Approximate Polytope Membership Queries

Sunil Arya[*]
Department of Computer Science and Engineering
The Hong Kong University of
Science and Technology
Clear Water Bay, Kowloon, Hong Kong
arya@cse.ust.hk

Guilherme D. da Fonseca[†]
Departamento de Informática Aplicada
Universidade Federal do Estado
do Rio de Janeiro (UNIRIO)
Rio de Janeiro, Brazil
fonseca@uniriotec.br

David M. Mount[‡]
Department of Computer Science and
Institute for Advanced Computer Studies
University of Maryland
College Park, Maryland 20742, USA
mount@cs.umd.edu

Appeared in STOC 2011, 579–586, 2011.

Abstract

We consider an approximate version of a fundamental geometric search problem, polytope membership queries. Given a convex polytope P in $\mathbb{R}^d$, presented as the intersection of halfspaces, the objective is to preprocess P so that, given a query point q, it is possible to determine efficiently whether q lies inside P subject to an error bound ε.

Previous solutions to this problem were based on straightforward applications of classic polytope approximation techniques by Dudley (1974) and Bentley *et al.* (1982). The former yields minimum storage, and the latter yields constant query time. A space-time tradeoff can be obtained by interpolating between the two. We present the first significant improvements to this tradeoff. For example, using the same storage as Dudley, we reduce the query time from $O(1/\varepsilon^{(d-1)/2})$ to $O(1/\varepsilon^{(d-1)/4})$. Our approach is based on a very simple algorithm. Both lower bounds and upper bounds on the performance of the algorithm are presented.

To establish the relevance of our results, we introduce a reduction from approximate nearest neighbor searching to approximate polytope membership queries. We show that our tradeoff provides significant improvements to the best known space-time tradeoffs for approximate nearest neighbor searching. Furthermore, this is achieved with constructions that are much simpler than existing methods.

1 Introduction

The field of computational geometry has witnessed the emergence of a number of results on geometric approximations. This has included efficient algorithms and data structures for approximate retrieval problems, such as approximate nearest neighbor searching and approximate range searching [3–5, 10, 14, 18, 21, 29]. It has also included algorithms for approximations

[*]Research supported by the Research Grants Council of Hong Kong, China under project number 610108.
[†]Research supported by FAPERJ grant E-26/110.091/2010 and CNPq/FAPERJ grant E-26/110.552/2010.
[‡]Research supported by NSF grant CCR-0635099 and ONR grant N00014-08-1-1015.

of statistical properties of geometric objects, as exemplified by work on coresets and clustering [1, 8, 11, 20, 22, 26]. Both areas of study have revealed a number of deep relationships between the accuracy of approximation, the combinatorial complexity of the approximating structures, and (for retrieval problems) the tradeoffs between the space and query times.

Convex polytopes are central to computational and combinatorial geometry. In this paper, we consider a fundamental search problem related to convex polytopes. Given a convex polytope P in $\mathbb{R}^d$, presented as the intersection of a set of halfspaces, the *polytope membership problem* involves preprocessing P so that it is possible to determine efficiently whether a given query point q lies within P. This problem is dually equivalent to answering halfspace emptiness queries in $\mathbb{R}^d$, and, as we shall see, our approximation algorithm for this problem will reveal new insights into approximate nearest neighbor searching.

Polytope membership queries find applications in many geometric areas, such as linear programming queries, ray shooting, nearest neighbor searching, and the computation of convex hulls [9, 12, 23, 25, 28]. In dimension $d \leq 3$, it is possible to build a data structure of linear size that can answer such queries in logarithmic time [16]. In higher dimensions, however, all exact data structures with roughly linear space take $\widetilde{O}(n^{1-1/\lfloor d/2 \rfloor})$ query time [24], which, except in small dimensions, is little better than brute-force search.

Throughout, we assume that P is a convex polytope lying within the hypercube $[-1, 1]^d$, which is represented as the intersection of the set of halfspaces that define its facets. Given $\varepsilon > 0$, we say that P' is an *ε-approximation* to P if the Hausdorff distance between P and P' is at most ε. (Typically, polytope approximation is defined relative to P's diameter, but by scaling P to lie within $[-1, 1]^d$, there is no loss of generality in this absolute formulation.) Dudley [17] showed that, for any convex body in $\mathbb{R}^d$, it is possible to construct an ε-approximating polytope with $O(1/\varepsilon^{(d-1)/2})$ facets. This bound is asymptotically tight in the worst case. Dudley's approximation has many applications in the construction of coresets [1].

Given a polytope P, a positive real ε, and a query point q, an *ε-approximate polytope membership query* determines whether q lies inside or outside of P, but it may return either answer if q's distance from P's boundary is at most ε. Approximate polytope membership queries arise in a number of applications, such as collision detection [19], training a support vector machine [7], and approximate nearest neighbor searching [14].

1.1 Polytope Membership Queries

Dudley's construction provides a naïve solution to the approximate polytope membership problem. Construct an ε-approximation P', and determine whether q lies within all its bounding halfspaces. This approach takes $O(1/\varepsilon^{(d-1)/2})$ query time and space. An alternative simple solution was proposed by Bentley *et al.* [6]. Create a d-dimensional grid with cells of diameter ε and, for every column along the x_d-axis, store the two extreme x_d values where the column intersects P. This algorithm produces an approximation P' with $O(1/\varepsilon^{d-1})$ facets. Given a query point q, it is easy to determine if $q \in P'$ in constant time (assuming a model of computation that supports the floor function), but the space required by the approach is $O(1/\varepsilon^{d-1})$.

These two extreme solutions raise the question of whether a tradeoff is possible between space and query time. Before presenting our results, it is illustrative to consider a very simple method for generating such a tradeoff. Given $r \in [\varepsilon, 1]$, subdivide the bounding hypercube into a regular grid of cells of diameter r and, for each cell that intersects the polytope's boundary, apply Dudley's approximation to this portion of the polytope. By a straightforward packing argument, the number of occupied cells is $O(1/r^{d-1})$. Since each cell is of diameter $O(r)$, each can be approximated to absolute error ε using $O((r/\varepsilon)^{(d-1)/2})$ facets per cell. Subject to minor technical details, the result is a data structure of space $O(1/(\varepsilon r)^{(d-1)/2})$ and query time

$O((r/\varepsilon)^{(d-1)/2})$. This interpolates nicely between the two extremes for $\varepsilon \leq r \leq 1$.

Given the optimality of Dudley's approximation, it may be tempting to think that the above tradeoff is optimal, but we will demonstrate that it is possible to do better. (To see why, observe that each piece of the polytope's boundary carries only a portion of the polytope's total curvature. Clarkson has shown that, when curvature is constrained, Dudley's bound is not tight [15].) Our main result is that it is possible to achieve significantly better space-time tradeoffs for approximate polytope membership. We show first that it is possible to build a data structure with storage $O(1/\varepsilon^{(d-1)/2})$ (the same as Dudley) that allows polytope membership queries to be answered significantly faster, in $O(1/\varepsilon^{(d-1)/4})$ time. In order to provide a tradeoff, we show that, by iterating a suitable generalization of this construction, it is possible to produce a succession of structures of increasingly better search times. The tradeoff is expressed (both in discrete and continuous forms) in the following theorem.

Theorem 1.1 *Given a polytope* P *in* $[-1,1]^d$ *and* $\varepsilon > 0$:

(i) *Given an integer constant* $k \geq 1$*, it is possible to answer* ε*-approximate polytope membership queries in time* $t = 1/\varepsilon^{(d-1)/2^k} \geq 1$ *from a data structure of space*

$$O\left(1/\varepsilon^{(d-1)\left(1-\frac{k}{2^k}\right)}\right).$$

(ii) *Given a real constant* $\alpha \geq 2$*, it is possible to answer* ε*-approximate polytope membership queries in time* $t = 1/\varepsilon^{(d-1)/\alpha} \geq 1$ *from a data structure of space*

$$O\left(1/\varepsilon^{(d-1)\left(1-\frac{1}{2^{\lfloor \log_2 \alpha \rfloor}}-\frac{\lfloor \log_2 \alpha \rfloor - 1}{\alpha}\right)}\right).$$

By way of contrast, observe that, if we express the tradeoff resulting from the above grid-based construction in this form, the space bound is $O(1/\varepsilon^{(d-1)(1-1/2^k)})$ for the same query time. These space-time tradeoffs are presented visually in Figure 1(a). (The lower bound shown in the figure will be discussed later.)

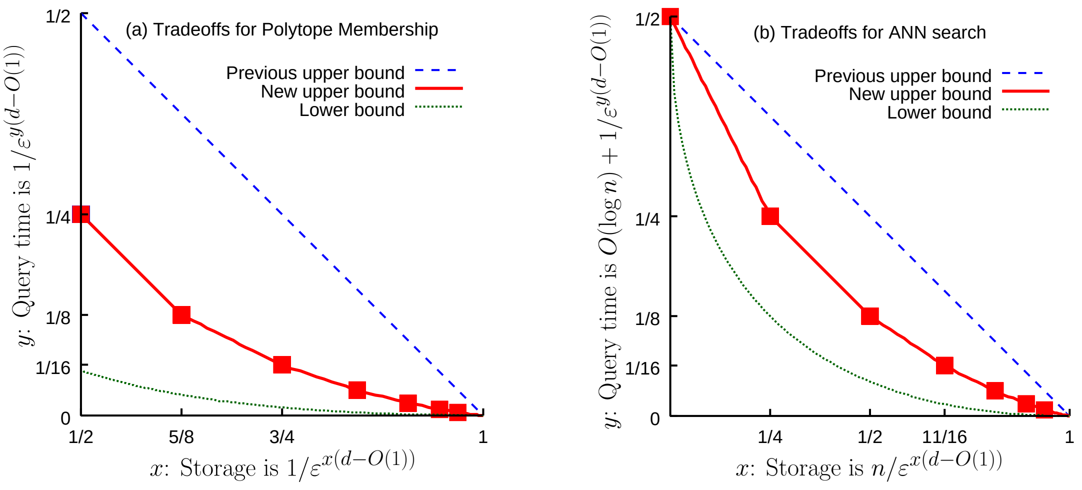

Figure 1: The multiplicative factor in the exponent of the $1/\varepsilon$ term for (a) polytope membership queries and (b) approximate nearest neighbor (ANN) queries. The $O(1)$ term in the exponent corresponds to a constant that does *not* depend on d.

The data structure and its construction are both extremely simple. The data structure consists of a quadtree of height $O(\log(1/\varepsilon))$, where each leaf cell stores a set of halfspaces whose intersection approximates the polytope's boundary within this cell. A query is answered by performing a point location in the quadtree followed by a brute force inspection of the halfspaces in the node. The data structure is constructed by the following recursive algorithm, called SplitReduce. It is given the polytope P, the approximation parameter ε, and the desired query time t. The initial quadtree box is $Q = [-1, 1]^d$.

```
SplitReduce($Q$):
1. Let $P'$ be an $\varepsilon$-approximation of $Q \cap P$.
2. If the number of facets $|P'| \leq t$, then $Q$ stores the hyperplanes bounding $P'$.
3. Otherwise, split $Q$ into $2^d$ quadtree boxes and invoke SplitReduce on each such box.
```

Although the algorithm itself is deceptively simple, the analysis of the storage as a function of the query time t is nontrivial. The algorithm is correct provided that, in the first step, the polytope P' is any ε-approximation, but the storage efficiency depends on the assumption that the number of facets of P' is approximately minimal. (This will be made precise in Section 2.) In addition to proving upper bounds on the space complexity of the above algorithm, we will establish a lower bound by demonstrating that, for $\alpha \geq 2$ and for query time $t = 1/\varepsilon^{(d-1)/\alpha}$, there exists a polytope for which this algorithm produces a data structure of space $\Omega\big(1/\varepsilon^{(d-1)(1-2\sqrt{2/\alpha}+3/\alpha)-1}\big)$ (see Figure 1(a) and Theorem 3.1).

1.2 Approximate Nearest Neighbor Queries

As further evidence of the value of this new perspective, we present a useful consequence for approximate nearest neighbor searching. In the *approximate nearest neighbor (ANN) problem* we are given a set S of n points in d-dimensional space, and we are to preprocess S in order to answer the following queries efficiently. Given a point q, find a point $p \in S$ whose Euclidean distance does not exceed $1+\varepsilon$ times the distance from q to its nearest neighbor. Data structures for ANN searching have been proposed by several authors [10, 14, 18, 21, 29].

The best space-time tradeoffs [4] for approximate nearest neighbor searching have query time roughly $O(1/\varepsilon^{d/\alpha})$ with storage roughly $O(n/\varepsilon^{d(1-2/\alpha)})$, for $\alpha \geq 2$. To better understand the tradeoff, note that the product of the ε terms in the storage and the *square* of the query time is $O(1/\varepsilon^d)$. These results are based on a data structure called an approximate Voronoi diagram (AVD). Assuming an AVD-based approach, lower bounds were also proved in [4]. The upper and lower bounds nearly match in both extremes: linear storage and logarithmic query time. By violating the AVD model, small improvements were obtained in [3]. Still, all previous data structures with query time $\widetilde{O}(1/\varepsilon^{(d-O(1))/\alpha})$ have storage $\Omega(n/\varepsilon^{(d-O(1))(1-2/\alpha)})$, where the $O(1)$ term in the exponent does *not* depend on d.

Given the maturity of this problem, one might expect that a significant increase in the $2/\alpha$ term in the exponent would involve fairly complex methods. We show that this is not the case by showing that better better tradeoffs can be achieved for approximate nearest neighbor searching through a simple reduction to the approximate polytope membership problem.

For example, the existing AVD-based tradeoff provides a query time of roughly $O(1/\varepsilon^{d/4})$ with space $O(n/\varepsilon^{d/2})$. By applying our polytope membership data structure, we can reduce the space bound to roughly $O(n/\varepsilon^{d/4})$ for the same query time. More generally, our reduction implies the following improved space-time tradeoffs for ANN queries (expressed both in discrete and continuous forms).

Theorem 1.2 *Given a set of n points in $\mathbb{R}^d$ and $\varepsilon > 0$:*

(*i*) *Let $k \geq 1$ be an integer constant. There is a data structure for approximate nearest neighbor searching with query time $O(\log n + (\log(1/\varepsilon))/\varepsilon^{d/2^k})$ and storage*

$$O\left(n/\varepsilon^{d\left(1-\frac{k+1}{2^k}\right)}\right).$$

(*ii*) *Let $\alpha \geq 2$ be a constant. There is a data structure for approximate nearest neighbor searching with query time $O(\log n + (\log(1/\varepsilon))/\varepsilon^{d/\alpha})$ and storage*

$$O\left(n/\varepsilon^{d\left(1-\frac{1}{2^{\lfloor \log_2 \alpha \rfloor}}-\frac{\lfloor \log_2 \alpha \rfloor}{\alpha}\right)}\right).$$

These space-time tradeoffs together with known upper bounds and lower bounds [4] are illustrated in Figure 1(b). Although the connection between the polytope membership problem and ANN has been noted before by Clarkson [14], we are the first to provide a reduction that holds for point sets with unbounded aspect ratio.

The remainder of the paper is organized as follows. In Section 2, we present our analysis for the SplitReduce algorithm. The lower bound analysis is presented in Section 3. The application to approximate nearest neighbor searching is discussed in Section 4. Concluding remarks follow in Section 5.

2 Upper Bound for Polytope Membership

In this section, we present upper bounds for the storage of the data structure obtained by the SplitReduce algorithm for a given query time t. We start by discussing the first step of the algorithm.

Step 1 consists of obtaining a polytope P' that ε-approximates $Q \cap P$. Some polytopes can be approximated with far fewer than the $\Theta(1/\varepsilon^{(d-1)/2})$ facets generated by Dudley's construction. For example, a simplex can be represented exactly with $d+1$ facets. The problem of generating a minimum-facet ε-approximation to a polytope can be reduced to a set cover problem [27]. By applying the well known greedy set-cover heuristic, it is possible to produce such an approximation in which the number of facets exceeds the optimum by a factor of $O(\log(1/\varepsilon))$. In particular, the set cover instance consists of one set for each facet f of the polytope, and the corresponding set contains all facets that are approximated by f's supporting hyperplane. Clarkson [13] presented a somewhat more complicated algorithm that does better. He showed that if c is the smallest number of facets required to approximate P, then it is possible to obtain an approximation with $O(c \log c)$ facets in $O(nc^2 \log n \log c)$ randomized time, where n is the number of facets of P.

For simplicity, we assume that the algorithm used in Step 1 produces an approximation to $Q \cap P$ with a minimum number of facets. In the full version it will be shown that an application of Dudley's algorithm to an appropriate subset of P suffices for our purposes. An alternative is to apply Chan's coreset construction [11] to P and then run Clarkson's approximation algorithm on the result. This yields a preprocessing time of $\widetilde{O}(n + 1/\varepsilon^{3(d-1)/2})$ but increases the query time by a negligible factor of $O(\log(1/\varepsilon))$.

For completeness, let us now describe Dudley's algorithm. Given a bounded polytope P, let the *size* of P, denoted σ_P, be the side length of the smallest axis aligned box Q that contains P. Through an appropriate translation we may assume that the center of Q is the origin. Dudley's algorithm obtains an approximation P' as follows. Let B be a ball of radius $\sigma_P\sqrt{d}$ centered at the origin. (Note that $P \subset B$.) Place a set W of $\Theta((\sigma_P/\varepsilon)^{(d-1)/2})$ points on the surface of

B such that every point on the surface of B is within distance $O(\sqrt{\varepsilon\sigma_P})$ of some point in W. For each point $w \in W$, let w' be its nearest point on the boundary of P. We call these points *samples.* For each sample point w', take the supporting halfspace passing through w' that is orthogonal to the vector from w' to w. P' is the intersection of these halfspaces.

A key element in the analysis of Dudley's construction is that, for each point p on the boundary of P, there is a sample point w' in the above construction (subject to a suitable adjustment of the constant factors) whose distance from p is at most $\sqrt{\varepsilon\sigma_P}$, and the distance from p to the supporting hyperplane at w' is at most ε. In summary, we may view Dudley's approximation as producing a set of $\Theta((\sigma_P/\varepsilon)^{(d-1)/2})$ halfspaces, where each halfspace is responsible for approximating a region of P's boundary of diameter at most $\sqrt{\varepsilon\sigma_P}$. We exploit the limited range of each Dudley halfspace in the following lemma.

Lemma 2.1 *Given a polytope P and $\varepsilon > 0$, Dudley's method produces a collection of $O((\sigma_P/\varepsilon)^{(d-1)/2})$ halfspaces whose intersection ε-approximates P. Furthermore, given a square grid of side length $O(\sqrt{\varepsilon\sigma_P})$, we may associate each halfspace with $O(1)$ grid cells, such that this halfspace is used for approximating the boundary of P within only these cells.*

Recall that our algorithm takes four inputs: polytope P, box Q, query time t, and approximation error ε. For simplicity, we refer to the algorithm as $\mathsf{SplitReduce}(Q)$, since the parameters P, t, ε remain unchanged throughout the recursive calls. The output of our algorithm is a quadtree whose leaf cells induce a subdivision of Q. Each leaf cell L stores a set of $t_L \leq t$ halfspaces whose intersection approximates $P \cap L$, where t_L is the minimum number of halfspaces required to approximate $P \cap L$. The storage of this quadtree is defined as the total number of stored halfspaces over all the leaf cells. Before establishing the space-time tradeoff, we show that the algorithm produces a data structure with query time $t = \Theta(1/\varepsilon^{(d-1)/4})$ and the same storage as Dudley's algorithm, $O(1/\varepsilon^{(d-1)/2})$.

Lemma 2.2 *The output of $\mathsf{SplitReduce}(Q)$ for*

$$t \geq \left(\frac{\sigma_Q}{\varepsilon}\right)^{(d-1)/4} \geq 1$$

is a quadtree with storage $O((\sigma_Q/\varepsilon)^{(d-1)/2})$.

Proof: Let T denote the quadtree produced by the algorithm. For each leaf cell L of T, let t_L be the number of halfspaces stored in L. We will show that $\sum_L t_L \leq (\sigma_Q/\varepsilon)^{(d-1)/2}$, which establishes the desired storage bound.

Towards this end, we first prove a lower bound on the size of any leaf cell L. We assert that there exists a constant c_1 such that every leaf cell L has size $\sigma_L \geq \sqrt{\varepsilon\sigma_Q/c_1}$. The assertion follows from Lemma 2.1. In particular, the standard Dudley technique applied to a cell of size $\sqrt{\varepsilon\sigma_Q/c_1}$ produces at most $c_D(\sigma_Q/c_1\varepsilon)^{(d-1)/4}$ halfspaces, where c_D is the constant arising from Dudley's method. By choosing c_1 to be a sufficiently large constant, the number of halfspaces is at most t, and the termination condition of our algorithm implies that such a cell is not further subdivided.

Let P_D be the polytope obtained by applying Dudley's algorithm to $P \cap Q$. Combining our assertion with Lemma 2.1 (where $P \cap Q$ plays the role of P in the lemma), each bounding halfspace of P_D is used in approximating $O(1)$ leaf cells. We assign each halfspace of P_D to these leaf cells. The correctness of Dudley's algorithm implies that the halfspaces assigned to any cell L provides an approximation of $P \cap L$. Thus, t_L is no more than the number of Dudley halfspaces assigned to L. Since the number of halfspaces of P_D is $O((\sigma_Q/\varepsilon)^{(d-1)/2})$, it follows that $\sum_L t_L = O((\sigma_Q/\varepsilon)^{(d-1)/2})$, which completes the proof. $\square$

We now use the previous lemma as a base case in order to extend the space-time tradeoff to other query times.

Theorem 2.3 *Let $\alpha \geq 2$ be a constant. The output of* SplitReduce(Q) *for* $t = (\sigma_Q/\varepsilon)^{(d-1)/\alpha} \geq 1$ *is a quadtree with storage*

$$O\left(\left(\frac{\sigma_Q}{\varepsilon}\right)^{(d-1)\left(1-\frac{1}{2^{\lfloor \log_2 \alpha \rfloor}}-\frac{\lfloor \log_2 \alpha \rfloor - 1}{\alpha}\right)}\right).$$

Proof: Let $k = \lfloor \log_2 \alpha \rfloor$. Our proof proceeds by induction on a constant number of steps k. The base case case $k = 1$ corresponds to Lemma 2.2. Next, assume that the theorem holds for $1, \ldots, k-1$, that is, for $\alpha < 2^k$. We need to prove that the theorem holds for $2^k \leq \alpha < 2^{k+1}$.

Let T denote the quadtree produced by the algorithm with $2^k \leq \alpha < 2^{k+1}$. Let T' denote the subtree induced by cells of size at least $\sqrt{\varepsilon\sigma_Q}/2$. Arguing exactly as in the proof of Lemma 2.2, the sum $\sum_L t_L$ over all leaf cells of T' is $O((\sigma_Q/\varepsilon)^{(d-1)/2})$. The leaf cells of T' that have $t_L \leq t$ are not refined by the algorithm and their total storage clearly satisfies the bounds of the theorem.

We now consider the subset $\mathcal{L}$ of leaf cells of T' such that for $L \in \mathcal{L}$ we have $t_L > t$. By Markov's inequality, $|\mathcal{L}| = O((\sigma_Q/\varepsilon)^{(d-1)/2})/t = O((\sigma_Q/\varepsilon)^{(d-1)(\frac{1}{2}-\frac{1}{\alpha})})$. Also, the size of the cells in $\mathcal{L}$ is between $\sqrt{\varepsilon\sigma_Q}/2$ and $\sqrt{\varepsilon\sigma_Q}$, since larger cells would have been subdivided.

Recall that the induction hypothesis states that the theorem holds for $\alpha < 2^k$. Since the size of the cells L in $\mathcal{L}$ is $\sigma_L \leq \sqrt{\varepsilon\sigma_Q}$, we have $t = (\sigma_L/\varepsilon)^{(d-1)/\alpha'}$, where $\alpha' \leq \alpha/2$ for sufficiently small $\varepsilon \leq \sigma_Q/4$ (if $\varepsilon > \sigma_Q/4$, then the theorem holds trivially). Therefore, we can use the induction hypothesis to obtain the following storage for each cell $L \in \mathcal{L}$:

$$\begin{aligned} & O\left(\left(\frac{\sigma_L}{\varepsilon}\right)^{(d-1)\left(1-\frac{1}{2^{k-1}}-\frac{k-2}{\alpha'}\right)}\right) \\ = \;& O\left(\left(\frac{\sigma_Q}{\varepsilon}\right)^{(d-1)\left(\frac{1}{2}-\frac{1}{2^k}-\frac{k-2}{\alpha}\right)}\right). \end{aligned}$$

We then multiply by $|\mathcal{L}| = O((\sigma_Q/\varepsilon)^{(d-1)(\frac{1}{2}-\frac{1}{\alpha})})$ completing the induction with total storage

$$O\left(\left(\frac{\sigma_Q}{\varepsilon}\right)^{(d-1)\left(1-\frac{1}{2^k}-\frac{k-1}{\alpha}\right)}\right).$$

□

Note that the height of the quadtree obtained by SplitReduce is $O(\log(1/\varepsilon))$, since cells of size ε are not subdivided. Therefore, it is straightforward to locate the quadtree cell containing the query point in $O(\log(1/\varepsilon))$ time. Theorem 1.1 follows as a straightforward consequence.

3 Lower Bound for Polytope Membership

In this section, we present lower bounds on the space-time tradeoffs obtained by our algorithm for polytope membership. Our main result is the following.

Theorem 3.1 *Let $\alpha \geq 2$ be a constant. There exists a polytope P such that the output of* SplitReduce$([-1,1]^d)$ *for* $t = 1/\varepsilon^{(d-1)/\alpha} \geq 1$ *is a quadtree with storage*

$$\Omega\left(1/\varepsilon^{(d-1)\left(1-2\sqrt{\frac{2}{\alpha}}+\frac{3}{\alpha}\right)-1}\right).$$

Our approach is similar to the lower bound proof of [4]. It is based on constructing a hypercylinder that takes dimension k, $1 \le k \le d-2$ as a parameter, and bounding the storage requirements of our data structure for it. We carry out this analysis in Lemma 3.3. Later, we will determine the value of k that yields the best lower bound on the storage as a function of t, ε, and d. The following preliminary lemma will be useful in Lemma 3.3.

Lemma 3.2 *In any dimension $d \ge 2$, there exists a polytope P_0 of diameter at most Δ such that any polytope P_0' that approximates P_0 requires $\Omega\left((\Delta/\varepsilon)^{\frac{d-1}{2}}\right)$ facets.*

Proof: Consider a d-dimensional ball B of radius $\Delta/2$. We first show that any polytope P_B that approximates B to within error 2ε must have at least $\Omega((\Delta/\varepsilon)^{(d-1)/2})$ facets. To see this, consider a hyperplane H passing through any facet of P_B. Translate H so it passes through a point x on the boundary of B. Clearly, x must be within distance 2ε of H. Applying Pythagoras' theorem, it follows that all points within distance 2ε of H are contained in a spherical cap that is centered at x and has angular radius $O(\sqrt{\varepsilon/\Delta})$. By a simple packing argument, it follows that P_B must have $\Omega((\Delta/\varepsilon)^{(d-1)/2})$ facets.

Next, consider any polytope P_0 that approximates B to within error ε. It follows from the triangle inequality property of Hausdorff distances that any polytope that approximates P_0 to within error ε approximates B to within error 2ε, and hence requires $\Omega((\Delta/\varepsilon)^{(d-1)/2})$ facets. $\square$

Lemma 3.3 *Let k and t be integers, where $1 \le k \le d-2$, $1 \le t \le 1/\varepsilon^{(d-1)/2}$, and let $0 < \varepsilon \le 1$. There exists a polytope P such that the output of SplitReduce$([-1,1]^d)$ for P is a quadtree with storage $\Omega\left(t\left(\frac{1}{\varepsilon t^{2/(d-k-1)}}\right)^k\right)$.*

Proof: By Lemma 3.2, there exists a d-dimensional polytope of unit diameter that requires at least $\Omega(1/\varepsilon^{(d-1)/2}) = \Omega(t)$ facets. It follows that $\Omega(t)$ is a lower bound on the space of the quadtree generated by our algorithm for this polytope.

We next tackle the more interesting case when the lower bound given by the lemma is better than $\Omega(t)$. Consider an infinite polyhedral hypercylinder C whose "axis" is a k-dimensional linear subspace K defined by k of the coordinate axes, and whose "cross-section" (i.e., intersection with any $(d-k)$-dimensional hyperplane orthogonal to K) is the polytope P_0 given in Lemma 3.2 for dimension $d-k$. It follows from Lemma 3.2 that we can choose $\Delta = O(\varepsilon t^{2/(d-k-1)})$ such that any polytope P_0' that approximates P_0 requires at least $(2^d+1)t$ facets. Define polytope P to be the truncated cylinder obtained by intersecting C with the hypercube $[-1,1]^d$.

We may assume that $\Delta \le 1$ since otherwise the bound given by the lemma is no better than $\Omega(t)$. We will show that the output of our algorithm for input P is a quadtree with storage as given in the lemma.

Consider a set $\mathbb{X}$ of points that are at least Δ apart placed on the intersection of $[-1,1]^d$ with the k-dimensional axis of the hypercylinder C. By a simple packing argument, we can ensure that the number of points in $\mathbb{X}$ is at least $\Omega(1/\Delta^k)$. Let $P_0^{\mathbb{X}}$ denote the set of cross-sections of C passing through the points of $\mathbb{X}$. Consider the set of leaf cells of the quadtree that overlap any cross-section $P_0 \in P_0^{\mathbb{X}}$. Recall from our construction that these cells must together contain at least $(2^d+1)t$ facets. We say that a leaf cell is *large* if its size is at least Δ and *small* if its size is less than Δ. By a simple packing argument, the number of large leaf cells intersecting P_0 is at most 2^d. Since each leaf cell contains at most t facets, the large leaf cells can together contain at most $2^d t$ facets. It follows that at least t facets are contained in the small leaf cells intersecting P_0. Noting that small leaf cells cannot intersect two cross-sections of $P_0^{\mathbb{X}}$, since they are at least Δ apart, it follows that the total space used by all the small leaf cells together is at least $\Omega(t|\mathbb{X}|) = \Omega(t/\Delta^k) = \Omega(t(1/(\varepsilon t^{2/(d-k-1)}))^k)$, which proves the lemma. $\square$

We next determine the value of k that yields the best lower bound in the above lemma, as a function of t, ε, and d. To put the lower bound in a more convenient form, set $t = 1/\varepsilon^\tau$, where $0 \leq \tau \leq (d-1)/2$. Up to constant factors, we obtain a lower bound on the storage of

$$\left(\frac{1}{\varepsilon}\right)^{k+\tau\left(\frac{d-3k-1}{d-k-1}\right)}.$$

To derive the best lower bound for fixed τ, we select k to maximize the exponent. Setting the derivative to zero, we obtain $k = (d-1) - \sqrt{2\tau(d-1)}$. (Although this value of k is not necessarily an integer, the resulting error is captured by the "-1" term in the exponent of Theorem 3.1.) Substituting this value of k, the exponent in the lower bound is

$$(d-1) - 2\sqrt{2\tau(d-1)} + 3\tau.$$

Finally, setting $\tau = (d-1)/\alpha$, where $\alpha \geq 2$ is a constant, this simplifies to $(d-1)(1-2\sqrt{2/\alpha}+3/\alpha)$, which proves Theorem 3.1.

4 Approximate Nearest Neighbor Searching

In this section, we present a novel reduction from approximate nearest neighbor searching to approximate polytope membership. Using our solution for the approximate polytope membership problem, we obtain significant improvements to the efficiency of approximate nearest neighbor searching. In order to determine the actual approximate nearest neighbor, instead of simply approximating its distance, we make the following assumption. When the query point is reported as outside the polytope, the approximate polytope membership data structure also returns a halfspace that defines a facet of the polytope and does not contain the query point. It is easy to modify our data structure to obtain such a witness.

The reduction is based on the approximate Voronoi diagram (AVD) construction from [4]. The AVD construction uses a BBD tree where each leaf cell of the BBD tree stores a set of *representative* points. In the AVD, a query is answered by locating the leaf cell that contains the query point and then determining the nearest representative from the corresponding cell, exactly or approximately. The cells of a BBD tree correspond to the set theoretic difference of two quadtree boxes.

The following lemma is central to our reduction and follows easily from the proofs of Lemmas 6.1 and 8.1 in [4]. Given a cell Q in a BBD tree, let σ_Q denote the side length of Q and let B_Q be the ball of radius $2\sqrt{d}\sigma_Q$ whose center coincides with the center of Q's outer box. The lemma is illustrated in Figure 2.

Lemma 4.1 *Let $0 < \varepsilon \leq 1/2$ be a real parameter and S be a set of n points in $\mathbb{R}^d$. It is possible to construct a BBD tree T with $O(n \log(1/\varepsilon))$ nodes, where each leaf cell Q stores a subset $R_Q \subset S$ satisfying the following properties:*

- *(i) R_Q is an ε-representative set for Q (that is, for any point q in Q, one of the points in R_Q is an approximate nearest neighbor of q).*
- *(ii) At most one point of R_Q is contained in the ball B_Q, and the remaining points of R_Q are contained in the annulus $cB_Q \setminus B_Q$ for some constant c.*
- *(iii) The total number of representatives over all the leaf cells is $O(n \log(1/\varepsilon))$.*

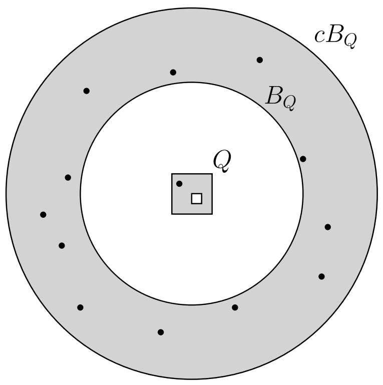

Figure 2: Illustration of Lemma 4.1.

Moreover, it is possible to compute the tree T and the sets R_Q for all the leaf cells in total time $O(n \log n \log(1/\varepsilon))$ and the cell that contains a query point can be located in $O(\log n + \log\log 1/\varepsilon)$ time.

The following lemma explains how to connect Lemma 4.1 with approximate polytope membership queries.

Lemma 4.2 *Let $0 < \varepsilon \leq 1/2$ be a real parameter and consider a quadtree box Q and a set of points R_Q as in Lemma 4.1. Given a data structure for approximate polytope membership in $(d+1)$-dimensional space with query time t_{d+1} and storage s_{d+1}, it is possible to preprocess R_Q into an ANN data structure for Q with $O(t_{d+1} \log(1/\varepsilon))$ query time and $O(s_{d+1})$ storage.*

Proof: Since at most one point of R_Q is contained in B_Q, the corresponding point may be inspected separately without increasing the complexity bounds. Therefore, we can safely assume for simplicity that all points in R_Q are contained in the annulus $cB_Q \setminus B_Q$ for some constant $c > 1$.

We make use of the following well known reduction from exact nearest neighbor searching to vertical ray shooting [2]. First, translate and scale the space in a way that the center of Q coincides with the origin and the side length of Q is 1. Let U denote the $(d+1)$-dimensional paraboloid $x_{d+1} = x_1^2 + \ldots + x_d^2$ and p_U denote the vertical projection of a given d-dimensional point p onto U. In order to answer nearest neighbor queries among the point set R_Q, we map each point $p \in R_Q$ into a hyperplane tangent to the paraboloid U at p_U. The nearest neighbor q' of a query point q corresponds to the first hyperplane hit by the vertical ray emanating downwards from the point q_U. Call this intersection point q'_U. Also, the squared distance $\|qq'\|^2$ (in the scaled space) is equal to the length of this vertical ray segment, $\|q_U q'_U\|$.

Since $R_Q \subset cB_Q \setminus B_Q$, we know that the distance between q and its nearest neighbor q' is at least $\sqrt{d} > 1$. Consequently, a point $q'_\varepsilon \in R_Q$ with $\|qq'_\varepsilon\| \leq \|qq'\| + \varepsilon$ is an approximate nearest neighbor of q and any hyperplane within vertical distance ε of q'_U corresponds to a valid approximate answer. In order to locate such a hyperplane, we perform a binary search involving $O(\log(1/\varepsilon))$ polytope membership queries, using the fact that $\|qq'\| = \Theta(1)$. Each search is answered by invoking the approximate polytope membership data structure, where the polytope is obtained using the hyperplanes tangent to the paraboloid and additional hyperplanes to bound the polytope into a box of constant size containing all relevant queries. $\square$

Combining the previous two lemmas, we obtain the following theorem.

Theorem 4.3 *Let $0 < \varepsilon \leq 1/2$ be a real parameter and S be a set of n points in $\mathbb{R}^d$. Given a data structure for approximate polytope membership in $(d+1)$-dimensional space with query time t_{d+1} and storage s_{d+1}, it is possible to preprocess S into an ANN data structure with $O(\log n + t_{d+1} \log(1/\varepsilon))$ query time and $O(n \log(1/\varepsilon) + n s_{d+1}/t_{d+1})$ storage.*

Proof: Construct a BBD-tree and the sets R_Q as in Lemma 4.1. For the nodes with $|R_Q| \leq t_{d+1} \log(1/\varepsilon)$, simply store the set R_Q and answer the corresponding queries by brute force. For the nodes with $|R_Q| > t_{d+1} \log(1/\varepsilon)$, use the construction from Lemma 4.2. The cell containing the query point can be located in $O(\log n + \log\log 1/\varepsilon)$ time. The bound for the total storage follows from the fact that the total number of representatives $\sum |R_Q| = O(n \log(1/\varepsilon))$ and the number of nodes with more than $t_{d+1} \log(1/\varepsilon)$ representatives is $O(n/t_{d+1})$. $\square$

By combining this with Theorem 1.1, we establish Theorem 1.2. While the previous data structure is not formally in the AVD model, we note that it is possible to convert it to the AVD model with an overhead of $O(\log(1/\varepsilon))$ in the storage and query time.

5 Concluding Remarks

In this paper, we have investigated a fundamental geometric retrieval problem, approximate polytope membership queries. We have presented the first nontrivial space-time tradeoffs for this problem. Our approach results in a very simple search structure, namely a quadtree whose leaves store a finite set of halfspaces. The construction algorithm, called SplitReduce, is also very simple and arguably of fundamental interest. It involves first computing an approximation of the intersection of a polytope and a quadtree box, and then splitting the box if this approximation involves an excessive number of halfspaces.

In spite of the simplicity of this algorithm, its analysis is nontrivial. Although we have demonstrated significant improvements over a naïve tradeoff, our best lower bound suggests that there is hope in believing that the algorithm's performance may be significantly better.

We have also presented a reduction of approximate nearest neighbor searching to approximate polytope membership queries. Surprisingly, this results in a remarkably simple construction for approximate nearest neighbor searching that provides significantly better space-time tradeoffs than any known structures. Further improvements in the space-time tradeoffs for approximate polytope membership queries will result in commensurate improvements in the efficiency of approximate nearest neighbor searching. Given the importance of convex polytopes in many other aspects of computer science, the study of approximate retrieval problems related to these objects is likely to be of broad interest.

References

[1] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Geometric approximation via coresets. In J. E. Goodman, J. Pach, and E. Welzl, editors, *Combinatorial and Computational Geometry.* 2005.

[2] P. K. Agarwal and J. Matoušek. Ray shooting and parametric search. *SIAM J. Comput.*, 22(4):794–806, 1993.

[3] S. Arya, G. D. da Fonseca, and D. M. Mount. A unifying approach to approximate proximity searching. In *Proc. 18th Euro. Symp. Algorithms (ESA)*, pages 374–385, 2010.

[4] S. Arya, T. Malamatos, and D. M. Mount. Space-time tradeoffs for approximate nearest neighbor searching. *J. ACM*, 57:1–54, 2009.

[5] S. Arya and D. M. Mount. Approximate range searching. *Comput. Geom.*, 17:135–163, 2000.

[6] J. L. Bentley, F. P. Preparata, and M. G. Faust. Approximation algorithms for convex hulls. *Commun. ACM*, 25(1):64–68, 1982.

[7] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2(2):121–167, 1998.

[8] M. Bădoiu, S. Har-Peled, and P. Indyk. Approximate clustering via core-sets. In *Proc. 34th ACM Symp. Theory of Comput. (STOC)*, pages 250–257, 2002.

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

[10] T. M. Chan. Closest-point problems simplified on the RAM. In *Proc. 13th ACM-SIAM Symp. Discrete Algorithms (SODA)*, pages 472–473, 2002.

[11] T. M. Chan. Faster core-set constructions and data-stream algorithms in fixed dimensions. *Comput. Geom.*, 35(1):20–35, 2006.

[12] T. M. Chan. Optimal partition trees. In *Proc. 26th ACM Symp. Comput. Geom. (SoCG)*, pages 1–10, 2010.

[13] K. L. Clarkson. Algorithms for polytope covering and approximation. In *Proc. 3rd Workshop Algorithms Data Struct. (WADS)*, pages 246–252, 1993.

[14] K. L. Clarkson. An algorithm for approximate closest-point queries. In *Proc. 10th ACM Symp. Comput. Geom. (SoCG)*, pages 160–164, 1994.

[15] K. L. Clarkson. Building triangulations using ε-nets. In *Proc. 38th ACM Symp. Theory of Comput. (STOC)*, pages 326–335, 2006.

[16] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 3rd edition, 2010.

[17] R. M. Dudley. Metric entropy of some classes of sets with differentiable boundaries. *Approx. Theory*, 10(3):227–236, 1974.

[18] C. A. Duncan, M. T. Goodrich, and S. Kobourov. Balanced aspect ratio trees: Combining the advantages of k-d trees and octrees. *J. Algorithms*, 38:303–333, 2001.

[19] J. Erickson, L. J. Guibas, J. Stolfi, and L. Zhang. Separation-sensitive collision detection for convex objects. In *Proc. 10th ACM-SIAM Symp. Discrete Algorithms (SODA)*, pages 327–336, 1999.

[20] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams. In *Proc. 41st Symp. Foundations of Computer Science (FOCS)*, pages 359–366, 2000.

[21] S. Har-Peled. A replacement for Voronoi diagrams of near linear size. In *Proc. 42nd Symp. Foundations of Computer Science (FOCS)*, pages 94–103, 2001.

[22] A. Kumar, Y. Sabharwal, and S. Sen. A simple linear time $(1+\varepsilon)$-approximation algorithm for k-means clustering in any dimensions. In *Proc. 45th Symp. Foundations of Computer Science (FOCS)*, pages 454–462, 2004.

[23] J. Matoušek and O. Schwarzkopf. On ray shooting in convex polytopes. *Discrete Comput. Geom.*, 10:215–232, 1993.

[24] J. Matoušek. Reporting points in halfspaces. *Comput. Geom.*, 2:169–186, 1992.

[25] J. Matoušek. Linear optimization queries. *J. Algorithms*, 14(3):432–448, 1993.

[26] J. Matoušek. On approximate geometric k-clustering. *Discrete Comput. Geom.*, 24(1):61–84, 2000.

[27] J. S. B. Mitchell and S. Suri. Separation and approximation of polyhedral objects. *Comput. Geom.*, 5:95–114, 1995.

[28] E. A. Ramos. Linear programming queries revisited. In *Proc. 16th ACM Symp. Comput. Geom. (SoCG)*, pages 176–181, 2000.

[29] Y. Sabharwal, S. Sen, and N. Sharma. Nearest neighbors search using point location in balls with applications to approximate Voronoi decompositions. *J. Comput. Sys. Sci.*, 72:955–977, 2006.