Approximating Center Points with Iterated Radon Points*

K. L. Clarkson† David Eppstein‡ Gary L. Miller§ Carl Sturtivant¶
Shang-Hua Teng‖

Abstract

We give a practical and provably good Monte Carlo algorithm for approximating center points. Let P be a set of n points in $\mathbb{R}^d$. A point $c \in \mathbb{R}^d$ is a β-center point of P if every closed halfspace containing c contains at least βn points of P. Every point set has a $1/(d+1)$-center point; our algorithm finds an $\Omega(1/d^2)$-center point with high probability. Our algorithm has a small constant factor and is the first approximate center point algorithm whose complexity is subexponential in d. Moreover, it can be optimally parallelized to require $O(\log^2 d \log\log n)$ time. Our algorithm has been used in mesh partitioning methods and can be used in the construction of high breakdown estimators for multivariate datasets in statistics. It has the potential to improve results in practice for constructing weak ϵ-nets. We derive a variant of our algorithm whose time bound is fully polynomial in d and linear in n, and show how to combine our approach with previous techniques to compute high quality center points more quickly.

1 Introduction

A *center point* of a set P of n points in $\mathbb{R}^d$ is a point c of $\mathbb{R}^d$ such that every hyperplane passing through c partitions P into two subsets each of size at most $nd/(d+1)$ [9, 27]. This balanced separation property makes the center point useful for efficient divide-and-conquer algorithms in geometric computing [1, 18, 20, 27] and large-scale scientific computing [19, 23]. Recently, Donoho and Gasko [8] have suggested center points as estimators for multivariate datasets. They showed such points as estimators are robust and have a high "breakdown point." Note that we are not

*The preliminary version of this paper appeared in *the 9th ACM Symp. Computational Geometry*, (1993).

†AT&T Bell Laboratories, Murray Hill, NJ 07974.

‡Department of Information and Computer Science, University of California, Irvine, CA 92717. Supported by NSF grant CCR-9258355. Work performed in part while visiting Xerox Palo Alto Research Center.

§School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213. Supported in part by National Science Foundation grant CCR-91-96113.

¶Department of Computer Science, University of Minnesota, Minneapolis, MN 55455. Part of the work was done while the author was a visiting professor at Carnegie Mellon University.

‖Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA 02139. Supported in part by AFOSR F49620-92-J-0125 and Darpa N00014-92-J-1799. Part of the work was done while the author was at Xerox Palo Alto Research Center and Carnegie Mellon University. Current address: Department of Computer Science, University of Minnesota, Minneapolis, MN 55455.

referring here to the center of mass, or centroid. For brevity hereafter we will call a center point just a *center*.

The existence of a center of any point set follows from a classical theorem due to Helly [6]. However, finding an exact center seems to be a difficult task. It is possible to compute centers by solving a set of $\Theta(n^d)$ linear inequalities, using linear programming. The only improved results are that a center in two dimensions can be computed in $O(n \log^5 n)$ time, and in three dimensions in $O(n^2 \log^7 n)$ time [5]; the two-dimensional result has been very recently improved to linear time [14].

For most applications, it suffices to have an *approximate* center, a point c such that every closed halfspace containing c contains at least $n(\frac{1}{d+1} - \epsilon)$ points of P.

Such a center may be found with high probability by taking a random sample of P and computing an exact center of that sample. In order to achieve probability $1 - \delta$ of computing a correct approximate center, a sample of size $\Theta((1/\epsilon^2)(d \log O(1)/\epsilon + \log 1/\delta))$ is required ([VC71]; see also [13, 17, 23] and §6.). Hence the time to compute such an approximate center is $O((d^2/\epsilon^2)(d/\log d/\epsilon + \log 1/\delta))^d$ [3]. This bound is constant in that it does not depend on n, but it has a constant factor exponential in d. Alternatively, a deterministic linear-time sampling algorithm can be used in place of random sampling [15, 23], but one must again compute a center of the sample using linear programming in time exponential in d.

This exponential dependence on d is a problem even when d is as small as three or four. For example, the experimental results show that the sampling algorithm must choose a sample of about five hundred to eight hundred points in three dimensions. The sampling algorithm thus solves a system of $\binom{500}{3} \approx 20$ million linear inequalities. Many of our applications, e.g., in mesh partitioning [18], require an approximate center in four or more dimensions, for which the number of sample points and inequalities required is even larger. The aforementioned application (Donoho and Gasko [8]) of centers in multivariate statistical estimation also calls for efficient computation of centers in higher dimensions. The seemingly efficient sampling algorithm is too expensive for practical applications!

In this paper, we give a practical and provably good method for computing centers, from which we derive several center approximation algorithms. A version of this method was originally proposed as a heuristic to replace the linear programming based sampling algorithm by Miller and Teng [17] and has been implemented as a subroutine in their geometric mesh partitioning algorithm [11, 18]. The experimental results are encouraging. Our algorithm can also be used as part of a method for quickly computing weak ϵ-nets with respect to convex sets [1] and in other geometric applications of centers.

The simplest form of our algorithm runs in $O((d \log n + \log 1/\delta)^{\log d})$ time and uses randomization to find an $\Omega(1/d^2)$-center (see Section 2 for the definition) with probability $1 - \delta$. It does not use linear programming and has a small constant factor, making it suitable for practical applications. It can be efficiently parallelized in $O(\log^2 d \log \log n)$ time on distributed- and shared-memory parallel machines. We next describe a slightly more complicated form of the algorithm which takes time polynomial in both d and $\log 1/\delta$ and again computes $\Omega(1/d^2)$-centers. To the best of our knowledge, it is the first approximate center algorithm whose complexity is fully polynomial in both d and n. Finally, we show how to combine our algorithm with the linear programming sampling method to compute $(\frac{1}{d+1} - \epsilon)$-centers with probability $1 - \delta$, in time $(d/\epsilon)^{O(d)} \log 1/\delta$.

It is worthwhile to point out that our algorithms when specialized to one dimension yield a simple and fast algorithm for approximating the median, see Section 4.

Michelangelo Grigni has noted that a method of Valiant [24] on constructing short monotone formulae for the majority function is very close to our construction (Algorithm 2) of approximate median, and Valiant's analysis can also be adapted to give a proof of Theorem 4.3. Weide also proposed basically the same algorithm for low-storage on-line median approximation.[26]

The outline of this paper is as follows. Section 2 reviews some fundamental geometrical facts, and introduces the *Radon point* of a set of points. Section 3 gives our basic algorithm, based on iterated computation of Radon points. This algorithm is analyzed first in one dimension, in Section 4, then in general in Section 5. Section 6 discusses the use of random sampling to eliminate dependence on the number of input points. Section 7 gives our polynomial-time variant. Section 8 shows how to combine our approach with the linear programming algorithm. Section 9 discusses the precision needed for floating point arithmetic to succeed in computing centers using our algorithms.

2 Centers and Their Relatives

Let P be a finite set of points in $\mathbb{R}^d$. A hyperplane h in $\mathbb{R}^d$ divides P into three subsets: $P^+ = h^+ \cap P$, $P^- = h^- \cap P$, and $P \cap h$. The *splitting ratio* of h over P, denoted by $\phi_h(P)$, is defined as

$$\phi_h(P) = \max\left(\frac{|P^+|}{|P|}, \frac{|P^-|}{|P|}\right)$$

For each $0 < \beta \leq 1/2$, a point $c \in \mathbb{R}^d$ is a *β-center* of P if every hyperplane containing c $(1-\beta)$-splits P. A $(\frac{1}{d+1})$-center is simply called a *center*.

Proposition 2.1 ([6]) *Each point set $P \subset \mathbb{R}^d$ has a center.*

This fact, first observed by Danzer *et al.*[6], is a corollary of Helly's Theorem (to be given below). (See also [18, 23]; Edelsbrunner's text [9] gives proofs for the results in this section.)

Theorem 2.2 (Helly) *Suppose $\mathcal{K}$ is a family of at least $d+1$ convex sets in $\mathbb{R}^d$, and $\mathcal{K}$ is finite or each member of $\mathcal{K}$ is compact. Then if each $d+1$ members of $\mathcal{K}$ have a common point, there is a point common to all members of $\mathcal{K}$.*

The proof that centers exist is roughly as follows: consider a set of $d+1$ halfspaces each containing fewer than βn points of P, for $\beta \leq 1/(d+1)$. There must be points of P not in any of these halfspaces; hence any set of $d+1$ halfspaces containing more than $n(1-\beta)$ points of P must have a common point. It follows from Helly's Theorem that the family of halfspaces each containing more than $n(1-\beta)$ points of P has nonempty intersection. Any point in that intersection is a β-center.

This proof sketch implies that β-centers form a convex region, the intersection of a family of halfspaces; it is not too hard to show that this region is the intersection of a *finite* family of halfspaces.

Theorem 2.3 ([6]) *If $P \subset \mathbb{R}^d$ has n points, then its set of β-centers is the intersection of a family of closed halfspaces whose members each contains at least $n(1-\beta)$ points of P, and has at least d points of P in its bounding hyperplanes.*

The *linear programming algorithm*, based on this result, seeks to find a point in the common intersection of a family of no more than $\binom{n}{d} < n^d$ halfspaces. This problem is linear programming in d dimensions, with less than n^d inequality constraints; it can be solved in $O(n^d)$ time using a deterministic algorithm [16, 4] or a randomized one [3]; the latter algorithm gives a much smaller constant factor.

Another consequence of this theorem, discussed in Section 5, is that a candidate center needs only be verified with respect to the orderings on the points induced by the normals to n^d hyperplanes; if for any given such ordering, the center properties hold for a candidate with probability $1-p$, with $p < 1/n^d$, then the candidate is a center with non-zero probability.

Helly's Theorem can be proven using another result important for this paper, Radon's Theorem.

Theorem 2.4 (Radon) *If $P \subset \mathbb{R}^d$ with $|P| \geq d+2$, then there is a partition (P_1, P_2) of P such that the convex hull of P_1 has a point common with the convex hull of P_2.*

Proof: Suppose $P = \{p_1, \ldots, p_n\}$ with $n \geq d+2$. Consider the system of $d+1$ homogeneous linear equations

$$\sum_{i=1}^{n} \alpha_i = 0 = \sum_{i=1}^{n} \alpha_i p_i^j \qquad (1 \leq j \leq d),$$

where $p_i = (p_i^1, ..., p_i^d)$ in the usual coordinates of $\mathbb{R}^d$. Since $n \geq d+2$, the system has a nontrivial solution $(\alpha_1, \ldots, \alpha_n)$. Let U be the set of all i for which $\alpha_i \geq 0$, and V the set for all j for which $\alpha_j < 0$, and $c = \sum_{i \in U} \alpha_i > 0$. Then $\sum_{j \in V} \alpha_j = -c$ and $\sum_{i \in U} (\alpha_i/c) p_i = \sum_{j \in V} (\alpha_j/c) p_j$.

Let $P_1 = \{p_i \mid i \in U\}$, and $P_2 = \{p_i \mid i \in V\}$. Then the partition (P_1, P_2) of P has the desired property: the convex hull of P_1 has a point common with the convex hull of P_2. $\square$

Call the partition (P_1, P_2) of the theorem a *Radon partition*. We will call the point common to the hulls of P_1 and P_2 a *Radon point* of P. These points are the basis of our algorithm.

Definition 2.5 (Radon points) *Let P be a set of points in $\mathbb{R}^d$. A point $q \in \mathbb{R}^d$ is a* Radon point *[6] if P can be partitioned into 2 disjoint subsets P_1 and P_2 such that q is a common point of the convex hull of P_1 and the convex hull of P_2.*

Radon's Theorem implies that any set P of more than $d+1$ points has a Radon point. To compute a Radon point, we need only to compute a Radon point for any $d+2$ points of P. As in the proof above, this requires a non-zero solution of a linear system of $d+2$ variables and $d+1$ equations, and so takes $O(d^3)$ time.

Why are Radon points useful in computing centers? A Radon point of a set of $d+2$ points is a $2/(d+2)$-center of that set: any closed halfspace containing a Radon point r must contain a point of P_1 and a point of P_2. Hence the splitting ratio of a hyperplane containing r is at most $d/(d+2)$.

3 The Basic Algorithm

We now describe our algorithm for approximate centers. The algorithm iteratively reduces the point set by replacing groups of $(d+2)$ points by their Radon points. Such a reduction is guided by a complete $(d+2)$-way tree. We will show that the final point of this reduction process is an approximate center with high probability.

Figure 1: The Radon point q of four points in $\mathbb{R}^2$. When no point is in the convex hull of the other three (the left figure), then the Radon point is the unique cross of two linear segments. Otherwise (the right figure), the point that is in the convex hull of the other three is a Radon point.

Figure 2: The Radon point of five points in $\mathbb{R}^2$. Two cases are similar to these of two dimensions.

Algorithm 1 (Iterated Radon Points):
Input: a set of points $P \subset \mathbb{R}^d$

1. Construct a complete balanced $(d+2)$-way tree T of L leaves (for an integer L that is a power of $(d+2)$).

2. For each leaf of T, choose a point from P uniformly at random, independent of other leaves.

3. Evaluate tree T in a bottom-up fashion to assign a point in $\mathbb{R}^d$ to each internal node of T such that the point of each internal node is a Radon point of the points with its $(d+2)$ children.

4. Output the point associated with the root of T.

A complete $(d+2)$-way tree of L leaves has at most $L(1/(d+2)+1/(d+2)^2+...1/L) \leq L/(d+1)$ internal nodes. The above algorithm take $O(d^2L)$ time, with a small constant factor. Clearly, our algorithm can be implemented in $O(\log^2 d \log L)$ time using $O(d^2L/(\log^2 d \log L))$ processors in parallel. Our experimental results suggest that, independent of the size of original point set, $L = 800$ is sufficient for three dimensions and $L = 1000$ for four dimensions. We will discuss these experimental results at the end of Section 9.

4 Analysis: One Dimension

We wish to show that Algorithm 1 above finds a $(1/d^2)$-center with small probability of error. We first give a proof for one dimension and then extend it to higher dimensions.

In one dimension, the center of a point set is essentially the median. If the point set has an odd number of points, then its median is the only center. Otherwise, every point in the closed interval between the two medians is a center. Algorithm 1, when restricted to one dimension, gives the following algorithm for approximating the median of a linearly ordered set.

Algorithm 2: (Fast Approximate Median)
Input: a set of real numbers $P = \{p_1, \ldots, p_n\}$

1. Construct a complete balanced 3-way tree T of L leaves (for an integer L that is a power of three).

2. For each leaf of T, choose an element from P uniformly at random, independent of other leaves.

3. Evaluate tree T in a bottom-up fashion: at each internal node, keep the median of the numbers of its three children.

4. Output the number associated with the root of T.

The *rank* of a number p_i is the position that p_i would take in the sorted list of values in P. By induction, it can be shown that number associated with each node of the tree T belongs to P. The operation of internal nodes is comparison based, only the relative ranks (not the values) matter. Without loss of generality we assume that P is a permutation of the set $\{1/n, 2/n, ..., 1\}$.

We first note that the expected rank of the output of Algorithm 2 is $n/2$. This is because the output of Algorithm 2 is always from P, and because the operation in each internal node of the tree is symmetric with respect to the ranks. We now show that Algorithm 2 finds an approximate median with high probability.

Let $f_h(x)$ be the probability that the output of Algorithm 2, when using a tree T of height h, is no larger than x. Because the number of a leaf of the tree is chosen uniformly at random from the set P, $f_0(x) \leq x$. We now express $f_h(x)$ in terms of $f_{h-1}(x)$.

Let r be the root of T and c_1, c_2, and c_3 be its three children. Let $I(v)$ be the number chosen by the node v in T. We have that $I(r)$ is the median of $I(c_1)$, $I(c_2)$, and $I(c_3)$. Thus, $I(r) \leq x$ if and only if at least two of $I(c_1)$, $I(c_2)$, and $I(c_3)$ are less than x. Notice that each value $I(c_i)$ ($i \in \{1, 2, 3\}$) is chosen as the output of Algorithm 2 on a tree of height $h-1$, and that each value $I(c_i)$ is independent of the other two such values. Hence,

$$\begin{aligned} f_h(x) &= \binom{3}{2}(f_{h-1}(x))^2(1 - f_{h-1}(x)) + (f_{h-1}(x))^3 \\ &= 3(f_{h-1}(x))^2 - 2(f_{h-1}(x))^3 \\ &\leq 3(f_{h-1}(x))^2. \end{aligned}$$

By induction, we have

$$\begin{aligned} f_h(x) &\leq 3(f_{h-1}(x))^2 \\ &\leq 3 \cdot 3^2 \cdots 3^{2^{h-1}} (f_0(x))^{2^h} \\ &= \frac{1}{3}(3x)^{2^h}, \end{aligned}$$

and we are done.

A number q is a β*-median* of a set $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}$ if both $|\{p_i < q\}| \leq (1-\beta)n$ and $|\{p_i > q\}| \leq (1-\beta)n$.

Theorem 4.1 *For any $\beta < 1/3$, Algorithm 2 on a tree of height h, i.e., of sample size $L = 3^h$, outputs a β-median with probability of error at most $(3\beta)^{2^h}$.*

For example, when $\beta = 1/4$, we have the following corollary.

Corollary 4.2 *Algorithm 2 finds a 1/4-median in random $O((\log 1/\delta)^{\log_2 3})$ time with probability of error at most δ.*

A better analysis can be used to show that Algorithm 2 finds a $(1/2 - \epsilon)$-median with very high probability for all constant $0 < \epsilon < 1/2$. We use equality $f_h(x) = 3(f_{h-1}(x))^2 - 2(f_{h-1})^3$. Suppose

$x = 1/2 - \epsilon$. We have $f_0(x) = x = 1/2 - \epsilon$ and $f_1(1/2 - \epsilon) = 1/2 - 3\epsilon/2 + 2\epsilon^3$. If $\epsilon < 1/4$ then $2\epsilon^3 \leq \epsilon/8$. So, $f_1(1/2 - \epsilon) \leq 1/2 - 11\epsilon/8$. If $h_1 \geq \log_{11/8}(1/4\epsilon)$, then $f_{h_1}(1/2 - \epsilon) \leq 1/4$. The analysis above then shows that a total height of $h_1 + \log_2 \log 1/\delta + O(1)$ suffices to ensure a failure probability of at most δ.

Theorem 4.3 (Approximating Median) *Algorithm 2 finds a $(1/2 - \epsilon)$-median in random*

$$O(\log_{11/8}(1/4\epsilon) + (\log 1/\delta)^{\log_2 3})$$

time with probability of error at most δ.

5 Analysis: Higher Dimensions

Theorem 4.1 can be extended to higher dimensions. We start with some structural properties of β-centers.

Let l be a line in $\mathbb{R}^d$. The *projection* of a point $p \in \mathbb{R}^d$ onto l is a point $q \in l$ such that the line passing through p and q is perpendicular to l. By assigning a direction to l, we can introduce a linear ordering among points on l. For a point set $P = \{p_1, \ldots, p_n\}$, let $rank_l(p_i)$ be the rank of the projection of p_i among all projections of P. If two lines l and l' are parallel to each other and have the same direction (in vector sense), then for all $i : 1 \leq i \leq n$, $rank_l(p_i) = rank_{l'}(p_i)$.

Lemma 5.1 *Let $P = \{p_1, \ldots, p_n\}$ be a point set in $\mathbb{R}^d$. Then a point c is a β-center of P if and only if for all lines l, the projection of c onto l is a β-median of the projections of P onto l.*

Proof: Suppose c is a β-center of P. Let H be the hyperplane passing through c normal to l. Clearly, the projection c' of c onto l is the intersection of H and l. Notice that the projections of two points is on the same side of c' (on line l) if and only if they belongs to the same halfspace defined by H. Therefore c' is a β-median of the projections of P. The other direction of the lemma can be proved similarly. □

In order to check whether a point c is a β-center of P, we need only check the splitting ratio of the $O(n^{d-1})$ combinatorially distinct hyperplanes through c. Equivalently by Lemma 5.1, it is sufficient to check $O(n^{d-1})$ lines (normal to the set of hyperplanes above) to see whether the projection of c is a β-median of the projections of P. Whereas, if c is unknown, then Theorem 2.3 implies that $O(n^d)$ possible hyperplanes or normal lines need to be checked.

Corollary 5.2 *For each point set P in $\mathbb{R}^d$, there is a set of $O(n^d)$ lines such that a point c is a β-center of P if and only if for each line l from this line set, the projection of c is a β-median of the projections of P onto l.*

We now study the projection of Algorithm 1 onto a given line. Suppose we have $d + 2$ points $p_1, \ldots, p_{d+2}$. Let r be the Radon point of $p_1, \ldots, p_{d+2}$ and (P_1, P_2) be a corresponding Radon partition. For each hyperplane H passing through r, each (open) halfspace of H contains at most d points from $\{p_1, \ldots, p_{d+2}\}$, because r belongs to the convex hull of both P_1 and P_2. Let l be the line passing through the origin that is normal to H. From the discussion above, the projection of r is between (inclusively) the second smallest and the second largest projections of P onto l. In our analysis we will forget the higher-dimensional constraints on the problem and assume that any point between the second-smallest and second-largest projections could be chosen in a worst-case pattern of such choices. Therefore, with respect to a given line, Algorithm 1 can be emulated by the following process in one dimension.

Algorithm 3: (Projection of Algorithm 1)
Input: a set of real numbers $Q = \{q_1, \ldots, q_n\}$

1. Construct a complete balanced $(d+2)$-way tree T of L leaves (for an integer L).
2. For each leaf of T, choose an element from P uniformly at random, independent of other leaves.
3. Evaluate tree T in a bottom-up fashion: at each internal node, choose a number arbitrarily between the second smallest and the second largest numbers of its $d+2$ children.
4. Output the number associated with the root of T.

The following lemma is parallel with Theorem 4.1.

Lemma 5.3 *Let $\beta_d = 1/\binom{d+2}{2}$. For any $\beta < \beta_d$, Algorithm 3 above on a tree of height h, (i.e., $L = (d+2)^h$), outputs a β-median with probability of error at most $(\beta/\beta_d)^{2^h}$.*

Proof: Because we only concern the relative ranks of the input set Q, without loss of generality, we assume that Q is a permutation of the set $\{1/n, 2/n, ..., 1\}$.

Let $f_h(x)$ be the probability that the output of Algorithm 3, when using a tree T of height h, is no larger than x. Because the number of a leaf of the tree is chosen uniformly at random from the set Q, $f_0(x) \leq x$. We now express $f_h(x)$ in terms of $f_{h-1}(x)$.

The inputs to the root r of a tree of height h are from the outputs of $d+2$ trees of height $h-1$. Let $I(r)$ be the number chosen by the root. We have $I(r) \leq x$ only if at least two of its $d+2$ inputs are less than x. Therefore,

$$\begin{aligned} f_h(x) &= 1 - (1 - f_{h-1}(x))^{d+2} \\ &\quad - (d+2) f_{h-1}(x)(1 - f_{h-1}(x))^{d+1}. \end{aligned}$$

We can write the precise inclusion and exclusion form of the left hand side of the equation above. But the following upper bound (also known as Bonferroni inequalities [21]) is good enough for our analysis. There are in total $\binom{d+2}{2}$ different pairs from the $(d+2)$ distinct inputs. We call a pair (a, b) *good* if both $a \leq x$ and $b \leq x$. The probability that a pair is good is equal to $(f_{h-1}(x))^2$. Therefore, $f_h(x)$, the probability that there exists at least such a pair is bounded above by $\binom{d+2}{2}(f_{h-1}(x))^2 = (f_{h-1}(x))^2/\beta_d$.

We have

$$f_h(x) \leq \frac{1}{\beta_d}(f_{h-1}(x))^2$$

$$
\begin{aligned}
&\le \frac{1}{\beta_d} \cdot \frac{1}{\beta_d^2} \cdots \frac{1}{\beta_d^{2^{h-1}}} f_0(x)^{2^h} \\
&= \beta_d (x/\beta_d)^{2^h}.
\end{aligned}
$$

□

Therefore, for any $\beta < \beta_d$, with very high probability (the error probability is doubly exponentially small with respect to the height of the tree), the projection of the output of Algorithm onto a line is a β-median of the projections of the input point set P onto the line. By Corollary 5.2, the probability that the output of Algorithm 1 is not a β-center of P is at most $n^d (\beta/\beta_d)^{2^h}$.

The following theorem is the immediate result. We use as usual $\lg x \equiv \log_2 x$.

Theorem 5.4 *For $\beta < \beta_d$, if Algorithm 1 is run with*

$$h \ge \lg \left[(\lg n^d / \delta) / \lg(\beta_d / \beta) \right],$$

a β-center is found with probability of error at most δ. For $\beta < \beta_d/2$, this requires

$$L = (d+2)^h < (d \lg n + \lg(1/\delta))^{\log d + 2},$$

giving a time bound of

$$O(d^3) L/(d+1) = O(d^2 (d \lg n + \lg(1/\delta))^{\lg d + 2}).$$

6 Random Sampling

Both the linear programming algorithm and Algorithm 1 have running times dependent on n, the number of input points. It is possible to eliminate this dependence by computing the center of a random sample of P; applied to the linear programming algorithm, the resulting *sampling algorithm* is only Monte Carlo, yielding an approximate center only with high probability. However, the reduction in running time is quite substantial. Applied to Algorithm 1, the random sampling results allow us to reduce the running time, in exchange for a β-center with smaller β.

Call $S \subset P$ an *ϵ-sample* if for any halfspace H with $|H \cap P| \ge 4\epsilon n$,

$$\frac{|H \cap S|}{|S|} \le \frac{|H \cap P|}{n} + \epsilon.$$

(This is a one-sided version of ϵ-approximations [25], sufficient for our purposes.)

We will show that a random sample, chosen without replacement, is an ϵ-sample with high probability, for suitable ϵ and sample size. The sample is chosen by picking an element from the set at random and putting it in the sample, with all elements equally likely. Choosing "without replacement" means that the number of elements in the sample may be smaller than the number of such sampling trials.

Before proving the ϵ-sample property for random samples, we note the relation to β-centers.

Lemma 6.1 *A β-center of an ϵ-sample S of P is a $(\beta-\epsilon)$-center of P, for $\beta > 4\epsilon$.*

Proof: Suppose c is a β-center of S. Any halfspace containing c contains $\beta|S|$ points of S, and so by the definition of ϵ-sample, contains at least $(\beta-\epsilon)n$ points of P. Therefore c is a $(\beta-\epsilon)$-center of P. □

Lemma 6.2 *A random sample using r trials is an ϵ-sample with probability at least $1-e^{-\epsilon^2 r}\binom{n}{d}$, for $\epsilon < 1/18$.*

Proof: Consider a given halfspace H. Let $\beta = |H \cap P|/n + \epsilon$. The random variable $|H \cap S|$ is binomially distributed, with r trials and with probability of success $p = \beta - \epsilon$. It follows by standard tail estimates [2] that

$$\text{Prob}\{|H \cap S| - pr > \epsilon r\} < e^{-(\epsilon r)^2/2pr + (\epsilon r)^3/(pr)^2} = e^{-\epsilon^2 r(1/2p - \epsilon/p^2)}.$$

Since $\epsilon < \beta/4$ and $\epsilon < 1/18$, the probability that S fails the ϵ-sample condition for H is less than $e^{-\epsilon^2 r}$. The lemma follows, since there are at most $\binom{n}{d}$ distinct sets $\{S \cap H \mid H \text{ is a halfspace}\}$ [9]. □

This result implies that sample size $r = O(d \log n)/\epsilon^2$ is sufficient to give an ϵ-sample. This is not the smallest possible sample size for an ϵ-sample, however: as shown in [25], the $\log n$ term can be reduced to $\log(1/\epsilon)$; we give a proof here for ϵ-samples, and an absolute bound, not an asymptotic one.

Theorem 6.3 *A random sample using a number of trials at least*

$$\frac{3}{\epsilon^2}\left(d \log \frac{30}{\epsilon} + \log \frac{1}{\delta}\right)$$

is an ϵ-sample with probability at least $1-\delta$, for $\epsilon\delta^{1/d} \le 1/100$, $\epsilon < 1/18$, and $d \ge 2$.

Proof: We prove the lemma inductively on $n - r$, with the result trivially holding for $r = n$. Let ϕ_1 and ϕ_2 be constants to be chosen later, and for given ϵ and δ, take a random sample R_1 of P with number of trials

$$\frac{3\phi_1^2}{\epsilon^2}\left(d \log \frac{30\phi_1}{\epsilon} + \log \frac{2}{\delta}\right)$$

and then take a random sample R_2 of R_1 with number of trials

$$r_2 = \frac{\phi_2^2}{\epsilon^2} \log \binom{r_1}{d} \frac{e}{\delta}.$$

As above, we choose random samples with replacement, choosing each element of the sample from among those of the set with equal likelihood. Note that R_2 is also a random sample of P. By the previous lemma, R_2 is an ϵ/ϕ_2-sample of R_1 with probability $1-\delta/e$. By the inductive hypothesis, R_1 is an ϵ/ϕ_1-sample of P with probability $1-\delta/\phi_1$. It follows easily that R_2 is an $(\epsilon/\phi_1 + \epsilon/\phi_2)$-sample of P with probability at least $1-\delta/2-\delta/e > 1-\delta$. We choose $\phi_1 = 9$ and $\phi_2 = 9/8$, so that R_2 is an ϵ-sample with probability $1-\delta$.

It remains only to verify that the number of trials r_2 is as small as claimed. Using the standard inequality $\binom{a}{b} \le (ae/b)^b$, we have

$$\begin{aligned} r_2 &\le \frac{\phi_2}{\epsilon^2}\left(d\log\frac{r_1 e}{d} + \log\frac{e}{\delta}\right) \\ &\le \frac{\phi_2}{\epsilon^2}\left(d + d\log\left[\frac{3\phi_1^2}{\epsilon^2}\log\frac{30\phi_1}{\epsilon\delta^{1/d}}\right] + 1 + \log\frac{1}{\delta}\right) \\ &= \frac{d\phi_2}{\epsilon^2}\left(1 + 1/d + \log\frac{1}{\epsilon^2\delta^{1/d}} + \log 3\phi_1^2 + \log\log\frac{30\phi_1}{\epsilon\delta^{1/d}}\right). \end{aligned}$$

To have $r_2 \le \frac{3}{\epsilon^2}\left(d\log\frac{30}{\epsilon} + \log\frac{1}{\delta}\right)$, it suffices that

$$3 \ge \frac{\phi_2^2}{\log\frac{30}{\epsilon\delta^{1/d}}}\left(1 + 1/d + 2\log\frac{1}{\epsilon\delta^{1/d}} + \log 3\phi_1^2 + \log\log\frac{30\phi_1}{\epsilon\delta^{1/d}}\right).$$

This bound holds for $1/\epsilon\delta^{1/d} = 100$, and the expression on the right is decreasing in $z = \epsilon\delta^{1/d}$. Taking its derivative and evaluating the constant terms, the derivative is for $z > 0$ is a positive multiple of

$$4.66752972 + 1.61866664z - 11.19684392\ln(5.598421959 + z) - 2.0\ln(5.598421959 + z)z$$

up to a small relative error. This is negative for $z = 100$, and easily seen to be decreasing in z.

Hence R_2 is an ϵ-sample with probability $1-\delta$, of size r_2 that is within the bound of the theorem. This completes the inductive step, and the proof. □

We can now restate the analysis of Algorithm 1 to avoid any dependence on n in the time bound. For a given failure probability δ and $\epsilon = \beta_d/8$, view the random sample chosen for the leaves in Algorithm 1 as a sample of a random sample of size r, where r is chosen using the bounds of Theorem 6.3 to be an ϵ-sample with probability of failure $\delta/2$. The exact value of r can be obtained from the theorem, and $r = O(d^4(d\log d + \log(1/\delta))$. Now run Algorithm 1 with $h \ge \lg\lg(2r^d/\delta)$ to obtain a $(3\beta_d/8)$-center with probability at least $1-\delta$.

The following theorem states the bounds asymptotically.

Theorem 6.4 *For any δ, Algorithm 1 finds an $(3\beta_d/8)$-center in random $O((d^5\log d + d^4\log 1/\delta)^{\log_2 d})$ time, with probability of error at most δ.*

7 A Polynomial Algorithm

Algorithm 1 is subexponentially dependent on d, but not polynomial. Moreover, the dependence on δ involves an exponent of $\log d$, showing that increasing the reliability of the algorithm is costly in time. The problem is in the tree-like structure of the algorithm and in the branching factor of $d+2$ in that tree. If n is small (e.g., after the sampling modification of the previous section is applied) the number of leaves in the tree may end up being much larger than n itself. We now give the polynomial-time Algorithm 4 without this excess. The structure of Algorithm 4 is a layered DAG rather than a tree, with greater height, but with much smaller width.

Algorithm 4 applies the following scheme $z = \Theta(d + \log\log n)$ times to a set T, which is P initially: independently choose n random samples of T each using $d+2$ trials; replace T by the Radon points of these subsets. After this loop, choose any point of the final T as a center.

With sufficiently large n ($n = \Omega(d^4 \log^2 d)$), this algorithm returns a $\beta_d/3e$-center with probability $1 - 1/n$; it takes $O(n(d^4 + \log\log n))$ time. (Recall that $\beta_d \equiv 1/\binom{d+2}{2}$.)

Our analysis begins with a tail estimate for binomial distributions.

Lemma 7.1 ([2], A.12) *Let X be a binomial random variable with n trials and success probability p; Then for $u > 1$,*

$$\mathrm{Prob}\{X \geq upn\} \leq (e/u)^{upn} e^{pn}.$$

Proof: Omitted. □

Theorem 7.2 *After $z = \Theta(d + \log\log n + \log_n 1/\delta)$ iterations, Algorithm 4 returns an $O(1/d^2)$-center with probability at least $1-\delta$, for any $n = \Omega(d^{4+\epsilon} + d^{2+\epsilon}\log 1/\delta)$. Algorithm 4 requires $O(nd^3(d + \log\log n + \log_n 1/\delta))$ time.*

Proof: As in Theorem 5.4, it is sufficient to analyze a similar algorithm, where P is the set of values $\{1/n, 2/n, \ldots, 1\}$, and a sample with $d+2$ trials yields its second-smallest element for the new version of T. (In fact, for the algorithm here, T is a multiset: two samples may yield the same value, which will then have multiplicity greater than one.) We will find an upper bound for the probability that the final T has any members less than k/n, for $k = \beta_d n/3e$.

Let T_i denote T after iteration i, let $U_i \equiv T_i \cap \{1/n \ldots k/n\}$, and let $t_i \equiv |U_i|$. Here $T_0 = S$ and $t_0 = k$. We want to bound the probability that $t_z > 0$.

The key observation is that for a given value of t_{i-1}, the events that two random subsets yield numbers in $1/n \ldots k/n$ are independent; thus t_i given t_{i-1} is bounded by a binomial random variable with n independent trials, each with success probability bounded above by $p_{i-1} \equiv (t_{i-1}/n)^2/\beta_d$.

We can use Lemma 7.1 to bound the probability that $t_i > \gamma t_{i-1}$: let $\alpha \equiv 1/\beta_d n$, and put $u = \gamma n p_{i-1} = \gamma/\alpha t_{i-1}$. Then by Lemma 7.1,

$$\begin{aligned} \mathrm{Prob}\{t_i \geq \gamma t_{i-1}\} &\leq (e/u)^{up_{i-1}n} e^{p_{i-1}n} && (1)\\ &< (e\alpha t_{i-1}/\gamma)^{\gamma t_{i-1}}. && (2)\end{aligned}$$

Now put $\gamma = 2e\alpha t_{i-1}$, and suppose $\gamma t_{i-1} > \hat{d}$, where $\hat{d} \equiv 1 + (d+3)\lg n + \log 1/\delta$. Then

$$\mathrm{Prob}\{t_i \geq \gamma t_{i-1} = 2e\alpha t_{i-1}^2\} \leq (1/2)^{1+(d+3)\lg n + \lg 1/\delta} \leq \delta/n^{d+3},$$

Let z' be the smallest value such that $t_{z'} \leq \hat{d}$. Then inductively for $i \leq z'$,

$$t_i \leq (2e\alpha k)^{2^i - 1} k < \left(\frac{3}{2}\right)^{2^i - 1} n,$$

and so $z' < \lg[1 + \log_{3/2}(n/\hat{d})] = O(\log\log n)$. The probability that t_i fails to be reduced by the claimed amount is no more than $z'\delta/n^{d+3}$.

Thus t_i is quite rapidly reduced, in z' steps, to $\hat{d}$; moreover, it subsequently stays below $\hat{d}$ with the same high probability, by applying (1) with $\gamma = \hat{d}/t_{i-1}$, so that $t_i \le \hat{d}$ for $i \ge z'$ with failure probability δ/n^{d+3}.

It remains to show that t_i actually becomes zero with high probability, for i not too large. The probability that $t_i > 0$ is no more than the expected value of t_i, which is np_{i-1}, and so

$$\mathrm{Prob}\{t_{z'+q} > 0\} \le (\alpha\hat{d}^2)^q,$$

which is less than δ/n^{d+3} for $n/\log^2 n = \Omega(d^{4+\epsilon} + d^{2+\epsilon}\log 1/\delta)$ and $q = \Theta(d + \log_n 1/\delta)$. (Note that explicit upper bounds for q and z' can be obtained.) Hence after $z = O(d + \log_n 1/\delta)$ iterations, every point in T is a $\beta_d/3e$-median point for a given line with probability at least $1 - 2d\delta/n^{d+3}$. By Corollary 5.2, every such point is a $\beta_d/3e$-center with probability at least $1-\delta$.

The time bound for Algorithm 4 follows from the bound on the number of iterations, and the $O(d^3)$ work needed to find a Radon point. $\square$

Corollary 7.3 *Algorithm 4 together with random sampling can be used to compute an* $1/3e\binom{d+2}{2}$*-center with probability at least* $1-\delta$*, in time* $O(d^9 \log d + d^8 \log 1/\delta + d^{5+\epsilon}\log^2 1/\delta)$.

8 High Quality Centerpoints

Our algorithms are very efficient, but only produce $O(1/d^2)$-centers. The linear programming algorithm can produce better centers, but much more slowly; in particular not only is there a constant factor that depends exponentially on d, but there is also a nonconstant term of the form $O(\log 1/\delta)^d$. We now show how to combine our algorithm with linear programming to eliminate this term.

Suppose we wish to compute a $(\frac{1}{d+1} - \epsilon)$-center, for some $\epsilon < 1/(d+1)$. We take a collection of k random samples, each of size $O(d/\epsilon^2 \log 1/\epsilon)$. The linear programming algorithm gives us a center in each, which is a center of our original set with probability at least $1 - \exp(\Omega(-d^3 \log d))$. If we choose $k = \Theta(\log 1/\delta)$, we can show using Lemma 7.1 that with probability $1 - \delta/2$, all but $O(k/d^2)$ of the linear programming solutions are centers of the original set. We now find an $\Omega(1/d^2)$-center of these k LP solutions with probability $1-\delta/2$; this must then also be an approximate center of our original set.

Theorem 8.1 *In time* $O((d/\epsilon^2 \log d/\epsilon)^{d+O(1)} \log 1/\delta)$ *we can find a* $(\frac{1}{d+1} - \epsilon)$*-center, with probability* $1-\delta$.

9 Computation with floating point arithmetic

Up to now we have assumed that computation is done in exact arithmetic. This section has a few remarks on the precision needed for approximate arithmetic to succeed in computing β-centers with results nearly comparable to those for exact arithmetic.

We consider approximate arithmetic because it is much more commonly used than say, exact rational arithmetic, and because it is faster. Also, exact arithmetic seems to be very expensive for our algorithms: each iteration of Radon point calculation implies at least a constant factor increase in bit complexity; this yields a bit complexity at least exponential in d for our algorithms, either from the number of operations or the bit complexity of those operations.

Note that with approximate arithmetic, we cannot hope to obtain centers in the exact sense: suppose P is a set of points in the plane that lie on a line. If q is a point near the $(1/2)$-center of P, but off the line containing P, then q is only a 0-center: there is always a line between q and P, no matter how close q is to the $(1/2)$-center. Hence the idea of a center must be changed to allow such a point q to be considered useful output. With this is mind, rather than find a β-center, we will seek only a point that is within some small distance μ of a β-center; call such a point a *μ-weak β*-center.

We will assume hereafter that all coordinates of all points in P have absolute value less than one. This condition (and our bounds) then hold by appropriate scaling.

We relate the modified definition of a center to our analysis techniques by noting that q is μ-weak β-center of P if and only if, for every line, its projection onto that line has that property also for the projection of P onto that line. Our proofs consider lines normal to hyperplanes defined by points of P. Our exact results above rely on the fact that if a point is a β-center for those $O(n^d)$ lines, then it is a β-center for P. The analogous relation does not hold for μ-weak β-centers without some conditions on P or stronger conditions on the computed centers.

The rest of this section considers various such conditions; after lemmas regarding solution of linear systems, and the error properties of these algorithms as projected onto lines, we consider bounds that can be obtained by requiring smaller failure probabilities for the algorithms for a given projection line. This allows larger families of projection lines to be used, which allows bounds on μ. Finally, we consider bounds based on assumptions about the bit complexity of the input.

To analyze the error properties of the algorithm that finds Radon points, we will be a bit more specific about it. To restate some of the proof of Theorem 2.4, let P be a set of $d+2$ points, and let A be a $(d+1)\times(d+2)$ matrix whose columns are the points of P. Let $\mathbf{1}$ denote the row $(d+2)$-vector of 1's, that is, row vector with $d+2$ coordinates, each of which is 1. To find a Radon point of P, solve

$$\begin{bmatrix} A \\ \mathbf{1} \end{bmatrix} x = 0.$$

For a vector v, let $\oplus(v)$ denote a vector with coordinate $\oplus(v)_i = v_i$ if $v_i > 0$, and 0 otherwise. Let $x_+ = \oplus(x)$, where x solves the above equation, and let $x_- = \oplus(-x)$. Scale x so that $\mathbf{1}x_+ = 1$. In exact arithmetic $Ax_+ = Ax_-$ is a Radon point, and is a convex combination of the points P_+ corresponding to nonzero entries of x_+, as well as a convex combination of the analogous points of P_-. In approximate arithmetic, we have only that Ax_+ is close to the convex hull of P_+ (within machine precision), and may be close to the convex hull of P_- if x is good solution to the above linear system.

Let the unit roundoff of the arithmetic operations be $\mathbf{u}$; each arithmetic operation is done with a relative error bounded by $\mathbf{u}$.

We will assume that x is found using Gaussian elimination, and use the following error bounds.[12]

Lemma 9.1 *If the $m \times m$ linear system $By = c$ is solved using Gaussian elimination, then for computed vector $\hat{y}$ there is a matrix F such that $(B + F)y = c$, and $\|F\|_\infty < \iota_m \|B\|_\infty \mathbf{u} + O(\mathbf{u}^2)$, where $\iota_m = O(m^3 2^{m-1})$.*

(Recall that for a matrix A, $\|A\|_\infty \equiv \sup_x \|Ax\|_\infty / \|x\|_\infty$. This is the maximum of the ℓ_1-norms of the rows of A.)) With this lemma, we can begin to consider the error of one Radon point calculation.

Lemma 9.2 *Let P, A, x, and x_+ and x_- be as above. Let v be a unit row d-vector. (Here $\|v\|_2 = 1$.) Then $vAx_+ \geq vp - O(\mathbf{u})vp$ for all $p \in P_+$, and $vAx_+ \geq vp - O(\iota_d d\sqrt{d}\mathbf{u})$ for all $p \in P_-$.*

Proof: Using the previous lemma, the vectors x_+ and x_- satisfy

$$\left(\begin{bmatrix} A \\ \mathbf{1} \end{bmatrix} + \begin{bmatrix} E \\ e \end{bmatrix} \right) (x_+ - x_-) = 0,$$

where e is a row $(d+2)$-vector. Since $\|B\|_\infty$ for a matrix B bounds the L_1 norms of its rows, the previous lemma implies that $\|E\|_\infty$ and $\|e\|_1$ are both less than $\iota_d d\mathbf{u} + O(\mathbf{u})^2$. We have $\|x_+\|_1 = 1 + O(\mathbf{u})$, and from $(\mathbf{1} + e)x_+ = (\mathbf{1} + e)x_-$, it is not hard to obtain

$$\|x_-\|_1 = 1 + O(d\iota_d\mathbf{u}). \tag{3}$$

From

$$(A+E)x_+ = (A+E)x_-,$$

we obtain

$$Ax_+ - \frac{Ax_-}{\mathbf{1}x_-} = Ax_-(1 - \frac{1}{\mathbf{1}x_-}) + E(x_- - x_+).$$

We have $\|Ax_-\|_\infty \leq 1 + O(\iota_d d\mathbf{u})$, and

$$\|E(x_- - x_+)\|_\infty \leq \|E\|_\infty \|x_- - x_+\|_\infty = O(\iota_d d\mathbf{u}),$$

and from (3), $1 - \frac{1}{\mathbf{1}x_-} = O(\iota_d d\mathbf{u})$. Hence

$$\left\| Ax_+ - \frac{Ax_-}{\mathbf{1}x_-} \right\|_\infty = O(\iota_d d\mathbf{u}).$$

It follows that vAx_+ is within $O(\iota_d d\sqrt{d}\mathbf{u})$ of the projection of a point in the convex hull of P_-. □

Next we bound the error from computing Radon points iteratively, considering only projections. We refer to β-centers produced by an algorithm with β equal to the value proven for the corresponding exact algorithm. We will use a parameter h which is the depth of the Radon point calculation, bounded by the tree-depth h of Agorithm 1, or the number of iterations of Algorithm 4.

Lemma 9.3 *With P a set of n points in R^d and v a unit row d-vector, Algorithms 1 and 4 give a point q such that vq is a $O(\iota_d hd\sqrt{d}\mathbf{u})$-weak β-center for the values $vP = \{vp \mid p \in P\}$.*

Proof: Let P^* be a set of $d+2$ points. Suppose inductively that each value of vP^* is within η of a value in vP', where P' is the set of points corresponding to P^* but computed with exact operations. Let q be the computed Radon point of P^*. Then the previous lemma says that vq is within $O(\mathbf{u})$ of the convex hull of vP^*_+, and within $O(\iota_d d\sqrt{d}\mathbf{u})$ of the convex hull of vP^*_-. Since the lower endpoint of the convex hull of vP^*_+ is within η of the corresponding values of vP', and similarly for vP^*_-, the value vq is within $\eta + O(\iota_d d\sqrt{d}\mathbf{u})$ of the second largest value of vP'. Noting that a similar claim holds for $-v$, it follows that the output has projected value within $O(h\iota_d d\sqrt{d}\mathbf{u})$ of a β-center of vP. $\square$

We now have a $\mu = O(h\iota_d d\sqrt{d}\mathbf{u})$ result for the projections of P; does this imply the result in d dimensions, as for the exact case? Unfortunately, no; we need more conditions. The output point of the algorithm is within μ of every hyperplane supporting a facet of the center polytope. (Recall that such hyperplanes are defined by points of P.) The problem is that not all such output points are necessarily within μ of the center polytope; in two dimensions, this could occur when the center polygon has a vertex with a sharp angle. We next give three ways of handling this problem. One approach uses a collection of evenly distributed projection lines to force the computed point into an approximation of the β-center polytope. A different approach uses a collection of projection lines derived from the input points. Another approach relies on the input representation: when the points of P are given with bounded-precision rational coordinates, the angles or corners of the β-center polytope can't be too sharp.

Theorem 9.4 *If the failure probabilities of Algorithms 1 and 4 are no more than η^d, $\eta > 0$, for a given projection line, then these algorithms yield $O(\iota_d hd\sqrt{d}\mathbf{u}(1+\eta) + \sqrt{d}\eta)$-weak β-centers.*

Proof: It is not hard to show that there is a collection $\mathcal{L}$ of $O(1/\eta)^{d-1}$ lines through the origin that are evenly distributed, in the following sense: take any line ℓ through the origin; there is a line $\ell' \in \mathcal{L}$ such that the angle between ℓ and ℓ' is no more than η.

Suppose for a given η we have such a collection $\mathcal{L}$, and our algorithms have found a point q satisfying the bounds of the previous lemma, for all lines in $\mathcal{L}$. (Here v is a unit vector in line ℓ.) We show that q is close to a β-center. Let q' be the closest β-center to q, and let H be the hyperplane through q' normal to $q - q'$. We know that there is a line $\ell \in \mathcal{L}$ whose angle with $q - q'$ is no more than η. Let v be a row unit vector in $\mathcal{L}$; by the previous lemma, there is a β-center a such that $vq \le va + O(\iota_d d^2\sqrt{d}\mathbf{u})$. Let $w = (q - q')^T/\|q - q'\|_2$, a row vector. Since H separates the β-center polytope from q, we have $w(a - q') \le 0$. Using the Euclidean norm,

$$\begin{aligned}
\|q - q'\| &= w(q - q') \\
&= v(q - q') + (w - v)(q - q') \\
&= v(q - a) + v(a - q') + (w - v)(q - q') \\
&\le O(\iota_d d^2\sqrt{d}\mathbf{u}) + v(a - q') + (w - v)(q - q') \\
&= O(\iota_d hd\sqrt{d}\mathbf{u}) + w(a - q') + (v - w)(a - q') + (w - v)(q - q') \\
&\le O(\iota_d hd\sqrt{d}\mathbf{u}) + \|a - q'\|\|v - w\| + \|q - q'\|\|v - w\|.
\end{aligned}$$

Both a and q' are β-centers, and so have coordinates all less than 1 in absolute value. So $\|a - q'\| \le \sqrt{d}$, and since $\|v - w\| < \eta$, we have

$$\begin{aligned}
\|q - q'\| &\le \frac{O(\iota_d hd\sqrt{d}\mathbf{u}) + \|a - q'\|\|v - w\|}{1 - \|v - w\|} \\
&\le O(\iota_d hd\sqrt{d}\mathbf{u}(1+\eta) + \sqrt{d}\eta).
\end{aligned}$$

$\square$

This theorem suggests that we need to increase the running time of the algorithms by a factor of of about $d\log(1/\eta)$, and so achieve an accuracy near machine precision, the overhead is about $d\log(1/\mathbf{u})$. If such accuracy is desired and this factor is larger than d^2, the following theorem is of interest; in combination with the previous theorem, it suggests that we must multiply the number of operations by about $d\min\{d, log(1/\eta)\}$.

Theorem 9.5 *If the failure probabilities of Algorithms 1 and 4 are no more than $1/n^{d^2}$, for a given projection line, then these algorithms yield $O(\iota_d d^3 h\mathbf{u})$-weak β-centers with high probability.*

Proof: The analyses of this paper generally rely on projection lines perpendicular to hyperplanes through d input points. The theorem follows from considering a much larger class $\mathcal{L}(P)$ of projection lines, defined as follows: for each hyperplane H through d input points, include in $\mathcal{L}(P)$ the projection line normal to H. Also include projection lines found as follows: project the remaining input points orthogonally onto H, giving a point set P_H. Include in $\mathcal{L}(P)$ the lines of $\mathcal{L}(P_H)$, constructed recursively within H. (Here k points in a k-flat yield the line that is contained in that k-flat and perpendicular to the $(k-1)$-flat containing the points.) The number of projection lines in $\mathcal{L}(P)$ is no more than $\binom{n}{d}\binom{n}{d-1}\cdots\binom{n}{3} < n^{d^2}$.

Suppose now that Algorithms 1 or 4 are run so that an output point q is a $O(\iota_d d^3\sqrt{d}\mathbf{u})$-weak center with respect to every projection line of $\mathcal{L}(P)$. Suppose q is separated from the appropriate center polytope by a hyperplane H through d points of P. The distance $d(q, H)$ of q to H satisfies $d(q, H) = O(\iota_d d^3\sqrt{d}\mathbf{u})$. The squared distance of q to the center polytope is no more than $d(q, H)^2$ plus the squared distance of the projection of q onto H to the projection of the center polytope onto H. The result follows by induction on dimension; this induction yields a distance bound within $\sqrt{d}$ times the distance bound of the previous lemma. □

Finally, here is a bound that does not require a decrease in failure probability, with a corresponding increase in the number of operations. The bound does, however, depend on the bit complexity of the input.

Theorem 9.6 *Let P be a set of n points in R^d, where the coordinates of P are rational and the total bit complexity of P is σ. Algorithms 1 and 4 return a $\iota_d d^2 h\mathbf{u}^2 2^{(d^{O(1)}\sigma)}\mathbf{u}$-weak β-center of P with high probability.*

Proof: Suppose q is an output point of Algorithms 1 or 4, and q^* is the closest β-center to q. It is a standard result of optimization theory that the vector $c = q - q^*$ is a positive linear combination of outward unit normal vectors to d or fewer hyperplanes that determine the center polytope. These hyperplanes each separate q from the center polytope. By the lemma just above we know that q is close to each such hyperplane. (Here we assume nondegeneracy of the input, easily insured by adding $d+1$ affinely independent points.) That is, $c = Bz$, where z is a column vector, and the matrix B has columns that are unit vectors each normal to some hyperplane determining the center polytope. As remarked, by the lemma just above we have $\|c^T B\|_\infty = O(\iota_d dh\sqrt{d}\mathbf{u})$. We have

$$\begin{aligned} c^T c &= c^T(BB^{-1})((B^T)^{-1}B^T)c \\ &= c^T B(B^T B)^{-1}B^T c \\ &\le \|c^T B\|_2 \|(B^T B)^{-1}B^T c\|_2 \end{aligned}$$

$$
\begin{aligned}
&\le \|c^T B\|_\infty \|(B^T B)^{-1} B^T c\|_\infty \\
&\le \|c^T B\|_\infty \|(B^T B)^{-1}\|_\infty \|B^T c\|_\infty \\
&= O(\iota_d^2 d^4 h^2 \mathbf{u}^2) \|(B^T B)^{-1}\|_\infty .
\end{aligned}
$$

We can assume that the columns of B are linearly independent, and so $B^T B$ has an inverse. The columns of B are solutions of linear systems with coefficients from coordinates of P; hence by standard bounds applied twice, [22, §3.2]

$$\|(B^T B)^{-1}\|_\infty = 2^{d^{O(1)}\sigma}.$$

□

10 Final Remarks

In this paper, we have present provably good algorithms for centerpoint computation that has direct impact on practical applications. Gilbert and Teng [11] have implemented our center point algorithms as a subroutine of a geometric-mesh-partitioning toolbox. The toolbox is written in Matlab. It includes both edge and vertex separators, recursive bipartition, nested dissection ordering, visualizations and demos, and some sample meshes. The complete toolbox is available by anonymous ftp from machine `ftp.parc.xerox.com` as file `/pub/gilbert/meshpart.uu`.

Our experiment with a large class of two and three dimensional mesh data showed that Algorithm 1, using 800 sample points, finds 0.4-centers in three dimensions with 95 percent probability. Similar results are obtained for four dimensions (with 1000 sample points). We have also implemented Algorithm 4 of Section 7. Our experiments showed that 600 sample points and 6 levels of Radon reduction achieves comparable quality for four dimensions.

Matlab uses double precision. Our experiments indicated that our Radon-reduction algorithms are robust numerically. It is worthwhile to point out that for applications such as geometric mesh partitioning [18], it is sufficient to use weak-centers in the sense that the theory of partitioning holds if a large proportion of hyperplanes passing through the weak-center evenly splits the point set. Thus, our numerical analysis of Section 9 provides a theoretical justification for our experimental observation.

Acknowledgments We would like to thank Mic Grigni for pointing out that Valiant's method in [24] adapts to our approximate median algorithm, Dan Spielman for helpful discussions, and the anonymous referees.

References

[1] N. Alon, I. Bárány, Z. Füredi, and D. J. Kleitman. Point selections and weak ϵ-nets for convex hulls. Manuscript, 1991.

[2] N. Alon, J. Spencer, and P. Erdős. *The Probabilistic Method.* Wiley, New York, 1992.

[3] K. L. Clarkson. A Las Vegas algorithm for linear programming when the dimension is small. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 452–456, 1988. Revised

version: Las Vegas algorithms for linear and integer programming when the dimension is small (preprint).

[4] B. Chazelle and J. Matoušek. On linear-time deterministic algorithms for optimization problems in fixed dimension. *4th ACM Symp. Discrete Algorithms* (1993) 281–289.

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

[6] L. Danzer, J. Fonlupt, and V. Klee. Helly's theorem and its relatives. *Proceedings of Symposia in Pure Mathematics* 7, Amer. Math. Soc. (1963) 101–180.

[7] M. E. Dyer. On a multidimensional search procedure and its application to the Euclidean one-centre problem. *SIAM J. Comput.* 13 (1984) 31–45.

[8] D. L. Donoho and M. Gasko. Breakdown properties of location estimates based on halfspace depth and projected outlyingness. *The Annals of Statistics* 20 (4) (1992) 1803–1827.

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

[10] A. M. Frieze, G. L. Miller, and S.-H. Teng. Separator based divide and conquer in computational geometry. *4th ACM Symp. Parallel Algorithms and Architectures* (1992) 420–430.

[11] J. R. Gilbert, G. L. Miller, and S.-H. Teng. Geometric Mesh Partitioning: Implementation and Experiments. In *IPPS*, to appear 1995.

[12] P. E. Gill, W. Murray, and M. H. Wright. *Numerical Linear Algebra and Optimization.* Addison-Wesley, New York, 1991.

[13] D. Haussler and E. Welzl. Epsilon nets and simplex range queries. *Discrete Comput. Geom.* 2 (1987) 127–151.

[14] S. Jadhav and A. Mukhopadhay. Computing a center of a finite planar set of points in linear time. *9th ACM Symp. Computational Geometry* (1993) 83-90.

[15] J. Matoušek. Approximations and optimal geometric divide-and-conquer. *23rd ACM Symp. Theory of Computing* (1991) 512–522.

[16] N. Megiddo. Linear programming in linear time when the dimension is fixed. *SIAM J. Comput.* 12 (1983) 759–776.

[17] G. L. Miller and S.-H. Teng. Centers and point divisions. Manuscript, Carnegie Mellon University, 1990.

[18] G. L. Miller, S.-H. Teng, W. Thurston, and S. A. Vavasis. Automatic mesh partitioning. In A. George, J. Gilbert, and J. Liu, editors, *Sparse Matrix Computations: Graph Theory Issues and Algorithms*, IMA Volumes in Mathematics and its Applications. Springer-Verlag, 1993.

[19] G. L. Miller, S.-H. Teng, W. Thurston, and S. A. Vavasis. Finite element meshes and geometric separators. *SIAM J. Scientific Computing*, to appear, 1995.

[20] G. L. Miller, S.-H. Teng, W. Thurston, and S. A. Vavasis. Separators for sphere-packings and nearest neighborhood graphs. submitted to *J. ACM*, 1995.

[21] J. Spencer. *Ten Lectures on the Probabilistic Method*, SIAM, Philadelphia, 1987.

[22] A. Schrijver. *Theory of Linear and Integer Programming.* Wiley, New York, 1986.

[23] S.-H. Teng. *Points, Spheres, and Separators: a unified geometric approach to graph partitioning.* PhD thesis, Carnegie-Mellon University, School of Computer Science, 1991. Tech. Rep. CMU-CS-91-184.

[24] L. Valiant. Short monotone formulae for the majority function. *J. Algorithms*, 5, 1984, 363–366.

[25] V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl.*, 16 (1971) 264-280.

[26] B. W. Weide. *Statistical Methods in Algorithm Design.* PhD Thesis, Carnegie-Mellon University, School of Computer Science, 1978. Tech. Rep. CMU-CS-78-142.

[27] F. F. Yao. A 3-space partition and its application. *15th ACM Symp. Theory of Computing* (1983) 258–263.