NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

MASTER THESIS

Polytope Membership in High Dimension

Author:
Evangelos ANAGNOSTOPOULOS

Supervisor:
Prof. Ioannis Z. EMIRIS

A thesis submitted in fulfilment of the requirements for the degree of Master of Science

in the

Graduate Program in Logic, Algorithms and Computation
$\mu\prod\lambda\forall$
Department of Mathematics

June 16, 2017

National and Kapodistrian University of Athens

Abstract

Department of Mathematics

Master of Science

Polytope Membership in High Dimension

by Evangelos ANAGNOSTOPOULOS

Polytopes in optimization and sampling problems are usually given by implicit representations through oracles. The most basic oracle is the polytope membership oracle which can identify whether a query point q lies inside P or not and is often used as the basis for more complex oracles, such as the separation oracle or the boundary oracle. In this work we aim to design, implement and analyse algorithms for approximating the membership oracle in polytopes given as the intersection of halfspaces in high dimension, by trading exactness for efficiency. Previous approaches were based on classic polytope approximation techniques which, however, have complexity that scales exponentially in the dimension and are, thus, intractable in high dimension. We establish a straightforward reduction from approximate polytope membership to approximate nearest neighbor search among points and obtain complexity bounds polynomial in the dimension, by exploiting recent progress in the complexity of nearest neighbor search. We then employ this new membership oracle to obtain a solution for the boundary oracle in high dimension. Lastly, we evaluate our algorithms experimentally and report results.

Acknowledgements

First and foremost I would like to thank my family for their continuous support. I would also like to thank my advisor Ioannis Z. Emiris and my co-advisor Vissarion Fisikopoulos for their patience and guidance. Furthermore, I would like to thank Ioannis Psarros and the remaining members of the ErGA lab for the helpful discussions.

Contents

Chapter 1

Introduction

1.1 Prerequisites

Let us start with some definitions about our objects of interest. We narrow our attention to convex polytopes defined as the bounded intersection of a finite number of halfspaces. This is a standard convex polytope representation. These polytopes are called *H-polytopes* and are defined as follows:

$$P = \{x \in \mathbb{R}^d \mid Ax \leq b,\ A \in \mathbb{R}^{n \times d},\ b \in \mathbb{R}^n\}.$$

The notation $Ax \leq b$ indicates n simultaneous linear inequalities defined by the rows of matrix A and b; each inequality constraint defines a halfspace where P must lie. We denote the i-th inequality, $1 \leq i \leq n$, as $a_i x \leq b_i$, where a_i is the i-th row of A and b_i is the i-th coordinate of b.

If P does not change when some halfspace constraint is ignored, then the respective hyperplane is called redundant; otherwise, the hyperplane is called non-redundant or supporting hyperplane. Clearly, the supporting hyperplanes are precisely those for which at least one point in P satisfies the corresponding equality constraint. The supporting hyperplanes also define the facets of the polytope P. We associate each *facet* of the polytope with its corresponding (in)equality and denote it as F_i. Formally:

$$F_i = \{x \in P \mid a_i x = b_i\},\ \ 1 \leq i \leq n$$

Additionally, we denote as ∂P the boundary of the polytope, i.e.:

$$\partial P = \{x \in P \mid \exists i,\ \ 1 \leq i \leq n \text{ s.t. } x \in F_i\}$$

We are interested in efficient solutions to the following fundamental polytope problems. Notice how each problem wants to answer a specific question about the polytope. Solutions to these problems are oracles which provide implicit access to the polytope, by answering the respective questions.

Definition 1 (Polytope Membership Problem)**.** Given a convex H-polytope $P \subset \mathbb{R}^d$, preprocess it into a data structure, so that it is possible to efficiently determine whether a query point $q \in \mathbb{R}^d$ lies in P or not.

Definition 2 (Polytope Boundary Problem)**.** Given a convex H-polytope $P \subset \mathbb{R}^d$, preprocess it into a data structure, so that it is possible to efficiently compute the point $p = r \cap \partial P$ given a query halfline, or ray, $r \subset \mathbb{R}^d$, where ∂P is the polytope boundary.

The polytope boundary problem is also referred to as *ray shooting* in the literature.

One obvious solution to these problems is to check the query q against all n inequalities of the defining halfspaces of P in a brute-force approach. Although this method may sound trivial, it is often a plausible solution in the exact setting, especially in the high-dimensional case.

1.2 Motivation

Our motivation stems from problems in geometric optimization. Let us mention a couple of important applications for the polytope oracles above. A concrete application is in randomly sampling convex H-polytopes, which is the main paradigm for approximating the volume of H-polytopes in polynomial time in general dimension. The inner loop of such algorithms contains the polytope operations studied in this paper. Specifically, randomized polynomial-time algorithms for volume approximation, e.g. [DFK91; LV06], rely on random walks that need an access to a membership or boundary oracle. The first implementation of randomized algorithms that scale in high dimension appeared in [EF14]. 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.

In this paper, we improve upon the complexity of these methods, specifically when the dimension d is an input parameter, by allowing ourselves to answer correctly within some approximation error ϵ and some success probability p. In order to achieve that we use a reduction to the problem of the approximate nearest neighbor (ANN) problem, which is the most famous problem with a solution in the high-dimensional setting, as we will see in the next section.

Definition 3 (Approximate Nearest Neighbor (ANN) Problem)**.** Given a point set $X \subset \mathbb{R}^d$ and an approximation parameter $\epsilon \in (0, 1)$, preprocess X into a data structure, so that, given a query point $q \in \mathbb{R}^d$, it is possible to efficiently find a point $x^* \in X$ such that $||q - x^*||_2 \leq (1 + \epsilon) \min_{x \in X} ||q - x||_2$

1.3 Previous work

Let us start with the Polytope Membership problem. In the exact setting, a typical solution is the one mentioned above where one tests all n inequalities for a complexity of $O(nd)$. In the approximate setting, where we allow an approximation error ϵ, there are two classical results. Dudley [Dud74] showed that any convex body $K \subset \mathbb{R}^d$ is ϵ-approximated by a polytope P with $O(1/\epsilon^{(d-1)/2})$ facets. This bound is asymptotically tight in the worst case. Answering membership queries on the ϵ-approximating polytope implies a (trivial) data structure for the approximate polytope membership problem with space and query time of $O(1/\epsilon^{(d-1)/2})$. Bentley et al [BPF82] gave another construction for an ϵ-approximating polytope by creating a d-dimensional grid with cells of size $\Theta(\epsilon \cdot diam(K))$ and storing for every column along each axis, the two extreme values where the column intersects K. Answering membership queries on this ϵ-approximating polytope can be done in constant time (assuming a model of computation that supports the floor function) and the corresponding storage grows to $O(1/\epsilon^{d-1})$.

More recent work on Approximate Polytope Membership is studied in the form of space-time trade-offs [AFM11; AFM12b; AFM12a] for fixed dimension. In these approaches the authors present a data structure called *SplitReduce*, which, given a parameter t, hierarchically subdivides the space using a quadtree until each cell lies entirely in P, or lies entirely outside P, or the intersection of the cell and P is approximated by an ϵ-approximating polytope using Dudley's approach with at most t halfspaces. Queries are then answered by descending the quadtree and checking the type of the cell of the appropriate leaf. If the cell lies entirely inside or outside we answer appropriately. Otherwise all (at most) t inequalities approximating the polytope locally are checked in a brute-force way, providing us with an answer. t is the controlling time-space trade-off parameter and the authors show that the quadtree approach achieves a space of $O(1/\epsilon^{(d-1)/2})$ with a query time $t = \Theta(\log(1/\epsilon)/\epsilon^{(d-1)/8})$.

Even more recently the same authors [AFM17], again for fixed dimension, dropped the quadtree approach in favour of a data structure employing a hierarchy of ellipsoids selected by a sampling process on classical structures from the theory of convexity defined on the polytope. Their new proposed data structure achieves space $O(1/\epsilon^{(d-1)/2})$ with an optimal query time of $\log(1/\epsilon)$.

Concerning the boundary oracle, in exact form, it is possible to achieve query time in $O(\log n)$ by using space in $O(n^d/\log^{\lfloor d/2 \rfloor} n)$ [Ram99]. The boundary oracle is dual to finding the extreme point in a given direction among a known pointset. This is ϵ-approximated through ϵ-coresets for measuring extent, in particular (directional) width, but requires a subset of $O((1/\epsilon)^{(d-1)/2})$ points [AHPV05]. 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, e.g., in 6-dimensional polytopes described by the set of their vertices [ZY13].

We will also present now current state-of-the-art approaches to the ANN problem as we will build atop of those for the design of our oracles. There are many solutions to this problem, but in principle, methods that scale well (polynomially) with the dimension d belong to two categories.

The first major category is the well studied approach of Locality Sensitive Hashing (LSH), first introduced by Indyk and Motwani in 1998 [IM98]. This method works by exploiting a data dependent space partition through the utilization of locality sensitive hash functions which are functions that are more likely to assign the same value to "similar" objects, under some notion of similarity.

The other major category focuses on random projections to create a drastically low dimensional representation of the pointset[AEP15] and then uses techniques that work well for fixed dimension, like BBD-trees[Ary+98]. Both approaches achieve sublinear query time with (near-)linear storage, while scaling polynomially with regards to the dimension. In addition, both approaches get to that by allowing a probability of success p. This is the same probability of success that we will inherit in our oracle.

1.4 Our contribution

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

We describe a simple constructional reduction from the Polytope Membership Problem to the problem of Nearest Neighbor and 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 improving the behaviour of our polytope membership oracle. We demonstrate this flexibility by using a high dimensional solution to the ann problem in order to offer a practical approximate polytope membership oracle in high dimension with complexity bounds polynomial in the dimension d and sublinear in the number of inequalities n.

We also present an iterative framework for creating boundary oracles for H-polytopes and describe a concrete instantiation of it. We then modify this algorithm to work in an approximate setting. Both approaches use the polytope membership oracle that we described above and which is based on (approximate) nearest neighbors.

We implement and experimentally examine our algorithms.Our implementation is linked to the software of [EF14] for polytope volume, so as to provide faster oracles. It exhibits encouraging improvements in the practical complexity of volume approximation.

The rest of the paper is organized as follows. The next chapter discusses polytope membership and the reduction to Nearest Neighbor, focusing on the approximate setting. Chapter 3 considers the boundary oracle and various algorithms for answering it in the approximate setting. The implementation and experiments are discussed in Chapter 4. We conclude in Section 5 with open questions.

Chapter 2

Polytope Membership

In this chapter we focus on the fundamental problem of testing membership in H-polytopes in general dimension. We first present a straightforward reduction to Nearest Neighbor search in the exact form and then analyse the conditions under which this reduction holds in the approximate setting. Then, we will exploit approaches in approximate nearest neighbor in high dimension that will associate a probability of success with the membership oracle in order to make the problem feasible in high dimension.

2.1 Exact Polytope Membership Oracle

We will establish here a reduction from the exact polytope membership problem to the exact nearest neighbor problem. This was established in [Aur87], where it was shown that there is a connection between the boundaries of polytopes in $\mathbb{R}^d$ and power diagrams in $\mathbb{R}^{d-1}$. In order to present this reduction we will need some definitions.

Definition 4 (Cell complex)**.** A cell complex C in $\mathbb{R}^d$ is a partition of $\mathbb{R}^d$ into finitely many polyhedra.

Definition 5 (Power Diagram)**.** A power diagram is a partition of the Euclidean space into into a cell complex defined from a set of spheres, where the cell for a given sphere S consists of all the points for which the power distance to S is smaller than the power distance to the other spheres. The power distance of a point p to a sphere S of center c and radius r is defined as

$$pow\_dist(p, S) = ||p - c||_2^2 - r^2$$

The power diagram is a form of generalized Voronoi diagram, and coincides with the Voronoi diagram of the sphere centers in the case that all the spheres have equal radii. For an example of a power diagram, check figure 2.1.

Now, let $\mathbb{R}^d$ be spanned by the coordinate axes $x_1, \ldots, x_d$ and let h_0 denote the hyperplane $x_d = 0$. Let a cell complex C be contained in some hyperplane of $\mathbb{R}^{d+1}$. Then, C and a polyhedron $P \subset \mathbb{R}^{d+1}$ are said to be affinely equivalent if there exists a central or parallel projection ϕ such that, for each face f of C, $f = \phi(g)$ holds for

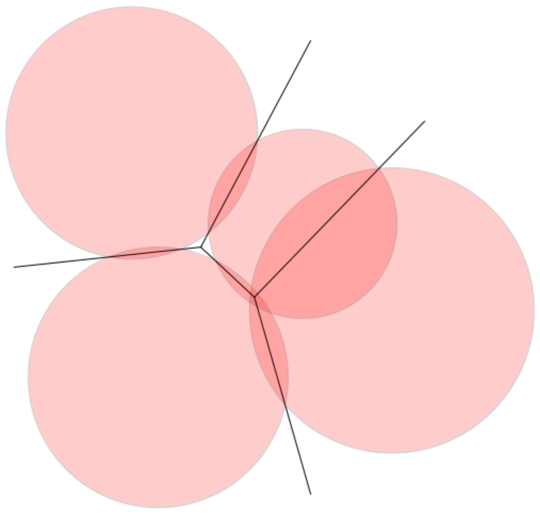

FIGURE 2.1: An example of a power diagram of 5 different circles in two dimensions. It can be seen as a generalization of the Voronoi diagram of the centers of the spheres.

some face g of P. The following proposition refers to upper halfspaces, which are precisely those that are unbounded in the positive direction of x_d, in other words these halfspaces contain a halfline directed in the positive direction of x_d.

Proposition 6. [Aur87, 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 $h_0 : x_d = 0$.*

The above result provides a reduction from ray shooting in a polyhedron to point location in a polyhedral complex. In the special 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 the problem of nearest neighbor.

Corollary 7. *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. As mentioned in the introduction, the defining halfspaces of P correspond to hyperplanes which define the facets of the polytope, as they are non-redundant by assumption.

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,

$$p_i = f_i + (f_i - p^*)$$

Conceptually we can think of each point p_i as the symmetric point of p^* about H_i.

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, 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^*. □

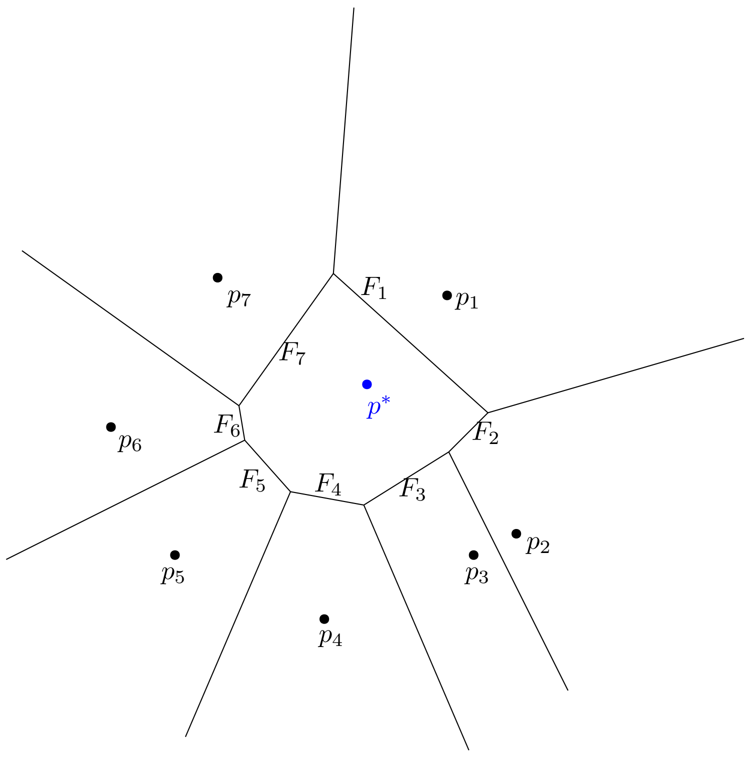

FIGURE 2.2: Construction of a pointset S from Corollary 7 for a 2d polytope. $p^* \in P \setminus \partial P$ is the selected internal point. Each p_i corresponds to the symmetric point of p^* about the facet F_i.

A nearest neighbor computation or data structure on these $n + 1$ points of corollary 7 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

In order to solve the Polytope Membership approximately, we allow for a relaxation of the problem definition. Given an approximation parameter ϵ, we still wish to decide membership in H-polytopes, in other words whether a query point $q \in P$, but we permit ourselves to answer either way if q's distance from ∂P is at most $\epsilon \cdot diam(P)$, where $diam(\cdot)$ stands for the diameter, i.e. the longest segment between two points in the polytope. The diameter is obviously obtained by two vertices of the polytope.

Definition 8 (Approximate Polytope Membership Problem)**.** Given a convex polytope $P \subset \mathbb{R}^d$ and an approximation parameter $\epsilon \in (0, 1)$, an ϵ-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.

Lemma 9. *Approximate Polytope Membership for an H-polytope P and an approximation parameter ϵ, 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 7.*

Proof. Let $p^* \in P^{-\epsilon}$ and S be the corresponding pointset from Lemma 7 for P. In addition, let $\Delta(P) = \max_{p_i \in S \setminus \{p^*\}} ||p_i - p^*||_2$. By construction, the following holds for $\Delta(P)$:

$$2\epsilon \cdot diam(P) < \Delta(P) < 2diam(P) \tag{2.1}$$

Let $q \in \mathbb{R}^d$ be a query point.

$$\begin{aligned} ||q - p^*||_2 \geq \frac{\Delta(P)}{2\epsilon} \Rightarrow & \quad \text{, by 2.1} \\ ||q - p^*|| > diam(P) \Rightarrow & \\ q \notin P & \end{aligned}$$

Therefore we will now focus on the case where $||q - p^*|| < \frac{\Delta(P)}{2\epsilon}$, as otherwise we will return "NO" after a simple test. We will distinguish the 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 ϵ' for the ANN problem such that:

$$(1 + \epsilon') < \frac{||p_i - q||_2}{||p^* - q||_2} \tag{2.2}$$

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

$$q_i = \frac{(p_i - p^*) \cdot q}{||p_i - p^*||_2}$$

and its distance from H_i as

$$a_i = d\left(q_i, H_i\right) \geq \epsilon \cdot diam(P)$$

Obviously now, as depicted in Figure 2.3:

$$\begin{aligned} ||p_i - q_i||_2 &= r_i + a_i \\ ||p^* - q_i||_2 &= r_i - a_i \end{aligned}$$

Therefore,

$$\begin{aligned} ||p_i - q||_2^2 = ||p_i - q_i||_2^2 + ||q - q_i||_2^2 = (r_i + a_i)^2 + k_i^2 \\ ||p^* - q||_2^2 = ||p^* - q_i||_2^2 + ||q - q_i||_2^2 = (r_i - a_i)^2 + k_i^2, \end{aligned}$$

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

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

Substituting in (2.2), yields:

$$\begin{aligned} (1 + \epsilon') < \sqrt{1 + 2\epsilon^2} \Rightarrow \\ \epsilon' < \sqrt{1 + 2\epsilon^2} - 1 \end{aligned}$$

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

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

Conceptually this means that p^* cannot be an ANN of q. Now, like before:

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

It follows that,

$$\epsilon' < \sqrt{e^4 \cdot diam(P)} - 1$$

Choosing $e' = \min\{\sqrt{e^4 \cdot diam(P)} - 1, \sqrt{1 + 2\epsilon^2} - 1\}$ and answering ϵ'-ANN queries on this set solves the original problem, because if a query point $q \in P^{-\epsilon}$, then we have ensured that the ϵ'-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. $\square$

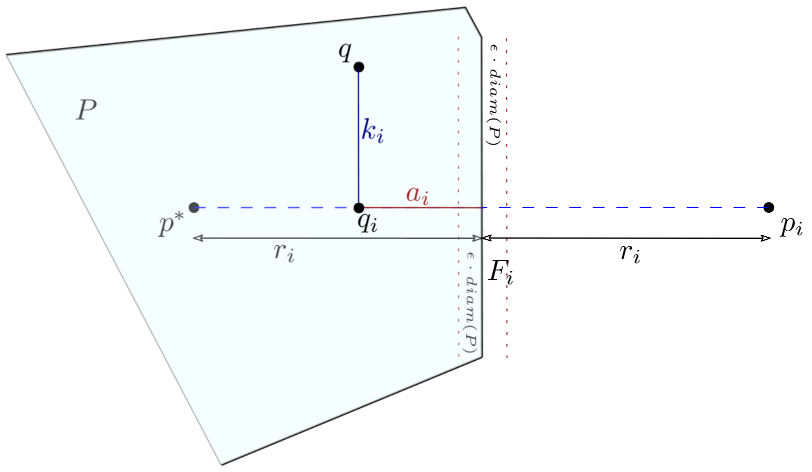

FIGURE 2.3: p_i corresponds to the symmetric point of p^* about the facet F_i. We decompose the distances $||p^* - q||_2$ and $||p_i - q||_2$ and express them in terms of a_i and k_i. Notice how $q \in P^{-\epsilon} \Rightarrow a_i \geq \epsilon \cdot diam(P)$ and how $k_i < diam(P)$, as q cannot be a vertex.

We can now employ approaches for high-dimensional ANN in order to obtain a polynomial bound on the dimension by further allowing the data structure to answer correctly with a probability (varying on the selection of the ANN algorithm). Below, $\tilde{O}$ omits logarithmic factors.

Theorem 10 (Approximate Polytope Membership Oracle). *For an H-polytope $P \subset \mathbb{R}^d$ and an approximation parameter ϵ, 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 with a probability of success $p = 1/(2(1+\epsilon)^2 - 1)$, using $\tilde{O}(n^{1+\rho+o(1)} + dn)$ space.*

Proof. The Chebyshev center of a polytope P is defined as the center of the largest inscribed ball of P. Formally it is given by:

$$\arg\min_{x \in P} \max_{y \in P} ||x - y||_2^2$$

Many points inside the polytope can be Chebyshev centers, but they will all share the same radius.

Let c be the Chebyshev center of P with radius r and assume that $c \notin P^{-\epsilon}$, in order to deduce an absurdity.

$$c \notin P^{-\epsilon} \Rightarrow r < \epsilon \cdot diam(P) \tag{2.3}$$

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

$$d(c', F_i) \geq \epsilon \cdot diam(P), \;\; 1 \leq i \leq n \Rightarrow B(c', \epsilon \cdot diam(P)) \subset P \tag{2.4}$$

Combining (2.3) and (2.4) produces an absurdity as we have found a larger inscribed ball in P, contradicting the property of c. Therefore, $c \in P^{-\epsilon}$.

We will use $p^* = c$ as the starting point of the construction of the pointset S in Lemma 9. Answering ANN queries on that pointset S using the LSH data structure of [AR15] completes the proof of this theorem. □

Chapter 3

Polytope Boundary

This chapter focuses on the boundary problem for H-polytopes. Like in the previous section we will show a solution for the exact case and then we will define an approximate version of this problem. Afterwards, we will make the jump to the high dimensional case where we will answer correctly under some probability p, like in the corresponding membership version.

Recall that a boundary oracle (i.e.. a solution to the boundary problem) given a ray r emanating from inside P, returns $p^* = r \cap \partial P$. A ray, or halfline, in $\mathbb{R}^d$ is defined by a point in $\mathbb{R}^d$, which is the apex, and a direction (unit) vector in $\mathbb{R}^d$. We will usually denote the ray's apex as s and the ray's direction as v.

Next we propose and study a generalized framework for the polytope boundary problem for the case where r is a random direction and analyse one specific instantiation of it.

3.1 Polytope Boundary Framework

Inspired by widely used iterative root-finding methods for polynomial equations, such as bisection or derivative-based methods, we introduce a framework for solving the polytope boundary problem. The framework consists of two steps:

1. Selecting a starting point t_1 such that $t_1 \notin P$, but $t_1 \in r$.

2. Iteratively computing points $t_i \in r$, $i \geq 2$, such that each point t_i is closer to ∂P (or to the ray's apex) than t_{i-1}, until a membership oracle decides that $t_i \in P$.

A number of strategies can be chosen for both of the two steps defined above. The conceptual similarity between root-finding methods is that the proposed framework considers the ray as an arbitrary function and starts with a random point on the function (ray) and iteratively closes in on the "root" of the function (ray) i.e.. the point where it meets the boundary of the polytope.

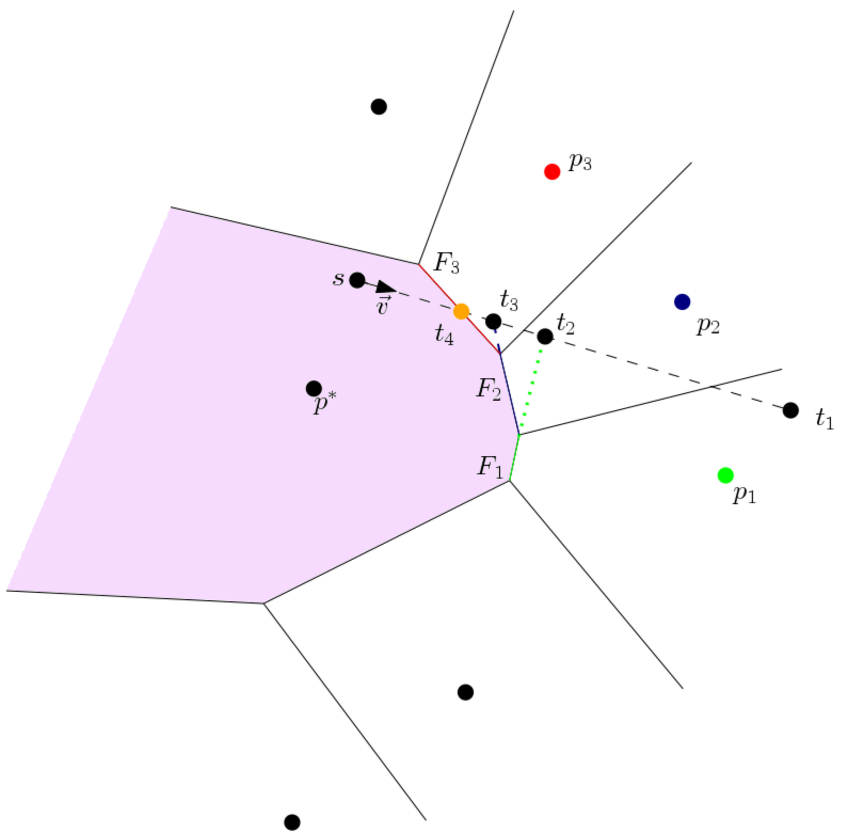

FIGURE 3.1: An example of the boundary oracle converging to a solution. The query ray is $r = (s, \vec{v})$ and $t_4 = r \cap \partial P$ is the solution. t_1, t_2, t_3, t_4 were computed in sequence.

3.2 Exact Polytope Boundary Oracle

We will now describe a concrete instantiation of this generalized procedure for a polytope P based on an exact nearest neighbor data structure `E_MEM` defined on the pointset S of Corollary 7 that we described in section 2.2. This exact nearest neighbor data structure will act as the exact membership oracle for the polytope P.

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

Below is the pseudocode for the exact version of the problem.

Algorithm 1: Exact boundary oracle, using exact membership

```
Input: H-polytope $P \subset \mathbb{R}^d$, ray r (pair (r.apex, r.direction))
Output: $t \in \mathbb{R}^d$ s.t. $t \in r \cap \partial P$
1  E_MEM = membership oracle for P, based on NN approach of Chapter 2;
2  Q = bounding_box(P);
3  $t = Q \cap r$;
4  do
5      $t_{prev} \leftarrow t$;
6      $p_i \leftarrow$ E_MEM($t$);
7      $H \leftarrow H_i$ (facet corresponding to $p_i$);
8      $t \leftarrow H \cap r$;
9  while E_MEM($t$) $\neq p^*$;
10 return $t$;
```

Lemma 11 (Correctness of Algorithm 1)**.** *Algorithm 1 always converges to a solution.*

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 ∂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 ∂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 ∂P where the iteration will stop and return $\partial P \cap r$. □

3.3 Approximate Polytope Boundary Oracle

As the focus of this work is on practical high dimensional solutions, it becomes apparent that we need to offer some room for flexibility in the answer of the boundary oracle, since exact nearest neighbor queries are very expensive.

To that end we define an approximate version of the polytope boundary problem.

Definition 12 (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 $d(r^*, \partial P \cap r) \leq \epsilon \cdot diam(P)$.

The first step towards a solution to the approximate polytope boundary problem would be to build an ANN data structure on the pointset S from Lemma 7 and substitute that as the membership data structure in algorithm 1. However that is not enough, since with only that change the algorithm could be fooled and get stuck in a "local optima". To see that, imagine that the algorithm has reached the point t_2, in Figure 3.1, whose nearest neighbor is p_2, but the ANN data structure returns p_3 as a valid approximate nearest neighbor. Then algorithm 1 would keep finding the same point again and again (and possibly never terminate).

For the approximate version we make two additional changes to the algorithm. 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 ϵ-step from t_i towards the ray's apex.

The second change concerns the termination criterion of the algorithm. 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 2: Approximate boundary oracle, using approximate membership

```
Input: H-polytope $P \subset \mathbb{R}^d$, ray $r$ (pair $(s, v)$), $\epsilon$
Output: $t \in \mathbb{R}^d$ s.t. $t \in r$ and $d(t, \partial P) \leq \epsilon diam(P)$
A_MEM = approximate membership oracle for P;
Q = bounding_box(P);
$t = Q \cap r$;
do
    $p_i \leftarrow A\_MEM(t)$;
    if $p_i == p^*$ then
        return $t + \frac{v}{||v||_2}\epsilon$;
    end
    $t_{prev} \leftarrow t$;
    $H \leftarrow H_i$ (facet corresponding to $p_i$);
    $t \leftarrow H \cap r$;
    if $||t - s||_2 \geq ||t_{prev} - s||_2$ then
        $t \leftarrow (t_{prev} - s) - \frac{v}{||v||_2}\epsilon$;
    end
    if $(t - s) \cdot v < 0$ then
        return $s + \frac{v}{||v||_2}\epsilon$;
    end
while True;
```

Convergence

Lemma 13 (Correctness of Algorithm 2)**.** *Algorithm 2 always converges to a solution for the approximate boundary problem.*

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 ϵ. 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 ϵ-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 ∂P this will happen while $(t_i - s) \cdot v > 0$. In this case we return the point $t_i + \frac{v}{||v||_2}\epsilon$ which of course lies within distance $\epsilon \cdot diam(P)$ from ∂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 a point $t_i + \frac{v}{||v||_2}\epsilon$. Alternatively, it will take an ϵ-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 ∂P and in which case we return the point $s + \frac{v}{||v||_2}\epsilon$ which lies in r and is at distance $< \epsilon \cdot diam(P)$ from ∂P.

Therefore, eventually, the algorithm will return a point t, such that $t \in P$ and $d(t, \partial P) \geq \epsilon \cdot diam(P)$. □

The analysis of the proof actually guarantees that the returned point will always lie inside P.

Chapter 4

Experiments

This section examines the practical aspects of the examined oracles, in particular, it introduces our implementation and discusses the experiments.

4.1 Experimental setup

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.

Implementation. All of our code is linked to the software of [EF14]. It is written in C++11 based on using the CGAL[1] library for the readily available data structures of d-dimensional objects, Eigen3[2] for some linear algebra computations and FALCONN[And+15] 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 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

[1]http://www.cgal.org/

[2]http://eigen.tuxfamily.org/index.php?title=Main_Page

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[Lv+07]. The data structure is built for every computed point besides the internal one. Then, assuming that for a query point q the FALCONN data structures returns an approximate nearest neighbor guess x_i; we compare $d(x_i, q)$ to $d(p^*, q)$ and return the point closet to q out of x_i, p^*. The parameters for FALCONN were selected in a manual approach, while trying to maintain a 90% success rate for the approximate boundary oracle.

Evaluation protocol For both the membership and boundary oracle we report preprocessing 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 4.1 depicts the total time in seconds for creating the approximate membership oracle on the random polytopes for different values of d and n.

The two following figures 4.1 and 4.2 depict the total time in seconds taken for all the queries to be completed. Parameters in all cases were tuned such that the corresponding oracles had a success rate of over 90%. Furthermore, the boundary oracle in all cases took on average at most 4 steps in order to converge to a solution. The results matched our expectations with regards to the behaviour of the oracles in the high dimensional case, where we can see a huge difference in the query time, especially as the number of facets grows larger as well.

		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

TABLE 4.1: Preprocessing time in seconds for creating the membership oracle. This includes computing the $n + 1$ pointset and creating the ANN data structure on top of it.

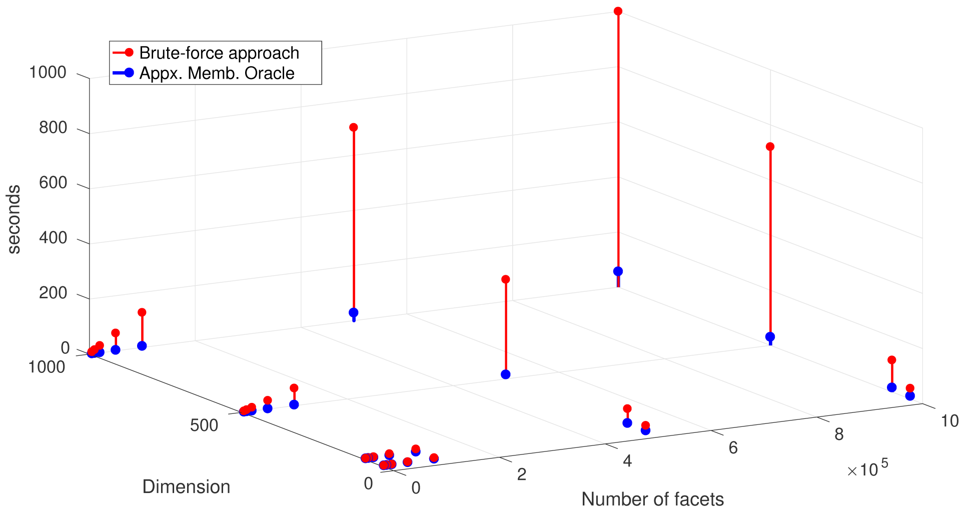

FIGURE 4.1: Average timing results for 2000 queries for varying n and d. Half of the queries were inside the random polytopes and half were outside.

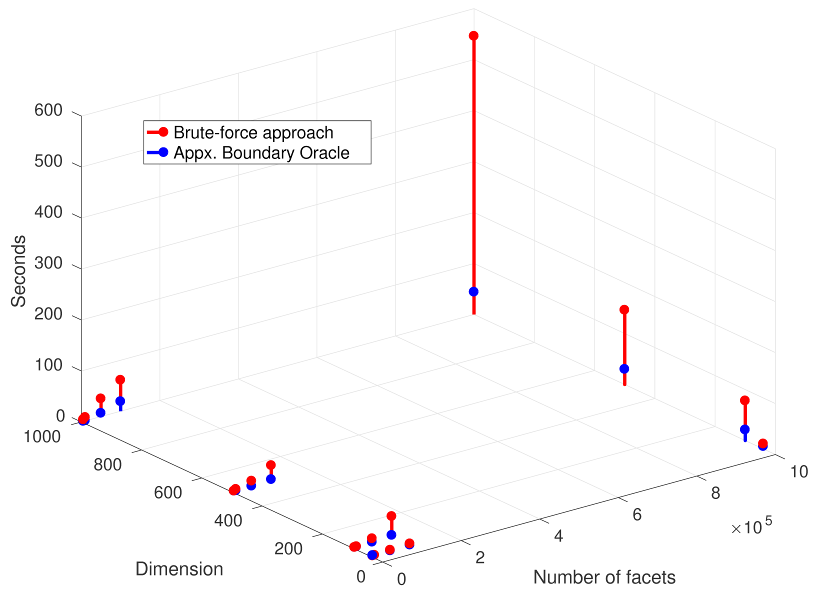

FIGURE 4.2: Average timing results for 1000 ray queries for varying n and d. The approximate boundary oracle took on average at most 4 steps.

Chapter 5

Conclusion and future work

In this work we have presented two practical solutions for the H-polytope membership and boundary problems in the high dimensional setting. Our approaches, as demonstrated in the experiments chapter, scale well as the dimension and the number of facets grow larger.

However, we have left some open question which we are looking forward in answering in the future. For the membership oracle it would be nice to see how the choice of the internal point affects the ϵ' of the ANN data structure. Our intuition tells us that the choice of the Chebyshev center as the choice for the internal point is the optimal one as it minimizes the maximum distance to any facet.

For the boundary oracle we would like to study its behaviour more and try to give a convergence rate. The experiments demonstrate that it adopts well to the boundary of the polytope and can converge very fast. Experiments on both fat and skinny H-polytopes could give a better insight into its behaviour as well.

Lastly, the holy grail of our efforts is to manage to incorporate the high dimensional version of the boundary oracle in sampling approaches for H-polytopes. Achieving that would provide a huge boost in current software computing approximate volume for polytopes. This involves either making the boundary oracle more robust to ANN fails, or adopting sampling methods, like hit-and-run, to boundary oracles with a success rate.

Bibliography

[AEP15] E. Anagnostopoulos, I.Z. Emiris, and I. Psarros. "Low-Quality Dimension Reduction and High-Dimensional Approximate Nearest Neighbor". In: *31st International Symposium on Computational Geometry (SoCG 2015)*. 2015.

[AFM11] S. Arya, G. Dias da Fonseca, and D.M. Mount. "Approximate polytope membership queries". In: *Proc. 43rd ACM Symp. Theory of Computing, STOC 2011, San Jose, USA*. June 2011.

[AFM12a] S. Arya, G. Dias da Fonseca, and D.M. Mount. "Optimal Area-Sensitive Bounds for Polytope Approximation". In: *Proc. Symp. on Computational Geometry*. 2012.

[AFM12b] S. Arya, G. Dias da Fonseca, and D.M. Mount. "Polytope approximation and the Mahler volume". In: *Proc. ACM/SIAM Symp. Discr. Algorithms (SODA)*. 2012.

[AFM17] S. Arya, G. Dias da Fonseca, and D.M. Mount. "Optimal Approximate Polytope Membership". In: *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. 2017.

[AHPV05] P.K. Agarwal, S. Har-Peled, and K.R. Varadarajan. "Geometric approximation via coresets". In: *COMBINATORIAL AND COMPUTATIONAL GEOMETRY, MSRI*. 2005.

[And+15] A. Andoni et al. "Practical and Optimal LSH for Angular Distance". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems*. 2015.

[AR15] A. Andoni and I. Razenshteyn. "Optimal Data-Dependent Hashing for Approximate Near Neighbors". In: *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*. 2015.

[Ary+98] S. Arya et al. "An Optimal Algorithm for Approximate Nearest Neighbor Searching Fixed Dimensions". In: *J. ACM* (1998).

[Aur87] F. Aurenhammer. "Power Diagrams: Properties, Algorithms and Applications". In: *SIAM J. Comput.* (1987).

[BPF82] J.L. Bentley, F.P. Preparata, and M.G Faust. "Approximation Algorithms for Convex Hulls". In: *Commun. ACM* (1982).

[DFK91] M. Dyer, A. Frieze, and R. Kannan. “A random polynomial-time algorithm for approximating the volume of convex bodies”. In: *J. ACM* (1991).

[Dud74] R.M Dudley. “Metric entropy of some classes of sets with differentiable boundaries”. In: *Journal of Approximation Theory* (1974).

[EF14] I.Z. Emiris and V. Fisikopoulos. “Efficient Random-Walk Methods for Approximating Polytope Volume”. In: *30th Annual Symp. Computational Geometry, SOCG’14, Kyoto, Japan, June 08 - 11, 2014*. 2014.

[IM98] P. Indyk and R. Motwani. “Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality”. In: *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*. 1998.

[Lv+07] Q. Lv et al. “Multi-probe LSH: Efficient Indexing for High-dimensional Similarity Search”. In: *Proceedings of the 33rd International Conference on Very Large Data Bases*. 2007.

[LV06] L. Lovász and S. Vempala. “Simulated annealing in convex bodies and an $O^*(n^4)$ volume algorithm”. In: *J. Comp. Syst. Sci.* (2006).

[Ram99] E.A. Ramos. “On Range Reporting, Ray Shooting and K-level Construction”. In: *Proc. Symp. on Computational Geometry*. 1999.

[ZY13] Y. Zheng and K. Yamane. “Ray-Shooting Algorithms for Robotics”. In: *IEEE Trans. Automation Science & Engineering* (2013).