Algorithms for Polytope Covering and Approximation, and for Approximate Closest-point Queries
draft *

Kenneth L. Clarkson
AT&T Bell Laboratories
Murray Hill, New Jersey 07974
e-mail: `clarkson@research.att.com`

Abstract

This paper gives an algorithm for *polytope covering*: let L and U be sets of points in R^d, comprising n points altogether. A *cover* for L from U is a set $C \subset U$ with L a subset of the convex hull of C. Suppose c is the size of a smallest such cover, if it exists. The randomized algorithm given here finds a cover of size no more than $c(5d \ln c)$, for c large enough. The algorithm requires $O(c^2 n^{1+\delta})$ expected time.[1] More exactly, the time bound is

$$O(cn^{1+\delta} + c(nc)^{1/(1+\gamma/(1+\delta))}),$$

where $\gamma \equiv 1/\lfloor d/2 \rfloor$. The previous best bounds were $cO(\log n)$ cover size in $O(n^d)$ time.[MS92b] A variant algorithm is applied to the problem of approximating the boundary of a polytope with the boundary of a simpler polytope. For an appropriate measure, an approximation with error ϵ requires $c = O(1/\epsilon)^{(d-1)/2}$ vertices, and the algorithm gives an approximation with $c(5d^3 \ln(1/\epsilon))$ vertices. The algorithms apply ideas previously used for small-dimensional linear programming. The final result here applies polytope approximation to the the *post office problem*: given n points (called sites) in d dimensions, build a data structure so that given a query point q, the closest site to q can be found quickly. The algorithm given here is given also a relative error bound ϵ, and depends on a ratio ρ, which is no more than the ratio of the distance between the farthest pair of sites to the distance between the closest pair of sites. The algorithm builds a data structure of size $O(n(\log \rho)/\epsilon^{d/2}$ in time $O(n^2(\log \rho))/\epsilon^d$. With this data structure, closest-point queries can be answered in $O(\log n)/\epsilon^{d/2}$ time.

*This manuscript merges [Cla93] and [Cla94]

[1] In this paper, δ will denote any fixed value greater than zero.

1 Introduction

Applications in graphics, design, and robotics lead to the problem of approximating a surface by a simpler one. The results here apply to the special case of approximating a surface that is the boundary of a convex body.

If P is a convex polytope (polyhedron) in R^d, then for $\epsilon > 0$, the points within ℓ_1 distance ϵ of the boundary of P lie between the boundaries of two nested convex polyhedra A and B with $A \subset P \subset B$. Thus the problem of finding a polytope P' with few facets whose boundary is close to that of P is reducible to that of separating two nested convex polyhedra by a polytope with few facets. In this way, an *approximation* problem leads naturally to a *separation* problem. Unfortunately, finding a minimal-facet separating polytope for arbitrary nested polyhedra is NP-hard[DJ90], and we hope only to find a polynomial-time algorithm that gives a separating polytope whose number of facets is within a small factor of the smallest possible number. Such an algorithm is given in §3.

It is not hard to show that if P' is any polytope separating A and B, then there is a coarsening A' of A that also separates A and B, such that A' has at most d times as many facets as P'. What is a "coarsening"? Recall that A can be described as the intersection of a family of halfspaces $\mathcal{H}(A)$: $A = \cap_{H \in \mathcal{H}(A)} H$. A coarsening of A is a polytope of the form $\cap_{H \in \mathcal{H}'} H$, where $\mathcal{H}' \subset \mathcal{H}(A)$. Thus in trying to find a simple separating polytope, we give away only a factor of d by restricting ourselves to coarsenings. This simple but crucial observation was made by Mitchell and Suri[MS92b].

Using projective duality, the problem of finding a coarsening with few facets is linear-time equivalent to finding a *covering* with few points: let L be a set of ℓ points in convex position, and let U be a set of u points, both in R^d. A *cover* for L from U is a set $C \subset U$ with L a subset of the convex hull of C. Seeking C with as few points as possible is equivalent to seeking a coarsening of A contained in B with as few facets as possible.

This paper gives an algorithm for covering such that the number of points in the returned cover is $O(d \log c)$ times the optimal number c. This implies an algorithm for separation that finds a polytope with $O(d^2 \log c)$ times as many facets as optimal. In the version of the approximation problem discussed above, the optimal cover size $c = O(d/\epsilon)^{d-1}$, independent of the combinatorial complexity of the polytope being approximated; hence the algorithm given below for covering can be applied to find an approximation within a factor of $5d^3 \ln(1/\epsilon)$ of optimal in number of vertices, as discussed in §3.

This dependence on $\ln c$ (and so $\ln(1/\epsilon)$) contrasts with previous results on this problem for $d > 2$, where the corresponding factor is $\log n$.[MS92b] Moreover, the algorithm given here is faster, sometimes much faster, than the previous $O(n^d)$; the new algorithm requires, in its simplest form,

$$O(c^2 \ell \log c + cu) \log(u/c)$$

expected time: linear in $n \equiv \ell + u$ for fixed c, and always $n^3 \log^{O(1)} n$. (A more complicated form of the algorithm has expected running time $cn^{1+\delta}n$ for $d = 3$.)

The algorithm is Las Vegas, and makes random choices; the expectation is with respect to the algorithm's behavior and independent of the input points.

The algorithm applies ideas used previously for linear programming[Cla88].

2 The covering algorithm

We will assume that the points of L and U are in general position: no $d+1$ are on the same hyperplane. For points p and q, say that p and q *see* each other, or are *visible* to each other, if the (relatively open) line segment $\overline{pq}$ does not meet P. Say that q and a facet F of a polytope P see each other if every $p \in F$ sees q. (Note that q sees F if and only if it sees at least one point of the relative interior of F.) A point q sees F if and only if q is in the halfspace bounded by the hyperplane through F and not containing P.

Let $\operatorname{conv} S$ denote the convex hull of the point set S.

Fact 2.1 *Let F be a facet of a polytope P. A point q sees F if and only if F is not a facet of* $\operatorname{conv} P \cup \{q\}$. *Moreover, q is contained in P if and only if q sees no facets of P.*

The algorithm needs answers to *visibility queries*: given a set S of n points in general position, and point p, the answer to such a query is a facet of $\operatorname{conv} S$ visible to p, or the answer that $p \in \operatorname{conv} S$ and no such facet exists. Visibility queries can be answered in $O(n)$ time using linear programming (e.g.,[Cla88]); with $m^{1+\delta}$ preprocessing, for any fixed $\delta > 0$, queries can be answered in $O(n(\log^{2d+1} n)/m^{\gamma})$ time, where $\gamma \equiv 1/\lfloor d/2 \rfloor$.[MS92a]

Let $C \subset U$ be some optimum cover of L, so $|C| = c$. The statement of the algorithm assumes that the optimal cover size c is known; this is no loss of generality, as discussed at the end of this section.

The development of the algorithm begins with Fact 2.1 above: for $R \subset U$, if L is not contained in $\operatorname{conv} R$, then there is a facet F of $\operatorname{conv} R$ that is visible to some point $p \in L$. Moreover, F must be visible to some point $q \in C$, since otherwise F is a facet of $\operatorname{conv} R \cup C$, which implies $p \notin \operatorname{conv} R \cup C \supset \operatorname{conv} C$. Let U_F be the set of points of U that see F. Then there is a point of C in the set U_F.

Lemma 2.2 below says that when R is a *random* subset of U, then U_F contains few points. Hence some information about C has been obtained: a small known set contains one of its members.

This may motivate the following algorithm outline, that closely follows an algorithm for linear programming[Cla88]: let each $p \in U$ have a weight w_p, with $w_p := 1$ for all $p \in U$ initially. Let $w(V)$ denote $\sum_{p \in V} w_p$ for $V \subset U$. Repeat the following: choose random $R \subset U$, by choosing each $p \in U$ independently to be in R with probability $1 - (1 - w_p/w(U))^r \leq r w_p/w(U)$, where $r = c4d \ln c$.

For each point $p \in L$ in turn, make visibility queries with respect to R; if there are no facets of $\operatorname{conv} R$ visible to a point $p \in L$, output R as a cover and quit. If there is such a facet F, and $w(U_F) \leq w(U)/2c$, then double the weights of the points of U_F, so $w_p \mathrel{*}= 2$ for $p \in U_F$. This completes the loop.

We turn to the analysis of this algorithm. Say that an iteration of the loop is *successful* if the weights were changed: there was a facet F visible to a point in L and with $|U_F| \leq w(U)/2c$. Call a facet of $\operatorname{conv} R$ an *L-facet* if it is seen by a point in L.

For a time bound, we need to know the chance of a successful iteration.

Lemma 2.2 *Given that an L-facet is found, the probability that an iteration will be successful is at least* $1/2$.

Proof. We show that with high probability, every facet of $\operatorname{conv} R$ is seen by few points of U; this follows easily from ancient results[HW87, Cla87], but is included for completeness. Suppose F is a potential facet of $\operatorname{conv} R$: an oriented simplex with d vertices in U, with at least j points $U_F \subset U$ on its positive side. The convex hull of R will have F as a facet if and only if its d vertices are in R, and points U_F that see F are not in R. The probability of this event for given F is

$$\begin{aligned}\prod_{p \in |F} (1-(1-\frac{w_p}{w(U)})^r) \prod_{p \in U_F} (1-\frac{w_p}{w(U)})^r &\leq \prod_{p \in |F} \frac{r w_p}{w(U)} e^{-rw(U_F)/w(U)} \\ &\leq \frac{r^d}{w(U)^d} e^{-rj/w(U)} \prod_{p \in |F} w_p.\end{aligned}$$

Let $\binom{U}{d}$ denote the set of subsets of U of size d. Since each $V \in \binom{U}{d}$ gives two F, it remains to bound

$$\sum_{V \in \binom{U}{d}} \prod_{p \in V} w_p,$$

subject to the conditions $\sum_{p \in U} w_p = w(U)$ and $|U| = n$. It's not hard to show that this expression is maximized when all $w_p = w(U)/n$, and so the probability that any facet of $\operatorname{conv} R$ is seen by more than j points of U is

$$\begin{aligned}\frac{r^d}{w(U)^d} e^{-rj/w(U)} \binom{n}{d} (w(U)/n)^d &\leq \frac{r^d}{n^d} \frac{(en)^d}{d^d} e^{-rj/w(U)} \\ &= \left(\frac{er}{d}\right)^d e^{-rj/w(U)},\end{aligned}$$

which is less than $1/2$ for $j = n/2c$, $r \geq c(4d \ln c)$, and c large enough. □

Lemma 2.3 *The expected number of iterations is at most* $1 + 8c \lg(u/c)$.

Proof. The proof follows arguments of Littlestone[Lit87] and Welzl[Wel88]. By Lemma 2.2, the number of "successful" iterations, in which the weights of U_F are changed, is on average at least half the total number of iterations. Consider the total weight $w(U)$. At each successful iteration, $w(U)$ increases by a factor of $1 + 1/2c < e^{1/2c} < 2^{3/4c}$. After I successful iterations, $w(U) \leq u2^{3I/4c}$. On the other hand, as noted above, U_F contained a member of C, and so that member is doubled in weight. Hence after I successful iterations, $w(C) \geq \sum_{p \in C} 2^{z_p}$, where $\sum_{p \in C} z_p = I$, and so by the convexity of the exponential function, $w(C) \geq c2^{I/c}$. Since $w(C) \leq w(U)$, the algorithm does at most $4c \lg(u/c)$ successful iterations, or $8c \lg(u/c)$ iterations on average. □

Theorem 2.4 *Let $\gamma \equiv 1/\lfloor d/2 \rfloor$, and let δ be any fixed value greater than zero. For known c, a cover of size no more than $c(4d \ln c)$ can be found in*

$$O(u^{1+\delta} + c\ell^{1+\delta} + c(\ell c)^{1/(1+\gamma/(1+\delta))} + (uc)^{1/(1+\gamma/(1+\delta))})$$

expected time, using sophisticated data structures. A simpler algorithm requires $O(\ell c \log c + u)c \log(u/c)$ expected time.

Proof. First consider the time needed for finding sets U_F and reweighting their points, over the whole algorithm. This requires answering $O(c \log(u/c))$ range queries on average, on a set of u points; with a simple algorithm, this requires $O(uc \log(u/c))$ time. Using sophisticated data structures, however, $O(u^{1+\delta} + (uc)^{1/(1+\gamma/(1+\delta))})$ expected time can be achieved, by trading off preprocessing time for query time.[Mat92]

It remains to bound the time required for answering visibility queries during each iteration; this requires answering no more than ℓ visibility queries on a set of points with expected size $r = O(c \log c)$. Using linear-time linear programming, this step requires $O(\ell c \log c)$ expected time per iteration. By again trading off preprocessing for query time, the queries can be answered in $O(\ell^{1+\delta} + (\ell c)^{1/(1+\gamma/(1+\delta))})$ time.[MS92a] This is multiplied by $O(c \log(u/c))$ for the bound.

(This has ignored any correlation between $|R|$ and $|U_F|$; it is quite likely that $|R| < 5r$, so if the algorithm is changed to make an iteration successful only if this holds, the change in $E|U_F|$ will be slight.) □

If the size c of an optimal cover is not known, the algorithm can postulate $c = (5/4)^i$ for $i = \lg d, \lg d + 1, \ldots$, and stop execution for a given value $(5/4)^i$ if the number of successful iterations exceeds the proven bound for covers of that size. In this way the cover returned is no more than $5/4$ as big as that for known c, and the work is dominated asymptotically by the work for the returned cover.

3 Approximation of polytopes

This section considers a polytope approximation problem: using the ℓ_1 distance measure, and given a set S of n points, find a polytope Q such that every point of $P \equiv \operatorname{conv} S$ is within ϵ distance of some point of Q, and every point of Q is within ϵ of some point of P. That is, suppose B is the set of points no farther than ϵ from the origin. We seek Q such that $Q \subseteq P + B$ and $P \subseteq Q + B$, with Q having as few vertices as possible. Here $P + B \equiv \{p + b \mid p \in P, b \in B\}$.

(An alternative approximation problem is simply to scale P by some $1 + \epsilon$, and solve the resulting separation problem. For very "flat" polytopes, this problem is plainly quite different from the one above, and arguably less useful.)

First, with small loss we need consider only a finite set of possible vertices for an approximating polytope. Let E be the set of $2d$ extreme points of B: these have coordinates all zero, except one that is either ϵ or $-\epsilon$.

Lemma 3.1 *If Q is an approximating polytope, there is another approximating polytope Q' that has vertices in $S+E$, and with at most d times as many vertices as Q.*

Proof. Since $Q \subset P+B = \operatorname{conv} S+E$, by a slight extension of Caratheodory's theorem, one can pick an arbitrary $v \in Q$ such that each vertex of Q is a convex combination of d points of $S + E$ together with v. For each vertex of Q, pick such points in $S + E$ and include them in a set C. So $Q' \equiv \operatorname{conv} C$ has no more than d times as many vertices as Q, and Q' is an approximating polytope since $P \subset Q + B \subset Q' + B$ and $Q' \equiv \operatorname{conv} C \subset \operatorname{conv} S + E = P + B$. □

We now have a covering problem: choose small $C \subseteq S + E$ such that $P \subset B + \operatorname{conv} C$. (Again, note that $\operatorname{conv} C \subset P + B$.)

To solve this problem, change the algorithm of the last section slightly: while the set U here is $S + E$, and the set L is S, we seek an "L-facet" that is a facet not of $\operatorname{conv} R$, but of $\operatorname{conv} R + E$, for $R \subset S + E$. The set U_F is computed not as the points of U that see F, but rather as the points $p \in U$ such that there is some $e \in E$ such that $p + e$ sees F.

With these changes, and with $r = c5d^3 \ln(1/\epsilon)$, the algorithm and its analysis are analogous to that for the covering problem; the only change in the analysis is the bound on potential L-facets: rather than $2\binom{r}{d}$, it is $2\binom{2dr}{d}$, since facets have vertices in $R + E$, not just R.

Just how large can c be? Dudley showed the following.[Dud74]

Lemma 3.2 *Let $P \subset R^d$ be compact and convex, and contained in a ball of radius 1. There is a convex polytope $P' \supset P$ with $O(1/\epsilon^{(d-1)/2}$ facets, and with P' within Haussdorf distance $1/\epsilon$ of P.*

The estimate $c = O(d/\epsilon)^{d-1}$ assumes that P is contained in an ℓ_∞ ball of radius one. Consider the regular grid of points with coordinates that are integral multiples of $2\epsilon/d$; every point of the boundary of P is within ℓ_∞ distance ϵ/d

of such a point, and so within ℓ_1 distance ϵ. Thus C can be taken to be the set of such grid points within ℓ_∞ distance ϵ/d. Letting $A(P)$ denote the surface area of P, the number of such grid points is $A(P)(d/2\epsilon)^{d-1}(1+O(\epsilon/d))$, and since P is contained in a cube of side length 2, $A(P) \leq d2^d$, yielding a bound of $d(d/\epsilon)^{d-1}$ for c.

4 Closest-point queries

A problem related to polytope approximation arises in an algorithm for the *post-office problem:* given a set S of n points (called sites) in d dimensions, build a data structure so that given a query point q, the closest site to q can be found quickly. The algorithm given here is given also a relative error bound ϵ, and depends on a ratio ρ, which is no more than the ratio of the distance between the farthest pair of sites to the distance between the closest pair of sites. The algorithm builds a data structure of size $O(n(\log\rho)/\delta^{d/2})$ in time $O(n^2(\log\rho))/\delta^d$. With this data structure, queries can be answered in $O(\log n)/\delta^{d/2}$ time, with high probability. The answer to the query is an ϵ-*closest* site: a site whose distance to the query point is within $1+\epsilon$ of closest site's distance. The algorithm uses randomization for the approximation problem, and also in a fashion similar to skip lists.[Pug90]

The approach here is based on that introduced by Arya and Mount.[AM93] The general idea is to find, for each site s, a list of sites N_s with the following property: if s is not the closest site to the query point q, then there is a site in N_s closer to q than s. With this property, a simple search procedure will lead to the closest site: pick any site s; if a site $t \in N_s$ is closer to q, assign t to s and repeat; otherwise return s as closest.

This approach is not so interesting just yet. The list N_s must be the set of Delaunay neighbors of s, as the interested reader can easily show. This makes for a space requirement of $\Omega(n^2)$ in the worst case, for $d > 2$. Also, the query time is $\Omega(n)$ in the worst case: there is no speedup over the obvious algorithm. (For uniformly distributed points, the query time is more like $O(n^{1/d})$, so this approach is not entirely useless, however.)

For more interesting results, make the problem easier: instead of the closest site, find the site whose distance to the query point is within $1+\epsilon$ of the distance of the closest site's, for some given $\epsilon > 0$. This is the approximate query problem solved by Arya and Mount. (More recently Arya *et al.* applied quadtree-like techniques to the problem.[AMN$^+$94]) Arya and Mount used a collection of narrow cones to obtain their lists, in a way similar to Yao's use of them for finding minimum spanning trees[Yao82]. Here the approach is to go from the desired conditions on the lists to a polytope approximation problem.

The modified construction begins as follows. For each site s, consider a list L_s with the following property: for any q, if there is $b \in S$ with

$$d(q,s) > (1+\epsilon)d(q,b),$$

then there is $b' \in L_s$ with

$$d(q,s) > (1+\epsilon')d(q,b'),$$

where $\epsilon' \equiv \epsilon/2$. Using such lists, the search procedure is as follows: start at any site s. If there is $t \in L_s$ with $d(q,s) > (1+\epsilon')d(q,t)$, then assign t to s and repeat. Otherwise, return s as the approximate closest. With L_s as defined, the site returned by this procedure will be within ϵ-closest. Moreover, the procedure makes progress at each step: the distance of the current site decreases by $1/(1+\epsilon')$ at each step.

Consider the condition satisfied by L_s in a contrapositive way. Fixing $s \in S$, let $\mathcal{N}_\epsilon(S)$ be defined by

$$\mathcal{N}_\epsilon(S) \equiv \{q \mid d(q,s) \leq (1+\epsilon)d(q,b) \text{ for all } b \in S\}.$$

This definition implies that

$$\mathcal{N}_{\epsilon'}(L_s) = \{q \mid d(q,s) \leq (1+\epsilon')d(q,b) \text{ for all } b \in L_s\}.$$

Thus the condition on L_s is equivalent to $\mathcal{N}_{\epsilon'}(L_s) \subset \mathcal{N}_\epsilon(S)$. The set $\mathcal{N}_\epsilon(S)$ is the Voronoi region of s in a multiplicatively weighted Voronoi diagram. The L_s we want is a small one that such that $\mathcal{N}_{\epsilon'}(L_s)$ is inside $\mathcal{N}_\epsilon(S)$; the problem of finding such an L_s can be solved by the techniques of the previous sections.

We'll look at this problem, and then at the application of the lists L_s to get a fast query time.

4.1 Finding L_s

The most direct approach to finding L_s via the ideas of §2 seem to require nonconvex optimization instead of linear programming subproblems. However, standard "lifting map" techniques yield convex programming subproblems, as we'll see next.

If we put s at the origin, the region $\mathcal{N}_\epsilon(S)$ is the intersection of all regions of the form $\{z \mid z^2 \leq (1+\epsilon)^2(z-b)^2\}$, where $b \in S$. The condition here is $z^2/(1+\epsilon)^2 \leq (z-b)^2$, or $\alpha z^2 \geq 2b \cdot z - b^2$, where $\alpha \equiv 1 - 1/(1+\epsilon)^2 \approx 2\epsilon$. Letting (z,y) denote a point in R^{d+1}, with $z \in R^d$ and $y \in R$, we have

$$\mathcal{N}_\epsilon(S) = \{z \mid (z,y) \in \mathcal{P}_\epsilon(S) \text{ and } y = z^2\},$$

where

$$\mathcal{P}_\epsilon(S) \equiv \bigcap_{b \in S} \mathcal{H}_{\epsilon,b}$$

with

$$\mathcal{H}_{\epsilon,b} \equiv \{(z,y) \mid \alpha y \geq 2b \cdot z - b^2\}.$$

Let $\Psi \equiv \{(z,y) \mid y \geq z^2\}$. Then for $\mathcal{N}_{\epsilon'}(L_s) \subset \mathcal{N}_\epsilon(S)$, it is enough that $\mathcal{P}_{\epsilon'}(L_s) \cap \Psi \subset \mathcal{P}_\epsilon(S) \cap \Psi$.

The problem of finding L_s satisfying this condition can be solved as in §2: apply the iterative reweighting scheme to the set of regions $\mathcal{H}_{\epsilon',b}$ with $b \in S$; at each step, check for each $b \in S$ that $\mathcal{P}_{\epsilon'}(R) \cap \Psi \subset \mathcal{H}_{\epsilon,b}$. This is a convex programming problem, and as shown by Adler and Shamir, it is solvable in $O(n)$ expected time using a randomized procedure similar to one for linear programming.[AS90] (The "base" case can be solved in polynomial time.[Vai89]) If all $b \in S$ satisfy this condition, then return R as L_s. Suppose the condition fails for some $\hat{b} \in S$. Then there will be some vertex of $\mathcal{P}_{\epsilon'}(R) \cap \Psi$ not in $\mathcal{H}_{\epsilon,\hat{b}}$. This vertex v is the analog of an L-facet. Find halfspaces $\mathcal{H}_{\epsilon',b}$ that do not contain v. If the number of such halfspaces is less than Cdn/r, then double the weights of the corresponding sites. (The constant C can be derived as for the covering algorithms above.)

4.2 The size of L_s

This algorithm returns a set L_s of size within $O(d \log c)$ of the best possible size c. How large can c be? In fact, $c = O(1/\epsilon)^{d/2} \log(\rho/\epsilon)$, as this subsection will show.

We'll apply Lemma 3.2 to bound the size of L_s. To do so, split Ψ into slabs $\Psi_i \equiv \{(z, y) \mid d_i \leq y \leq d_{i+1}\}$, for $i = 0 \ldots m$, where $d_0 \equiv (1/2) \min_{b \in S} \|b\|$, and $d_i = 2d_{i-1}$, and m is large enough that $d_m > (2/\alpha) \max_{b \in S} \|b\|$. We have $m = \log O(\rho/\alpha)$. We can assume that $d_0 \geq 1$, by the appropriate scaling. It is not hard to show that every halfspace $\mathcal{H}_{\epsilon,b}$ contains all points of Ψ not in some Ψ_i: only the points of Ψ_i need be considered.

Consider a given slab Ψ_i, and the two halfspaces $\mathcal{H}_{\epsilon,b}$ and $\mathcal{H}_{\epsilon',b}$. That is, we have the points (z, y) such that $\alpha y \geq 2b \cdot z - b^2$, and the points (z, y) such that $\alpha' y \geq 2b \cdot z - b^2$, where $\alpha' \equiv 1 - 1/(1 + \epsilon')^2$. Since $\alpha > \alpha'$ and $y \geq d_i$, we have

$$\alpha' y \leq \alpha y - d_i(\alpha - \alpha') \leq \alpha y,$$

and so $\mathcal{H}^*$ contains $\mathcal{H}_{\epsilon',b} \cap \Psi_i$, where $\mathcal{H}^*$ is a halfspace whose boundary hyperplane is a translation by $d_i(\alpha - \alpha')$ of the boundary of $\mathcal{H}_{\epsilon,b}$. It is not hard to show that Ψ_i is contained in a ball B' of radius no more than $d_i + 1$. Let B be a ball with the same center as B' and with radius $(d_i + 1)(1 + 2\epsilon)$. Now $\mathcal{P}_{\epsilon'}(S) \cap B' \subset \mathcal{P}_{\epsilon}(S) \cap B$, and moreover, every point of the former is farther than $d_i\epsilon/2$ from any point not in the latter, for $\epsilon < 1$. Thus we can apply Lemma 3.2 to find a polytope P_d with $O(1/\epsilon)^{d/2}$ facets such that $\mathcal{P}_{\epsilon'}(S) \cap B' \subset P_d \subset \mathcal{P}_{\epsilon}(S) \cap B$. It follows that $\mathcal{P}_{\epsilon'}(S) \cap B' \subset P_d \cap B' \subset \mathcal{P}_{\epsilon}(S) \cap B'$; just as with polytope covering, it follows that there is a coarsening $\mathcal{P}_{\epsilon'}(L_s^i)$ of $\mathcal{P}_{\epsilon'}(S)$ with at most $d + 1$ times as many facets as P_d such that $\mathcal{P}_{\epsilon'}(L_s^i) \cap B' \subset \mathcal{P}_{\epsilon}(S)$. Since $\Psi_i \subset B$, we have $\mathcal{P}_{\epsilon'}(L_s^i) \cap \Psi_i \subset \mathcal{P}_{\epsilon}(S)$

The list L_s is now obtained as the union of the lists L_s^i, with the size bound mentioned.

4.3 Solving closest-point problems

How can a fast query time be obtained using the lists L_s? Just as with Arya and Mount's work, a skip-list approach is helpful.[Pug90] Choose a family of subsets of S as follows: let $R_0 \equiv S$; to obtain R_{j+1} from R_j, pick each element of R_j to be in R_{j+1} with probability 1/2. Repeat this process until an empty R_k is obtained. If $s \in R_j$ but not R_{j+1}, say that s has level j. Construct the lists $L_{s,j}$ for each R_j, and so $s \in S$ has lists for each subset up to its level. To answer a query, start with some $s \in R_{k-1}$, and find the ϵ-closest site t_{k-1} in R_{k-1} using the lists $L_{s,k-1}$. Now find an ϵ-closest site t_{k-2} in R_{k-2}, starting with t_{k-1}. Repeat until t_0 is found, and return t_0 as an ϵ-closest site in S.

The correctness of this procedure should be clear. How much time does it take? Since each list is bounded in size by $Cdc \log c$, where $c = O(1/\epsilon)^{d/2}(\log \rho + \log(1/\epsilon) \log \epsilon)$, the query time is equal to $Cdc \log c$ times the number of sites visited in the procedure.

It is worthwhile to compare this procedure with one that finds the closest site in R_j at stage j, not just the ϵ-closest. Suppose we have t_j as the ϵ-closest at some stage, but indeed a site t is closest in R_j. When finding the ϵ-closest in R_{j+1}, the approximate procedure will in two steps find a site t' in R_{j+1} such that $d(q,t') \leq d(q,t)/(1+\epsilon')^2$. (Here we assume that the search in R_{j+1} takes at least two steps.) Since $(1+\epsilon')^2 \geq (1+\epsilon)$ for $\epsilon \geq 0$, we know that t' is closer to q than t. The number of sites visited at stage $j+1$ for the exact procedure is proportional to the number of sites of R_{j+1} closer to q than t; hence the number of sites visited for the approximate procedure in R_{j+1} is no more than 2 plus the number for the exact procedure.

To analyze the exact search procedure, we can follow Sen's analysis of skip lists.[**?**] Look at the search procedure "backwards": starting at the closest site to q in R_j, visit sites in order of increasing distance, until a site also in R_{j+1} is encountered. Call this a *level jump*. Once the level jump occurs, only sites in R_{j+1} are visited. The probability of a level jump at a given visited node is 1/2. Thus the probability that at least k level jumps occur in v node visits is the probability that a binomially distributed random variable has at least k successes in v trials. The query time can be greater than V only if either the number of level jumps exceeds K or if fewer than K level jumps occur in V attempts; the former probability is no more than $n/2^K$, which we'll need less than some probability P_1. This implies $K \geq \lg(n/P_1)$. The probability of fewer than K level jumps in V trials can be bounded using Chernoff bounds for the binomial; letting $\gamma \equiv 2K/V$, it is $\exp(-V(1-\gamma)^2/2)$. The probability that the query time exceeds $2\lg(n/P_1)/\gamma$ for a given point q is therefore $P_1 + \exp(-\lg(n/P_1)(1-\gamma)^2/\gamma)$. With $\gamma = 1/3$, this is less than $2P_1$. Hence a $O(Q)\log n$ query time is achievable with failure probability $1/n^Q$.

This analysis applies only to a single given point q; what about arbitrary points? As with similar situations in randomized geometric algorithms, a good query time holds for all points because there are $n^{d^{O(1)}}$ combinatorially distinct classes of points. That is, in an exact search algorithm, two points q_1 and q_2 will have the same sequence of visited sites, and so the same query time, if the distance order on the sites induced by the two points is the same. In other words, whether we sort the sites in order of distance from q_1, or sort them in order of distance from q_2, we get the same sorted order. How many classes of points are distinct in this way? Let $\mathcal{B}$ be the set of $\binom{n}{2}$ perpendicular bisector hyerplanes of pairs of sites, and let $\mathcal{A}(\mathcal{B})$ be the subdivision of R^d induced by those bisectors. Then all points in one cell (block) of $\mathcal{A}(\mathcal{B})$ induce the same distance orders, and so have the same query time. The number of cells of $\mathcal{A}(\mathcal{B})$ is $\binom{\binom{n}{2}}{d} < n^{2d}$. Thus a query time for any point of $O(\log n)$ occurs with probability $1 - 1/n^{\Omega(1)}$.

Queries can be made a bit faster by using splitting up the each list $L_{s,j}$ into lists $L^i_{s,j}$, where the superscript corresponds to the slabs Ψ_i in §4.2. When searching for a given site s at a given stage j, the list $L^i_{s,j}$ with $i = 2 \lg d(s,q)$ can be used. The gives a query time independent of ρ. It is also worth remarking that the while ρ was described as $\max_{b,b',b''\in S} \|b,b'\|/\|b,b''\|$, in fact the relevant quantity is the analogous expression with b' and b'' restricted to sites determining $\mathcal{P}_\epsilon(S)$.

5 Concluding remarks

Of course, the most interesting open question is whether the $\log c$ factor in the performance ratio can be reduced, as well as the d factors. The bound $4d \ln c$ can be easily sharpened to $2d \ln(Kc \ln c)$, for a small constant K.

In three dimensions, the bounds reduce to $O(cn^{1+\delta})$; this can readily be sharpened to $cn \log^{O(1)} n$.

Related ideas yield an output-sensitive algorithm for extreme points, requiring $O(an)$ time to find the a extreme points of a set of n points. This result will be reported elsewhere.

Acknowledgements. I'm grateful to Pankaj Agarwal, Michael Goodrich, and Subhash Suri for helpful comments. Of course, they aren't to blame.

References

[AM93] S. Arya and D. M. Mount. Approximate nearest neighbor queries in fixed dimensions. In *Proc. 4th ACM-SIAM Sympos. Discrete Algorithms*, pages 271–280, 1993.

[AMN+94] S. Arya, D. M. Mount, N. S. Netanyuhu, R. Silverman, and Angela Wu. An optimal algorithm for approximate nearest neighbor searching. In *Proc. 5th ACM-SIAM Sympos. Discrete Algorithms*, 1994.

[AS90] I. Adler and R. Shamir. A randomization scheme for speeding up algorithms for linear and convex quadratic programming problems with a high constraints-to-variables ratio. Technical Report 21-90, Rutgers Univ., May 1990. To appear in *Math. Programming*.

[Cla87] K. L. Clarkson. New applications of random sampling in computational geometry. *Discrete and Computational Geometry*, 2:195–222, 1987.

[Cla88] 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).

[Cla93] K. L. Clarkson. Algorithms for polytope covering and approximation. In *Proc. 3rd Workshop on Algorithms and Data Structures*, pages 246–252, 1993.

[Cla94] K. L. Clarkson. An algorithm for approximate closest-point queries. In *Proc. Tenth Annual Symposium on Computational Geometry*, pages 160–164, 1994.

[DJ90] G. Das and D. Joseph. The complexity of minimum nested polyhedra. In *Canadian Conference on Computational Geometry*, 1990.

[Dud74] R. M. Dudley. Metric entropy of some classes of sets with differentiable boundaries. *J. Approximation Theory*, 10:227–236, 1974.

[HW87] D. Haussler and E. Welzl. Epsilon-nets and simplex range queries. *Discrete and Computational Geometry*, 2:127–151, 1987.

[Lit87] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. In *Proc. 28th IEEE Symp. on Foundations of Computer Science*, pages 68–77, 1987.

[Mat92] J. Matoušek. Reporting points in halfspaces. *Computational Geometry: Theory and Applications*, pages 169–186, 1992.

[MS92a] J. Matoušek and O. Schwartzkopf. Linear optimization queries. In *Proc. Eighth ACM Symp. on Comp. Geometry*, pages 16–25, 1992.

[MS92b] J. Mitchell and S. Suri. Separation and approximation of polyhedral objects. In *Proc. 3rd ACM Symp. on Discrete Algorithms*, pages 296–306, 1992.

[Pug90] W. Pugh. Skip lists: a probabilistic alternative to balanced trees. *Commun. ACM*, 35:668–676, 1990.

[Vai89] P. M. Vaidya. A new algorithm for minimizing convex functions over convex sets. In *Proc. 30th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 338–343, 1989.

[Wel88] E. Welzl. Partition trees for triangle counting and other range searching problems. In *Proc. Fourth ACM Symp. on Comp. Geometry*, pages 23–33, 1988.

[Yao82] A. C. Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM J. Comput.*, 11:721–736, 1982.