# Algorithms for Deciding Membership in Polytopes of General Dimension

Evangelos Anagnostopoulos, Ioannis Z. Emiris, Vissarion Fisikopoulos

## ▶ To cite this version:

HAL Id: hal-01897486

https://inria.hal.science/hal-01897486

Submitted on 19 Oct 2018

# Algorithms for Deciding Membership in Polytopes of General Dimension

Evangelos Anagnostopoulos[1], Ioannis Z. Emiris[1] and Vissarion Fisikopoulos[2]

[1] National and Kapodistrian University of Athens, Athens, Greece,
{aneva,emiris}@di.uoa.gr
[2] Oracle Corp., Greece, vissarion.fysikopoulos@oracle.com

**Abstract.** We study the fundamental problem of polytope membership aiming at large convex polytopes, i.e. in high dimension and with many facets, given as an intersection of halfspaces. Standard data-structures as well as brute force methods cannot scale, due to the curse of dimensionality. We design an efficient algorithm, by reduction to the approximate Nearest Neighbor (ANN) problem based on the construction of a Voronoi diagram with the polytope being one bounded cell. We thus trade exactness for efficiency so as to obtain complexity bounds polynomial in the dimension, by exploiting recent progress in the complexity of ANN search. We employ this algorithm to present a novel boundary data structure based on a Newton-like iterative intersection procedure. We implement our algorithms and compare with brute-force approaches to show that they scale very well as the dimension and number of facets grow larger.

## 1 Introduction

In geometric optimization, convex polytopes are very important objects appearing also as feasible regions in linear programming. Let us consider a convex polytope $P$ in H-representation, that is as the intersection of a finite set of linear inequalities: $P = \{x \in \mathbb{R}^d \mid Ax \leq b, A \in \mathbb{R}^{n \times d}, b \in \mathbb{R}^n\}$. An important question on such a polytope is that of point membership. We wish to preprocess $P$ in order to obtain a membership data structure which, given a query point $q$, efficiently decides whether $q$ lies inside or outside $P$. A decision can be reached by testing all $n$ inequalities for a complexity of $O(nd)$. This trivial approach is often a plausible exact solution, especially in the high-dimensional case. In order to design a more efficient algorithm in high dimension, we will focus on the approximate polytope membership problem where the membership data structure is allowed to answer incorrectly for points lying very close to the boundary of the polytope. A formal definition will be provided later in Section 2.2.

Algorithms used to solve combinatorial optimization problems, such as the ellipsoid, interior point or randomized methods (for the latter see [1]), usually rely on randomly sampling convex polytopes. The inner loop of such algorithms needs access to a membership or a boundary oracle, where the latter is the procedure that computes the intersection of a ray with the boundary of the polytope and

is equivalent to membership via binary search. The oracle specification means that we are not interested in how the solution is computed or of its computational complexity. Grötschel et al. [2] proposed the oracle model of computation and among other results they prove the polynomial time equivalence of basic oracles such as optimization, separation, and membership. This has become a commonly employed tool in combinatorial optimization mainly for studying the computational complexity of problems. Another important example of application is volume approximation [3,4] which has also an established connection to combinatorial optimization. For example, the volume of order polytopes gives the number of linear extentions of the associated partial order set.

From a practical point of view opening the oracle black box, in particular membership, and improving their complexity, implies improvements to the applicability of the aforementioned algorithms. For example, the first implementation of randomized algorithms that scale in high dimension appeared in [5]. Their approach relies on the standard random walks known as hit-and-run, which require a boundary oracle. Notice that, although this software can handle polytopes in spaces whose dimension goes up to 200, it cannot scale as efficiently for specific classes of polytopes with a large number of facets. In particular, it cannot approximate the volume of cross-polytopes of dimension 20 or more.

Here, we radically shift the aforementioned paradigm and, moreover, improve upon the complexity of membership and boundary data structures, when dimension $d$ is an input parameter. We exploit the approximate setting and allow ourselves to answer correctly within some approximation error $\epsilon$ and with some success probability. Our new paradigm uses a reduction to the Approximate Nearest Neighbor (ANN) problem, which is the most fundamental problem among those today with a practical, poly-time solution in high-dimensions.

*Previous Work.* There are two classical results for the approximate membership problem, both based on creating $\epsilon$-approximating polytopes and answering membership on them. Any convex body is $\epsilon$-approximated by a polytope with $O(1/\epsilon^{(d-1)/2})$ facets, which is asymptotically tight in the worst case [6]. This leads to a membership data structure with space and query complexity in $O(1/\epsilon^{(d-1)/2})$. Using a $d$-dimensional grid, membership takes constant time (assuming a model of computation that supports the floor function) and space grows to $O(1/\epsilon^{d-1})$ [7].

A relevant line of work on approximate membership in fixed $d$ uses space-time trade-offs [8,9] to achieve a space of $O(1/\epsilon^{(d-1)(1-(2\lfloor \log t \rfloor -2)/t)})$ with query time $O(\log(1/\epsilon)/\epsilon^{(d-1)/t})$, for trade-off parameter $t \geq 4$. In [10], again for fixed $d$, they opt for a hierarchy of ellipsoids selected by a sampling process on classical structures from the theory of convexity defined on the polytope. They achieve space $O(1/\epsilon^{(d-1)/2})$ with an optimal query time of $\log(1/\epsilon)$.

We present state-of-the-art approaches to ANN as we build atop of those for our oracles. There are many solutions to this problem, but in principle, methods that scale polynomially with $d$ belong to two categories. First, the well studied Locality Sensitive Hashing (LSH) [11]. The other category focuses on random projections [12], then uses fast algorithms in fixed dimension. Both achieve sub-

linear query time with (near-)linear storage, while scaling polynomially in $d$, and both have a probability of success $p$.

*Our contribution.* We describe a simple constructive reduction from the polytope membership problem to ANN, then show under which conditions this reduction holds for the respective approximate versions of the problems. This gives us the flexibility to exploit advances in the research of ANN in order to offer, the first (as far as the authors are aware) practical approximate polytope membership data structure in high dimension with complexity bounds polynomial in the dimension $d$ and sublinear in the number of inequalities $n$. This is our main result, in Theorem 5. We also present an application of this membership data structure for creating boundary data structures for H-polytopes. We implement and experimentally examine our algorithms; we illustrate that they scale well as dimension and number of facets grow larger. Our implementation is linked to the software of [5] for polytope volume, so as to provide faster oracles.

The rest of the paper is organized as follows. The next section discusses (approximate) membership and the reduction to ANN. Section 3 considers the boundary data structures. The implementation and experiments are in Section 4. We conclude with open questions.

## 2 Approximate Polytope Membership

We assume that the given H-polytope $P$ is full dimensional and that its representation is minimal, i.e. that it does not contain redundant inequalities.

We denote the $i$-th (in)equality of $P$ as $a_i x \leq b_i, 1 \leq i \leq n$. We associate each facet of the polytope with a corresponding (in)equality and denote it as $F_i$. Formally: $F_i = \{x \in P \mid a_i x = b_i\}$, $1 \leq i \leq n$. The hyperplanes that define non-empty $F_i$'s, i.e. for which $F_i \neq \emptyset$ are called non-redundant or supporting and we extend that label to their inequalities. We denote as $\partial P$ the boundary of $P$: $\partial P = \{x \in P \mid \exists i, \ 1 \leq i \leq n \ \text{s.t.} \ x \in F_i\}$.

### 2.1 Exact Polytope Membership Oracle

A reduction from the exact polytope membership problem to the exact nearest neighbor problem was established in [13], where it was shown that there is a connection between the boundaries of polytopes in $\mathbb{R}^d$ and power diagrams in $\mathbb{R}^{d-1}$. Power diagrams define a partition of the Euclidean space into a cell complex based on a set of spheres. Each sphere identifies a specific cell and that cell consists of all the points whose power distance is minimized for that sphere. The power diagram is a generalized Voronoi diagram, and coincides with the Voronoi diagram of the sphere centers if all spheres have equal radii.

**Theorem 1.** *[13, Thm.4] For any polyhedron $P \in \mathbb{R}^d$, which is expressible as the intersection of upper halfspaces, there exists an affinely equivalent power diagram in hyperplane $h_0 : x_d = 0$.*

A cell complex $C$ and a polyhedron $P \subset \mathbb{R}^{d+1}$ are said to be affinely equivalent if there exists a central or parallel projection $\phi$ such that, for each face $f$ of $C$, $f = \phi(g)$ holds for some face $g$ of $P$. This provides a reduction from ray shooting in a polyhedron to point location in a polyhedral complex. In the case of polytope membership, the polyhedral complex becomes a single cell (the polytope) and the power diagram becomes a Voronoi diagram. This provides a reduction from polytope membership to Nearest neighbor.

**Corollary 2** *Let $P \subset \mathbb{R}^d$ be a convex polytope described as the intersection of $n$ non-redundant halfspaces. For every point $p^* \in P \setminus \partial P$ it is possible to compute a set $S$ of $n + 1$ points such that, $p^* \in S$ and, given a query point $q$, the exact Polytope Membership test for a query point $q$ reduces to finding the Nearest Neighbor of $q$ among these $n + 1$ points.*

*Proof.* We initialize $S = \{p^*\}$. We will describe for completeness the procedure to compute the remaining $n$ points of $S$ such that the corresponding Voronoi diagram of these $n$ points and $p^*$ will have the polytope $P$ as the voronoi cell of $p^*$. These $n + 1$ points will be the points of the corollary.

For each facet $F_i$ and its corresponding hyperplane $H_i := a_i x = b_i$, $1 \leq i \leq n$, we compute the projection of $p^*$ on $H_i$ and denote it as $f_i$. Then, we compute the point $p_i$, $1 \leq i \leq n$, such that the line segment $(p^*, p)$ is perpendicular to $H_i$ and $d(p^*, H_i) = ||p^* - f_i||_2 = d(p_i, H_i)$, where $d(p, S) = \min_{x \in S} ||p - x||_2$. Equivalently, $p_i = f_i + (f_i - p^*)$.

We now have a set of points $S = \{p^*, p_1, \ldots, p_n\}$ of $n+1$ points that have the following property. In the Voronoi diagram of $S$, by construction, the cell that corresponds to $p^*$ is precisely the input polytope $P$. By the Voronoi property, the following holds: $q \in P \Leftrightarrow ||p^* - q||_2 \leq ||q - s||_2$, $\forall s \in S$. Polytope membership returns "YES" iff the nearest neighbor of $q$ is $p^*$. $\square$

*Remark.* A nearest neighbor computation or data structure on these $n + 1$ points of corollary 2 provides us with an exact Membership Oracle for the polytope $P$. We also emphasize that the choice of $p^* \in P$ is arbitrary. This means that a set $S$ satisfying the Corollary can be computed for each point $p^* \in P \setminus \partial P$.

## 2.2 Approximate Polytope Membership Oracle

Let us consider the following relaxation.

**Definition 3 (Approximate Polytope Membership Problem)** *Given a convex polytope $P \subset \mathbb{R}^d$ and an approximation parameter $\epsilon \in (0, 1)$, an $\epsilon$-approximate polytope membership query decides whether a query point $q \in \mathbb{R}^d$ lies inside or outside of $P$, but may return either answer if $q$'s distance from the boundary of $P$ is at most $\epsilon \cdot diam(P)$.*

We define $P^{-\epsilon} = \{x \in P \mid d(x, \partial P) > \epsilon \cdot diam(P)\}$. Obviously the aforementioned problem makes sense only when $P^{-\epsilon} \neq \emptyset$. Otherwise, we can always return "NO" for a query point $q$ and be correct.
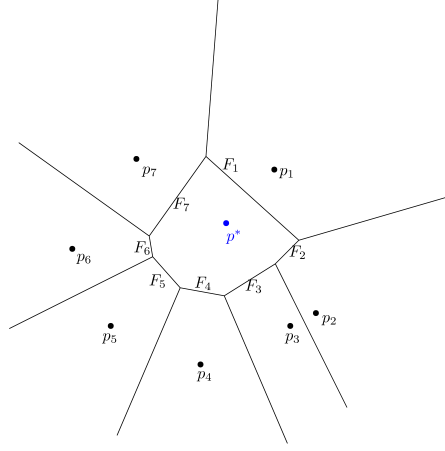
**Fig. 1.** A conceptual presentation of the constructive proof in the case of d=2. Each $p_i$ corresponds to the symmetric point of $p^*$ about the facet $F_i$.

**Theorem 4 (Approximate Membership Oracle (AMO)).** *Approximate Polytope Membership for an H-polytope $P$ and an approximation parameter $\epsilon$, such that $P^{-\epsilon} \neq \emptyset$, reduces to the ANN problem on the pointset $S = \{p^*, p_i : 1 \leq i \leq n\}$, where $p^* \in P^{-\epsilon}$ and the remaining $p_i$ are computed as in the proof of Corollary 2.*

*Proof.* Let $p^* \in P^{-\epsilon}$ and $S$ be the corresponding pointset of Lemma 2 for $P$. Let $\Delta(P) = \max\limits_{p_i \in S\setminus\{p^*\}} ||p_i - p^*||_2$. By construction, the following holds for $\Delta(P)$:

$2\epsilon \cdot diam(P) < \Delta(P) < 2diam(P)$. Let $q \in \mathbb{R}^d$ be a query point such that $||q - p^*|| < \frac{\Delta(P)}{2\epsilon}$. For any other $q' \in \mathbb{R}^d$, we return "NO", because $||q' - p^*||_2 \geq \frac{\Delta(P)}{2\epsilon} \Rightarrow ||q' - p^*|| > diam(P) \Rightarrow q' \notin P$. We distinguish two cases when $q \in P^{-\epsilon}$ and $q \in \{\mathbb{R}^d \mid q \notin P \ \wedge \ d(q, \partial P) > \epsilon \cdot diam(P)\}$.

– Let $q \in P^{-\epsilon}$, we wish to select an $\epsilon'$ for the ANN problem such that:

$$(1 + \epsilon') < ||p_i - q||_2 / ||p^* - q||_2 \tag{1}$$

Essentially, this would imply that $p^*$ is the nearest neighbor of $q$, while every $p_i \in S \setminus \{p^*\}$ is not an $\epsilon'$-NN of $q$.

Let $r_i = d(p^*, H_i) \geq \epsilon \cdot diam(P)$, where $H_i$ is the hyperplane defining facet $F_i$. By construction, $d(p^*, H_i) = d(p_i, H_i)$. It follows that the segment $p^*p_i$ has length $2r_i$, as it is perpendicular to $H_i$.

Next, we define the projection of $q$ on the line spanned by the segment $p^*p_i$ as $q_i = (p_i - p^*) \cdot q / ||p_i - p^*||_2$ and its distance from $H_i$ as $a_i = d(q_i, H_i) \geq \epsilon \cdot diam(P)$ Obviously now, as depicted in Fig. 2:

$$||p_i - q_i||_2 = r_i + a_i, \ \ ||p^* - q_i||_2 = r_i - a_i$$

Therefore,

$$||p_i - q||_2^2 = ||p_i - q_i||_2^2 + ||q - q_i||_2^2 = (r_i + a_i)^2 + k_i^2$$
$$||p^* - q||_2^2 = ||p^* - q_i||_2^2 + ||q - q_i||_2^2 = (r_i - a_i)^2 + k_i^2,$$

where $k_i = ||q - q_i||_2^2 < diam(P)$. It follows that,

$$\frac{||p_i - q||_2^2}{||p^* - q||_2^2} = \frac{(r_i + a_i)^2 + k_i^2}{(r_i - a_i)^2 + k_i^2} = 1 + \frac{4 r_i a_i}{(r_i - a_i)^2 + k_i^2} \geq$$
$$\geq 1 + \frac{4\epsilon^2 (diam(P))^2}{(r_i - a_i)^2 + k_i^2} \geq 1 + \frac{4\epsilon^2 (diam(P))^2}{2(diam(P))^2} \geq 1 + 2\epsilon^2$$

Substituting in (1), yields: $(1 + \epsilon') < \sqrt{1 + 2\epsilon^2} \Rightarrow \epsilon' < \sqrt{1 + 2\epsilon^2} - 1$.

– Let $q \in \{\mathbb{R}^d \mid q \notin P \ \wedge \ d(q, \partial P) > \epsilon \cdot diam(P)\}$. Assume the nearest neighbor of $q$ is $p_i \in S \setminus \{p^*\}$. Similarly, we are looking for an $\epsilon'$ such that:

$$(1 + \epsilon') < ||p^* - q||_2 / ||p_i - q||_2$$

This means $p^*$ cannot be an ANN of $q$. Now, like before:

$$\frac{||p^* - q||_2^2}{||p_i - q||_2^2} = \frac{(r_i + a_i)^2 + k_i^2}{(r_i - a_i)^2 + k_i^2} = 1 + \frac{4 r_i a_i}{(r_i - a_i)^2 + k_i^2} \geq$$
$$\geq 1 + \frac{4(\epsilon \cdot diam(P))^2}{(r_i - a_i)^2 + k_i^2} \geq 1 + \frac{4(\epsilon \cdot diam(P))^2}{2\left(\frac{2\Delta(P)}{2\epsilon}\right)^2} \geq$$
$$\geq 1 + \frac{4\epsilon^4 \cdot diam^2(P)}{2\Delta^2(P)} > 1 + \frac{4\epsilon^4 \cdot diam^2(P)}{4 \cdot diam(P)} >$$
$$> 1 + e^4 \cdot diam(P)$$

It follows that, $\epsilon' < \sqrt{e^4 \cdot diam(P)} - 1$.

Choosing $\epsilon' = \min\{\sqrt{e^4 \cdot diam(P)} - 1, \sqrt{1 + 2\epsilon^2} - 1\}$ and answering $\epsilon'$-ANN queries on this set solves the original problem, because if a query point $q \in P^{-\epsilon}$, then we have ensured that the $\epsilon'$-ANN data structure will correctly identify $p^*$ as the only approximate nearest neighbor of $q$. Similarly in a symmetric argument, for every $q \notin P$, such that $d(q, \partial P) > \epsilon \cdot diam(P)$, $p^*$ will not be an approximate nearest neighbor of $q$. Lastly, if $d(q, \partial P) \leq \epsilon \cdot diam(P)$ the response from the ANN data structure does not matter. Therefore, the reduction is complete. □

We now employ approaches for high-dimensional ANN to obtain a polynomial bound on the dimension by introducing a probability of success. Below, $\tilde{O}$ omits logarithmic factors.

**Theorem 5.** *[AMO in High Dimension] For an H-polytope $P \subset \mathbb{R}^d$ and an approximation parameter $\epsilon$, such that $P^{-\epsilon} \neq \emptyset$, we can solve the Approximate Polytope membership problem on $P$ by building a data structure on $P$ answering queries in $\tilde{O}(dn^{\rho + o(1)})$ time and using $\tilde{O}(n^{1 + \rho + o(1)} + dn)$ space, with a high probability of success, where $\rho = 1/(2(1 + \epsilon')^2 - 1)$ and $\epsilon' = \min\{\sqrt{e^4 \cdot diam(P)} - 1, \sqrt{1 + 2\epsilon^2} - 1\}$.*
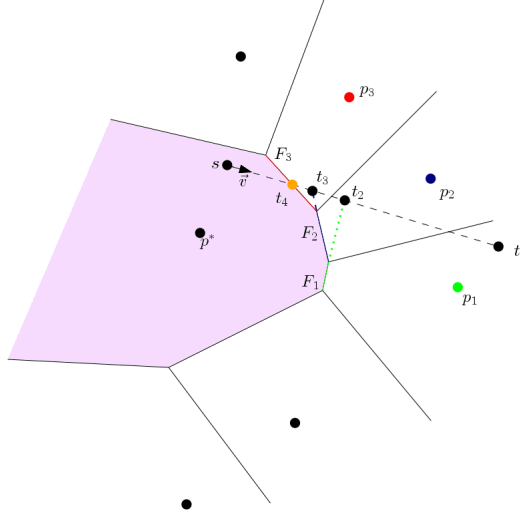
**Fig. 2.** $p_i$ corresponds to the symmetric point of $p^*$ about the facet $F_i$. We decompose the distances $||p^* - q||_2$ and $||p_i - q||_2$ and express them in terms of $a_i$ and $k_i$. Notice how $q \in P^{-\epsilon} \Rightarrow a_i \geq \epsilon \cdot diam(P)$ and how $k_i < diam(P)$, as $q$ cannot be a vertex.

*Proof.* The Chebyshev center of a polytope $P$ is the center of the largest inscribed ball. Formally: $\arg \min_{x \in P} \max_{y \in P} ||x - y||_2^2$. Let $c$ be the Chebyshev center of $P$ with radius $r$ and assume $c \notin P^{-\epsilon}$, in order to deduce an absurdity.

$$c \notin P^{-\epsilon} \Rightarrow r < \epsilon \cdot diam(P) \tag{2}$$

Take a point $c' \in P^{-\epsilon}$, as $P^{-\epsilon} \neq \emptyset$.

$$d(c', F_i) \geq \epsilon \cdot diam(P), \quad 1 \leq i \leq n \Rightarrow B(c', \epsilon \cdot diam(P)) \subset P \tag{3}$$

Combining (2) and (3) produces an absurdity as we have found a larger inscribed ball in $P$, contradicting the property of $c$. Therefore, $c \in P^{-\epsilon}$. We use $p^* = c$ as the starting point of the construction of the pointset $S$ in the proof of Theorem 4. Answering ANN queries on $S$ using the LSH data structure of [14], completes this proof. $\square$

*Remark.* Any high-dimensional ANN solution can be utilized in the last step of Theorem 3 and we can inherit its complexity and its properties.

## 3 Application to Polytope Boundary Problem

The polytope boundary problem consists of creating a data structure for an H-polytope $P$ such that, given a query ray emanating from inside the polytope, we can efficiently compute the point $p = r \cap \partial P$. It is possible to achieve query time in $O(\log n)$ by using space in $O(n^d / \log^{\lfloor d/2 \rfloor} n)$ [15]. The boundary oracle is dual to finding the extreme point in a given direction among a known pointset. This is $\epsilon$-approximated through $\epsilon$-coresets for measuring extent, in particular (directional) width, but requires a subset of $O((1/\epsilon)^{(d-1)/2})$ points [16]. The exponential dependence on $d$ or the linear dependence on $n$ make these methods of little practical use in high dimensions. Ray shooting has been studied in practice only in low dimensions, as well.

**Fig. 3.** An example of the boundary oracle converging to a solution. The query ray is $r = (s, \boldsymbol{v})$ and $t_4 = r \cap \partial P$ is the solution. $t_1, t_2, t_3, t_4$ were computed in sequence.

*Exact Polytope Boundary Oracle.* We now describe an iterative procedure for $P$ based on an exact nearest neighbor data structure E_MEM defined on the pointset $S$ of Corollary 2 that we described in section 2.2. This exact nearest neighbor data structure will act as the exact membership oracle for the polytope $P$. We call this algorithm `BoundaryOracle`.

*Finding the starting point.* The first step is to find a starting point $t_1$ such that $t_1 \in r$ and $t_1 \notin P$. We may use the intersection of $r$ with a bounding box around $P$. A bounding box of $P$ can be readily computed by solving $2d$ linear programs to compute the farthest points on $P$ along the coordinate directions.

*Finding the intersection point.* We obtain an efficient method following a derivative-like approach. Given starting point $t_1 \notin P$: let $p_i$ be the nearest neighbor of $t_1$ using the data structure defined for membership: $p_i = \text{E\_MEM}(t_1)$. Let $H_i$ be the hyperplane supporting the facet $F_i$ used to define $p_i$; $F_i$ separates the cell of $p_i$ from $P$ in the Voronoi diagram. Let $t_2 = (H_i \cap r)$. Iterate by computing $t_3, t_4, \ldots$, until membership decides $t_n \in P$.

**Lemma 6 (Correctness of algorithm `BoundaryOracle`)** *`BoundaryOracle` always converges to a solution for the boundary problem for a given polytope $P$.*

The proof of this lemma is presented in the Appendix A.1.

*Approximate Polytope Boundary Oracle.* Now, we define an approximate version of the polytope boundary problem.

**Definition 7 (Approximate Polytope Boundary Problem)** *Given a convex H-polytope $P \subset \mathbb{R}^d$ and an approximation parameter $\epsilon \in (0, 1)$, preprocess $P$ into a data structure such that, given a query ray $r \subset \mathbb{R}^d$ emanating*

*from inside $P$, it is possible to efficiently compute a point $r^* \in r$ such that $d(r^*, \partial P) \leq \epsilon \cdot diam(P)$.*

We make two additional changes to the algorithm presented in the previous section. First, we compare $t_i$'s and $t_{i+1}$'s distance from the ray's source point $s$. If the distance is not improved, then we discard the current $t_{i+1}$ and set it as $t_{i+1} = (t_i - s) - \frac{v}{||v||_2}\epsilon$. In other words, in this case we take an $\epsilon$-step from $t_i$ towards the ray's apex. The second change concerns termination. Now we stop when the approximate membership oracle identifies a point $t_i$ as being inside the polytope, or when the point $t_i$ lies in the opposite direction of the ray.

**Algorithm 1.** Approximate Boundary Oracle

```
Input: H-polytope P ⊂ ℝ^d, ray r (pair (s,v)), ε
Output: t ∈ ℝ^d s.t. t ∈ r and d(t,∂P) ≤ εdiam(P)
```

```
A_MEM = approximate membership oracle for P
```
$Q$ `= bounding_box(`$P$`)`
$t = Q \cap r$`;`
`do`
  $p_i$ `= A_MEM(t);`
  `if` $p_i$`==`$p$ `then return` $t + \frac{v}{||v||_2}\epsilon$`; end`
  $t_{prev} = t$
  $H = H_i$ `//facet corresponding to pi`
  $t = H \cap r$
  `if` $||t - s||_2 \geq ||t_{prev} - s||_2$ `then` $t = (t_{prev} - s) - \frac{v}{||v||_2}\epsilon$`; end`
  `if` $(t - s) \cdot v < 0$ `then return` $s + \frac{v}{||v||_2}\epsilon$`; end`
`while True;`

**Lemma 8 (Correctness of Algorithm 1)** *Algorithm 1 always converges to a solution for the approximate boundary problem.*

We present the proof of this lemma in the Appendix A.1.

## 4 Implementation and Experiments

*Implementation.* All of our code[3] is linked to the software of [5]. It is written in C++11 based on using the CGAL[4] library for the readily available data structures of d-dimensional objects, Eigen3 for some linear algebra computations and FALCONN[17] for the approximate nearest neighbor data structure. We remind the reader at this point that for a polytope $P(d, n, i)$ we compute $n + 1$ points, out of which one point $p^* \in P$ while all remaining $n$ points $p_i \notin P, 1 \leq i \leq n$. FALCONN offers LSH only for angular distances so in order to take advantage

---

[3] https://github.com/van51/volume_approximation
[4] http://www.cgal.org/

of that we use it in the following manner. We consider our pointset already centered around the internal point, in our case the origin. We build a FALCONN data structure using the Hyperplane LSH family and setting $k = 11, l = 1$, number of probes=40, when the number of facets $n \geq 10000$. Otherwise, we set them to $l = 1$, $k = 8$ and number of probes=150. $l$ corresponds to the number of hash tables built, $k$ corresponds to the number of hash functions used per hash table and number of probes is a parameter for the multi-probe LSH scheme [18]. The data structure is built for every computed point besides the internal one. Then, assuming that for a query $q$ FALCONN returns an approximate nearest neighbor guess $x_i$, we compare $d(x_i, q)$ to $d(p^*, q)$ and return the point closest to $q$ out of $x_i, p^*$. The parameters for FALCONN were selected manually, while trying to maintain a 90% success rate for membership.

*Datasets.* We experiment on a synthetic dataset consisting of high-dimensional polytopes with a large number of facets. In particular, for the following set of possible dimensions $\boldsymbol{d} = \{40, 100, 500, 1000\}$ and the following set of possible number of facets $\boldsymbol{n} = \{5000, 10000, 20000, 50000, 100000, 500000, 1000000\}$, we generate 5 polytopes for every combination of $\boldsymbol{d} \times \boldsymbol{n}$. Each polytope $P(d, n, i), d \in \boldsymbol{d}, n \in \boldsymbol{n}, i \in \{1, 2, 3, 4, 5\}$ lives in a $d$-dimensional Euclidean space and is described by $n$ inequalities of the form: $a_j x \leq 1000, 1 \leq j \leq n$, where $a_j \sim mod(U(0, 32767), 1000)$. The notation $U(i, j)$ denotes the uniform real distribution over $[i, j]$. By construction, each polytope contains the origin 0, which we use as the internal point needed by the approximate membership oracle. If that assumption was not satisfied, we could have computed an internal point either by solving a linear program or by computing an important point of the polytope, like the Chebyshev center.

*Evaluation protocol.* For both oracles we report pre-processing time, total query time, and success rate vs $n$ and $d$ as $n$ and $d$ vary in their respective sets $\boldsymbol{n}, \boldsymbol{d}$. Specifically for the boundary oracle we also report the average number of steps that it required in order to reach a solution and we also compute the min,max and average distances of the point returned from our approximate boundary oracle to the actual point that the exact ray shooting problem should have computed. We compare the query time to the naive approach of checking all $n$ facets of $P$. For the membership oracle we sample 1000 query points inside the polytope via the popular hit-and-run paradigm and then move these points sufficiently far from the origin so that they lie outside the polytope. This generates another 1000 points to form a total of 2000 points. Similarly for the boundary oracle we use 1000 query points in total.

*Results.* Table 1 depicts the total time in seconds for creating the approximate membership oracle on random polytopes for different values of $d, n$. Figure 4 depicts total time in seconds for all queries to be completed. Parameters were tuned such that the membership oracle achieved an accuracy of $> 90\%$, i.e. at least 9 out of 10 queries succeed on average. The results matched our expectations with regards to the behaviour of the oracles in high dimension, where we can see a huge difference in the query time, especially as the number of facets grows larger as well.

**Table 1.** Preprocessing time in seconds for membership oracle. This includes computing the $n + 1$ pointset and creating the ANN data structure on top of it.

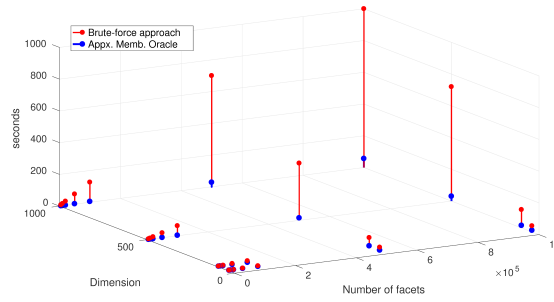| | | Number of facets | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | **5000** | **10000** | **20000** | **50000** | **100000** | **500000** | **1000000** |
| **Dimension** | **40** | 0.006s | 0.013s | 0.027s | 0.057s | 0.125s | 0.518s | 0.795s |
| | **100** | 0.015s | 0.035s | 0.057s | 0.121s | 0.230s | 1.005s | 1.885s |
| | **500** | 0.055s | 0.108s | 0.193s | 0.419s | 0.717s | 3.396s | 6.744s |
| | **1000** | 0.101s | 0.192s | 0.342s | 0.783s | 1.470s | 5.500s | 10.770s |



**Fig. 4.** Average timing results for 2000 queries for varying $n$ and $d$. Half of the queries were inside the random polytopes and half were outside.

# References

1. Bertsimas, D., Vempala, S.: Solving convex programs by random walks. J. ACM **51**(4) (2004) 540–556
2. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Volume 2 of Algorithms and Combinatorics. Springer (1988)
3. Dyer, M., Frieze, A., Kannan, R.: A random polynomial-time algorithm for approximating the volume of convex bodies. J. ACM (1991)
4. Lovász, L., Vempala, S.: Simulated annealing in convex bodies and an $O^*(n^4)$ volume algorithm. J. Comp. Syst. Sci. (2006)
5. Emiris, I., Fisikopoulos, V.: Efficient random-walk methods for approximating polytope volume. In: Proc. Symp. Comput. Geometry, Kyoto. (2014)
6. Dudley, R.: Metric entropy of some classes of sets with differentiable boundaries. J. Approximation Theory (1974)
7. Bentley, J., Preparata, F., Faust, M.: Approximation algorithms for convex hulls. Commun. ACM (1982)
8. Arya, S., da Fonseca, G.D., Mount, D.: Approximate polytope membership queries. In: Proc. ACM Symp. Theory of Computing, STOC, San Jose, USA. (2011)
9. Arya, S., da Fonseca, G.D., Mount, D.: Polytope approximation and the Mahler volume. In: Proc. ACM/SIAM Symp. Discr. Algorithms (SODA). (2012)
10. Arya, S., da Fonseca, G.D., Mount, D.: Optimal approximate polytope membership. In: Proc. ACM-SIAM Symposium on Discrete Algorithms. (2017)
11. Indyk, P., Motwani, R.: Approximate nearest neighbors: Towards removing the curse of dimensionality. In: Proc. STOC. (1998)

12. Anagnostopoulos, E., Emiris, I., Psarros, I.: Low-Quality Dimension Reduction and High-Dimensional Approximate Nearest Neighbor. In: 31st International Symposium on Computational Geometry (SoCG 2015). (2015)
13. Aurenhammer, F.: Power diagrams: Properties, algorithms and applications. SIAM J. Comput. (1987)
14. Andoni, A., Razenshteyn, I.: Optimal data-dependent hashing for approximate near neighbors. In: Proc. ACM STOC. (2015)
15. Ramos, E.: On range reporting, ray shooting and k-level construction. In: Proc. Symp. on Computational Geometry. (1999)
16. Agarwal, P., Har-Peled, S., Varadarajan, K.: Geometric approximation via coresets. In: Combinatorial and Computational Geometry (MSRI). (2005)
17. Andoni, A., Indyk, P., Laarhoven, T., Razenshteyn, I., Schmidt, L.: Practical and optimal lsh for angular distance. In: Proc. Conf. NIPS. (2015)
18. Lv, Q., Josephson, W., Wang, Z., Charikar, M., Li, K.: Multi-probe lsh: Efficient indexing for high-dimensional similarity search. In: Proc. Conf. VLDB. (2007)

# A    Appendix

## A.1    Proofs of section 3

Proof of lemma 6.

*Proof.* Let $t_1, t_2, \ldots$ denote the sequence of successive points computed on the ray $r$ by the above algorithm. Let $x_1, x_2, \ldots$ be a sequence of points in $S$, each representing the nearest neighbor of the point $t_i$. This means that the nearest neighbor of $t_i$ was $x_i$ at the $i$-th step. We assume without loss of generality that each $t_i$ has a single nearest neighbor, because otherwise it would mean that $t_i$ falls on the intersection of a line (the ray), a Voronoi facet and a supporting hyperplane which is highly degenerate. However, even in that case we could consider every nearest neighbor of the point and take the one that improves the distance the most. For correctness, assume that we have reached the $i$-th step. There are two cases for $t_{i+1}$. Either it lies on $\partial P$ in which case the membership data structure E_MEM will identify it as being inside and the algorithm will terminate. Otherwise, by convexity of the cell of $x_i$, $t_{i+1}$ lies between $\partial P$ and $t_i$, since $t_{i+1}$ lies on an "extension" of the facet (meaning on $H_i \setminus F_i$) between the cell of $x_i$ and $P$. Since $H_i \setminus F_i$ cannot belong to a Voronoi facet, $t_{i+1}$ will always belong to a new Voronoi cell. Therefore the sequence $x_i$ will not have any repeating points and the algorithm will eventually reach $\partial P$ where the iteration will stop and return $\partial P \cap r$.

Proof of lemma 8.

*Proof.* Observe that the successive points $t_i$ lying on the ray $r$ are always improving the distance to the ray's apex, by a factor of at least $\epsilon$. Additionally, by definition, the ray's apex always lies inside $P$. We separate two cases for the ray's apex, which we will from now on denote as $s$.

1. $d(s, \partial P) > \epsilon \cdot diam(P) + \epsilon$

2. $d(s, \partial P) \leq \epsilon \cdot diam(P) + \epsilon$

In case 1, the algorithm will eventually reach a point $t_i$, after performing a number of $\epsilon$-steps, such that $t_i \in P$ and $d(t_i, \partial P) \geq \epsilon \cdot diam(P)$. Since the ray's apex $s$ is at distance $> \epsilon \cdot diam(P) + \epsilon$ from $\partial P$ this will happen while $(t_i - s) \cdot v > 0$. In this case we return point $t_i + \frac{v}{||v||_2}\epsilon$ which lies within distance $\epsilon \cdot diam(P)$ from $\partial P$.

In case 2, the point $t_i$ will either reach $d(t_i, \partial P) > \epsilon \cdot diam(P)$ and will be identified as being inside and in which case the algorithm will correctly return point $t_i + \frac{v}{||v||_2}\epsilon$. Alternatively, it will take an $\epsilon$-step and move to the opposite direction of the ray. In that case, $s$ is identified as lying at distance at most $\epsilon diam(P) + \epsilon$ from $\partial P$ and in which case we return point $s + \frac{v}{||v||_2}\epsilon$ which lies in $r$ at distance $< \epsilon \cdot diam(P)$ from $\partial P$.

Eventually, the algorithm returns point $t$: $t \in P$ and $d(t, \partial P) \geq \epsilon \cdot diam(P)$.