on sets of $\lceil n/r \rceil$ points and combines them in sets of $(r-1)(d+1)+1$. The analysis is virtually identical to the original version except we give up a factor of $r/2$ in the depth of the output. As for the running time, the new algorithm now has a recursion tree with higher fan out and the resulting running time is $O((d+2)^{\log_r n} T(r)) = O(n^{\lg(d+2)/\lg r} T(r))$.

6. Conclusions and open problems

We have presented the IteratedTverberg algorithm, the first algorithm that deterministically computes an approximate centerpoint in time sub-exponential in d. By combining intuition from both Helly's Theorem and Tverberg's Theorem, our method sheds an interesting new light on the problem of computing approximate centerpoints. It still remains open whether it is possible to compute approximate centerpoints deterministically in time polynomial in n and d. We conjecture that it is.

We also extended both our algorithm and the IteratedRadon algorithm by looking at the impact of solving larger subproblems. One consequence of this work is that any new results on quickly computing centerpoints for small point sets can be used to improve these algorithms. Currently, it is not known how to compute centerpoints of more than $2d+2$ points in time polynomial in d. However, we conjecture that computing the centerpoint of $2d+3$ points in $\mathbb{R}^d$ is NP-hard.

In the IteratedRadon algorithm, it was not clear if the factor of d appearing in the output depth approximation ratio was intrinsic to the problem or merely an artifact of the analysis. This same factor of d shows up in our analysis of the IteratedTverberg algorithm in a completely different way, implying that perhaps it is intrinsic to the problem. Finding the optimal approximation ratio that can be achieved by a deterministic algorithm in sub-exponential time remains an open problem.

It is also open as to whether the IteratedTverberg algorithm can be significantly sped up by doing the pruning step lazily. In such a variant, the "extra" points that get pruned out would be points with proofs of their own, possibly allowing the reuse of more computation. Thus far we have been unable to analyze the lazy pruning version of the algorithm.

The reader is directed to Kalai's survey for many other interesting problems related to Tverberg points [10].

The computation of centerpoints draws a compelling correspondence between fundamental theorems in convexity theory, Helly's Theorem and Tverberg's Theorem, and fundamental complexity classes of NP and coNP. It is our hope that future work will further elucidate this correspondence.

Acknowledgements

We would like to thank Richard Peng, Todd Philips, Venkat Guruswami, and Danny Sleator for their helpful feedback in the preparation of this work. We would also like to thank the anonymous reviewers for their helpful comments.

References

[1] T. Chan, An optimal randomized algorithm for maximum Tukey depth, in: SODA: ACM-SIAM Symposium on Discrete Algorithms, 2004, pp. 430–436.
[2] K. Clarkson, D. Eppstein, G. Miller, C. Sturtivant, S.-H. Teng, Approximating center points with iterated Radon points, International Journal of Computational Geometry and Applications 6 (3) (1996) 357–377, invited submission.
[3] R. Cole, M. Sharir, C.K. Yap, On k-hulls and related problems, SIAM Journal on Computing 16 (1) (1987) 61–77.
[4] L. Danzer, B. Grünbaum, V. Klee, Helly's theorem and its relatives, in: Proceedings of Symposia in Pure Mathematics, vol. VII, 1963.
[5] K. Fukuda, V. Rosta, Data depth and maximum feasible subsystems, in: D. Avis, A. Hertz, O. Marcotte (Eds.), Graph Theory and Combinatorial Optimization, Springer, 2005.
[6] J. Gil, W.L. Steiger, A. Wigderson, Geometric medians, Discrete Mathematics 108 (1–3) (1992) 37–51.
[7] J. Gilbert, G. Miller, S. Teng, Geometric mesh partitioning: Implementation and experiments, SIAM Journal on Scientific Computing 19 (6) (1998) 2091–2110.
[8] E. Helly, Über Mengen konvexer Körper mit gemeinschaftlichen Punkten, Jber. Deutsch. Math. 32 (1923) 175–176.
[9] S. Jadhav, A. Mukhopadhyay, Computing a centerpoint of a finite planar set of points in linear time, Discrete & Computational Geometry 12 (3) (1994) 291–312.
[10] G. Kalai, Combinatorics with a geometric flavor: Some examples, Geometric and Functional Analysis (2000).
[11] J. Matoušek, Approximations and optimal geometric divide-and-conquer, in: 23rd ACM Symposium on Theory of Computing, 1991, pp. 512–522.
[12] J. Matoušek, Lectures on Discrete Geometry, Springer-Verlag, 2002.
[13] N. Naor, M. Sharir, Computing a point in the center of a point set in three dimensions, in: CCCG: Canadian Conference in Computational Geometry, 1990, pp. 10–13.
[14] R. Rado, A theorem on general measure, Journal of London Mathematical Society 21 (1947) 291–300.
[15] J. Radon, Mengen Konvexer Körper, die Einen Gemeinschaftlichen Punkt Enthalten, Mathematische Annalen 83 (1921) 113–115.
[16] S.-H. Teng, Points, spheres, and separators: A unified geometric approach to graph partitioning, Ph.D. thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, August 1991. Available as Technical Report CMU-CS-91-184.
[17] H. Tverberg, A generalization of Radon's theorem, Journal of London Mathematical Society 41 (1966) 123–128.
[18] K. Verbarg, Approximate center points in dense point sets, Information Processing Letters 61 (5) (1997) 271–278.