Approximate Polytope Membership Queries*

Sunil Arya† Guilherme D. da Fonseca‡ David M. Mount§

Abstract

In this paper, we consider an approximate version of a fundamental geometric search problem, polytope membership queries. Given a convex polytope P in $\mathbb{R}^d$, 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 allowed error ε. Previous solutions 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 a significant improvement to this tradeoff. For example, using the same storage as Dudley, we reduce the query time from $O(1/\varepsilon^{(d-1)/2})$ to $O(1/\varepsilon^{(d-1)/4})$.

To establish the relevance of our results, we introduce a reduction from approximate nearest neighbor searching to approximate polytope membership queries, providing significant improvements to the best known space-time tradeoffs for approximate nearest neighbor searching.

1 Introduction

The problem of determining whether a query point q lies within a convex polytope P, represented as the intersection of halfspaces, is dually equivalent to answering halfspace emptiness queries. Such queries find applications in many geometric problems, such as linear programming queries, ray shooting, nearest neighbor searching, and the computation of convex hulls. In dimension $d \leq 3$, it is possible to build a data structure of linear size that can answer such queries in logarithmic time. In higher dimensions, however, all exact data structures with roughly linear space take $\widetilde{O}(n^{1-1/\lfloor d/2 \rfloor})$ query time, 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 formulation.) Dudley [6] showed that, for any convex body, 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.

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, training a support vector machine, and approximate nearest neighbor searching.

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 in [2]. 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. 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 $\varepsilon \leq r \leq 1$.

*A more complete version will appear in STOC 2011.

†Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong, arya@cse.ust.hk.

‡Departamento de Informática Aplicada, Universidade Federal do Estado do Rio de Janeiro (UNIRIO), Brazil, fonseca@uniriotec.br. Research supported by FAPERJ grant E-26/110.091/2010 and CNPq/FAPERJ grant E-26/110.552/2010.

§Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, USA, mount@cs.umd.edu. Research supported by NSF grant CCR-0635099 and ONR grant N00014-08-1-1015.

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. 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. By iterating a suitable generalization of this construction, it is possible to produce a succession of structures of increasingly better search times.

Theorem 1 *Given a polytope P in $[-1,1]^d$, a parameter $\varepsilon > 0$, and a constant $\alpha \geq 2$, it is possible to answer ε-approximate polytope membership queries in time $t = 1/\varepsilon^{(d-1)/\alpha} \geq 1$ with storage space*

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

These space-time tradeoffs are presented visually in Figure 1(a). The lower bound will be discussed later.

The data structure and its construction are both extremely simple. The data structure consists of a quadtree where each leaf cell stores a set of halfspaces whose intersection approximates the polytope's boundary within the 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 ε-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 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 (see Figure 1(a) and Theorem 6).

To establish the relevance of our results, we introduce a reduction from approximate nearest neighbor searching to approximate polytope membership queries, providing significant improvements to the best known space-time tradeoffs for approximate nearest neighbor searching [1]. Our reduction implies the following improved space-time tradeoffs for ANN queries.

Theorem 2 *Given set of n points in $\mathbb{R}^d$, a parameter $\varepsilon > 0$, and a constant $\alpha \geq 2$. 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 [1] are illustrated in Figure 1(b). Although the connection between the polytope membership problem and ANN has been noted before by Clarkson [5], we are the first to provide a reduction that holds for point sets with unbounded aspect ratio. Details are omitted in this version.

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 may assume that P is presented succinctly to our algorithm, for example, as the output of Chan's coreset construction [3]. 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 $\Theta(1/\varepsilon^{(d-1)/2})$ facets. The problem of approximating a polytope with the minimum number of facets reduces to a set cover problem [7] and an $O(\log(1/\varepsilon))$ approximation can be obtained by a greedy algorithm. Clarkson [4] showed that if c is the smallest number of facets required to approximate P, then we can 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 run Clarkson's approximation algorithm on $Q \cap P$. This increases the query time by a negligible factor of $O(\log(1/\varepsilon))$.)

For completeness, we now describe Dudley's algorithm. Let P be a polytope, and let the *size* σ_P of P denote the side length of the smallest axis aligned box Q that contains P. We assume that the center of Q is the origin. Dudley's algorithm obtains an approximation polytope P' in the following manner. Let B be a ball of radius $\sigma_P\sqrt{d}$ centered at the origin. Place a set J 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 J. For each point $j \in J$, determine the nearest facet of P and add the corresponding halfspace to P'. We consider that each point $j \in J$ generates a facet of P' of diameter at most

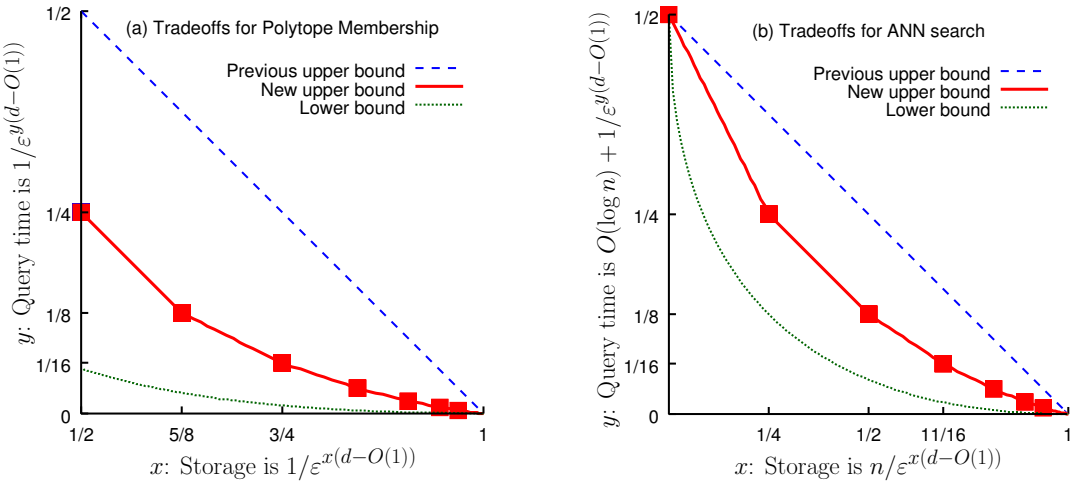

Figure 1: The multiplicative factor in the exponent of the $1/\varepsilon$ term for (a) polytope membership queries and (b) approximate nearest neighbor (ANN) queries. The $O(1)$ term in the exponent corresponds to a constant that does *not* depend on d.

$\sqrt{\varepsilon\sigma_P}$, even if these facets are coplanar. The following lemma follows from standard results on Dudley's algorithm.

Lemma 3 *Given a polytope P of size σ_P, Dudley's algorithm produces a polytope P' with $O((\sigma_P/\varepsilon)^{(d-1)/2})$ facets that approximates P. Furthermore, all facets of P' have diameter at most $\sqrt{\varepsilon\sigma_P}$.*

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 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 an approximation of $P \cap L$ with $t_L \leq t$ facets, where t_L is the minimum number of facets required to approximate $P \cap L$. The storage of this quadtree is defined as the total number of stored facets over all the leaf cells. Before we prove the whole 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 4 *The output of SplitReduce(Q) for $t \geq (\sigma_Q/\varepsilon)^{(d-1)/4} \geq 1$ 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 facets stored in L. We will show that $\sum_L t_L \leq (\sigma_Q/\varepsilon)^{(d-1)/2}$, which establishes the storage bound and proves the lemma.

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 3. In particular, the standard Dudley technique applied to a cell of size $\sqrt{\varepsilon\sigma_Q/c_1}$ produces a polytope with at most $c_D(\sigma_Q/c_1\varepsilon)^{(d-1)/4}$ facets, where c_D is the constant arising from Dudley's method. By choosing c_1 to be a sufficiently large constant, the number of facets is at most t and the termination condition of our algorithm implies that such a cell is not subdivided.

Let P_D be the polytope obtained by applying Dudley's algorithm to $P \cap Q$. Combining our assertion with Lemma 3, each facet of P_D intersects $O(1)$ leaf cells. We assign each facet to all leaf cells that it intersects. The correctness of Dudley's algorithm implies that the facets assigned to any cell L provides an approximation of $P \cap L$. Thus, t_L is no more than the number of Dudley facets assigned to L. Since the number of facets 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. □

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

Theorem 5 *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 4. 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 4, 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, $|\mathcal{L}| = O((\sigma_Q/\varepsilon)^{(d-1)/2})/t = O((\sigma_Q/\varepsilon)^{(d-1)\left(\frac{1}{2}-\frac{1}{\alpha}\right)})$. 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 $L \in \mathcal{L}$:

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

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

Note that the height of the quadtree obtained by SplitReduce is $O(\log(1/\varepsilon))$, since cells of size ε are never subdivided. Therefore, it is straightforward to locate the quadtree cell containing the query point in $O(\log(1/\varepsilon))$ time. Theorem 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 6 *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 [1]. It is based on constructing a hypercylinder that takes dimension k, $1 \leq k \leq d-2$ as a parameter, and bounding the storage requirements of our data structure for it. We carry out this analysis in Lemma 7.

Lemma 7 *Let k and t be integers, where $1 \leq k \leq d-2$, $1 \leq t \leq 1/\varepsilon^{(d-1)/2}$, and let $0 < \varepsilon \leq 1$. 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. (sketch) 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" is a polytope P_0 that approximates a $(d-k)$-dimensional ball of diameter Δ. If we choose $\Delta = O(\varepsilon t^{2/(d-k-1)})$, then 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 $[-1,1]^d$.

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 *small* if its size is at most Δ. By a packing argument, 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. □

Setting k appropriately, we obtain the lower bound in Theorem 6.

References

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

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

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

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

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

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

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