Asymmetric Convex Intersection Testing

Luis Barba[*] *Wolfgang Mulzer*[†]

Abstract

We consider *asymmetric convex intersection testing* (ACIT).

Let $P \subset \mathbb{R}^d$ be a set of n points and $\mathcal{H}$ a set of n halfspaces in d dimensions. We denote by $\textsc{ch}(P)$ the polytope obtained by taking the convex hull of P, and by $\textsc{fh}(\mathcal{H})$ the polytope obtained by taking the intersection of the halfspaces in $\mathcal{H}$. Our goal is to decide whether the intersection of $\mathcal{H}$ and the convex hull of P are disjoint. Even though ACIT is a natural variant of classic LP-type problems that have been studied at length in the literature, and despite its applications in the analysis of high-dimensional data sets, it appears that the problem has not been studied before.

We discuss how known approaches can be used to attack the ACIT problem, and we provide a very simple strategy that leads to a deterministic algorithm, linear on n and m, whose running time depends reasonably on the dimension d.

1 Introduction

Let $d \in \mathbb{N}$ be a fixed constant. Convex polytopes in dimension d can be implicitly represented in two ways, either by its set of vertices, or by the set of halfspaces whose intersection defines the polytope. A polytope represented by its vertices is usually called a *V-polytope*, while a polytope represented by a set of halfspaces is known as an *H-polytope*. Note that the actual complexity of the polytopes can be much larger than the size of their representations [20, Theorem 5.4.5]. In this paper, we study the problem of testing the intersection of convex polytopes with different implicit representations. When both polytopes have the same representation, testing for their intersection reduces to linear programming. However, when there is a mismatch in the representation, the problem changes in nature and becomes more challenging.

To formalize our problem, let $P \subset \mathbb{R}^d$ be a set of n points in $\mathbb{R}^d$, and let $\mathcal{H}$ be a set of n halfspaces in $\mathbb{R}^d$.[1] Just as P implicitly defines the polytope $\textsc{ch}(P)$ obtained by taking the convex hull of P, the set $\mathcal{H}$ implicitly defines the polytope $\textsc{fh}(\mathcal{H})$ obtained by taking the intersection of the halfspaces in $\mathcal{H}$. In the *asymmetric convex intersection problem* (ACIT), our goal is to decide whether the intersection of $\mathcal{H}$ and the convex hull of P are disjoint.

We may assume that $\textsc{fh}(\mathcal{H})$ is nonempty. Otherwise, ACIT becomes trivial. If $\textsc{ch}(P)$ and $\textsc{fh}(\mathcal{H})$ intersect, we would like to find a *witness point* in both $\textsc{ch}(P)$ and $\textsc{fh}(\mathcal{H})$; if not, we would like to determine the closest pair between $\textsc{ch}(P)$ and $\textsc{fh}(\mathcal{H})$ and a separating hyperplane.

Even though ACIT seems to be a natural problem that fits well into the existing work on algorithmic aspects of high-dimensional polytopes [1], we are not aware of any prior work on it. While intersection detection of convex polytopes has been a central topic in computational geometry [5, 8, 9, 13, 14, 21], when we deal with an intersection test between a V-polytope and an H-polytope, the problem seems to remain unstudied. Even the seemingly easy case of this problem in dimension $d = 2$ has no trivial solution running in linear time.

[*]Department of Computer Science, ETH Zürich, Switzerland, luis.barba@inf.ethz.ch

[†]Institut für Informatik, Freie Universität Berlin, mulzer@inf.fu-berlin.de. Supported in part by ERC StG 757609.

[1] We will assume that both P and $\mathcal{H}$ are in *general position* (the exact meaning of this will be made clear later)

The lack of a solution for ACIT may be even more surprising considering that ACIT can be used in the analysis of high-dimensional data: given a high-dimensional data set, represented as a point cloud P, it is natural to represent the *interpolation* of the data as the convex hull $\text{CH}(P)$. Then, we would like to know whether the interpolated data set contains an item that satisfies certain *properties*. These properties are usually represented as linear constraints that must be satisfied, i.e., the data point must belong to the intersection of a set of halfspaces. Then, a witness point corresponds to an interpolated data point with the desired properties, and a separating hyperplane may indicate which properties cannot be fulfilled by the data at hand.

Even though ACIT has not been addressed before, several approaches for related problems[2] may be used to attack the problem. The range of techniques goes from simple brute-force, over classic linear programming [10], the theory of LP-type problems [7, 24] (also in implicit form [2]), to parametric search [19]. In Section 2, we will examine these in more detail and discuss their merits and drawbacks. Briefly, several of these approaches can be applied to ACIT. However, as we will see, it seems hard to get an algorithm that is genuinely simple and at the same time achieves linear (or almost linear) running time in the number of points and halfspaces, with a reasonable dependency on the dimension d.

Thus, in Section 4, we present a simple recursive primal-dual pruning strategy that leads to a deterministic linear time algorithm with a dependence on d that is comparable to the best bounds for linear programming. Even though the algorithm itself is simple and can be presented in a few lines, the analysis requires us to take a close look at the polarity transformation and how it interacts with two disjoint polytopes (Section 3). Its analysis is also non-trivial and its correctness spans over the entire Section 4.3. We believe in the development of simple and efficient methods. The analysis can be complicated, but the algorithm must remain simple. The simpler the algorithm, the more likely it is to be eventually implemented.

2 How to solve ACIT with existing tools

The first thing that might come to mind to solve ACIT is to cast it as a linear program. This is indeed possible, however the resulting linear program consists of n variables and $O(n+m)$ constraints. We want to find a point x subject to being inside all halfspaces in $\mathcal{H}$, and being a convex combination of all points in P. That is, we want $x = \sum_{p\in P} \alpha_p\, p$, where $\sum_{p\in P} \alpha_p = 1$, and $\alpha_p \geq 0$, for all $p \in P$. Moreover, we want that $x \in H$, for all $H \in \mathcal{H}$, which can be expressed as m linear inequalities by looking at the scalar product of x and the normal vectors of the bounding hyperplanes of the halfspaces. Because the best combinatorial algorithms for linear programming provide poor running times when both the number of variables and constraints are large, this approach is far from efficient unless n is really small.[3]

Another trivial way to solve ACIT, the *brute force* algorithm, is to compute all facets of $\text{CH}(P)$. That is, we can compute $\text{CH}(P)$ explicitly to obtain a set $\mathcal{H}_P$ of the $O(n^{\lfloor d/2 \rfloor})$ halfspaces with $\text{CH}(P) = \text{FH}(\mathcal{H}_P)$ [6, 12]. With this representation, we can test if $\text{FH}(\mathcal{H}_P)$ and $\text{FH}(\mathcal{H})$ intersect using a general linear program with d variables, or compute the distance between $\text{FH}(\mathcal{H}_P)$ and $\text{FH}(\mathcal{H})$ using either an LP-type algorithm (see below), or algorithms for convex quadratic programming [16, 17]. The running time is again quite bad for larger values of n and d, since the size of $\mathcal{H}_P$ might be as high as $\Theta(n^{\lfloor d/2 \rfloor})$ [20, Theorem 5.4.5].

A more clever approach is to use the LP-type framework directly, as described below.

The LP-type Framework

The classic *LP-type framework* that was introduced by Sharir and Welzl [24] in order to extend the notion of low-dimensional linear programming to a wider range of problems. An LP-type problem $(\mathcal{C}, w)$ consists of a set $\mathcal{C}$ of k *constraints* and a *weight function* $w : 2^{\mathcal{C}} \to \mathbb{R}$ that assigns a real-valued weight $w(C)$ to each set

[2] In particular, checking for the intersection of the convex hulls of to d-dimensional point sets

[3] In fact, in the traditional computational model of computational geometry, the Real RAM [22], we cannot solve general linear programs in polynomial time, since the best known algorithms (e.g., ellipsoid, interior point methods) are only *weakly* polynomial with a running time that depends on the bit complexity of the input.

$C \subseteq \mathcal{C}$ of constraints.[4] The weight function must satisfy the following three axioms:

- **Monotonicity**: For any set $C \subseteq \mathcal{C}$ of constraints and any $c \in \mathcal{C}$, we have $w\big(C \cup \{c\}\big) \leq w(C)$.
- **Existence of a Basis**: There is a constant $\tilde{d} \in \mathbb{N}$ such that for any $C \subseteq \mathcal{C}$, there is a subset $B \subseteq C$ with $|B| \leq \tilde{d}$ and $w(B) = w(C)$.
- **Locality**: For any $B \subseteq C \subseteq \mathcal{C}$ with $w(B) = w(C)$ and for any $c \in C$, we have that if $w\big(C \cup \{c\}\big) < w(C)$, then also $w\big(B \cup \{c\}\big) < w(B)$.

For $C \subseteq \mathcal{C}$, an inclusion-minimal subset $B \subseteq C$ with $w(B) = w(C)$ is called a *basis* for C. Solving an LP-type problem $(\mathcal{C}, w)$ amounts to computing a basis for $\mathcal{C}$. Many algorithms have been developed for this extension of linear programming, provided that base cases with a constant number of constraints can be solved in $O(1)$ time. Seidel proposed a simple randomized algorithm with expected $O(\tilde{d}!k)$ running time [23]. From there, several algorithms have been introduced improving the dependency on $\tilde{d}$ in the running time [3, 11, 23, 24]. The best known randomized algorithm solves LP-type problems in $O(\tilde{d}^2 k + 2^{O(\sqrt{\tilde{d}\log\tilde{d}})})$ time, while the best deterministic algorithms have still a running time of the form $O(\tilde{d}^{O(\tilde{d})}k)$. We would like to obtain an algorithm with a similar running time for ACIT.

ACIT as an LP-type problem

To use these existing machinery, one can try to cast ACIT as an LP-type problem. To this end, we fix $\mathcal{H}$, and define an LP-type problem (P, w) as follows. The *constraints* are modeled by the points in P. The weight function $w : 2^P \to \mathbb{R}$ is defined as $w(Q) = d\big(\textsc{ch}(Q), \textsc{fh}(\mathcal{H})\big)$, for any $Q \subseteq P$, where $d(\cdot, \cdot)$ is the smallest Euclidean distance between any pair of points from the two polytopes. It is a pleasant exercise to show that this indeed defines an LP-type problem of combinatorial dimension d.

Thus, the elegant methods to solve LP-type problems mentioned above become applicable. Unfortunately, this does not give an efficient algorithm. This is because the set $\mathcal{H}$ remains fixed throughout, making unfeasible to solve the base cases consisting of $O(1)$ constraints of P in constant time.

A randomized algorithm for ACIT

As an extension of the LP-type framework, Chan [2] introduced a new technique that allows us to deal with certain LP-type problems where the constraints are too numerous to write explicitly, and are instead specified "implicitly". More precisely, as mentioned above, ACIT can be seen as a linear program, with m constraints coming from $\mathcal{H}$, and $O(n^{\lfloor d/2 \rfloor})$ constraints coming from all the facets of $\textsc{ch}(P)$. The latter are implicitly defined by P using only n points. Thus an algorithm capable of solving implicitly defined linear programs would provide a solution for ACIT. The technique developed by Chan achieves this by using two main ingredients: a decision algorithm, and a partition of the problem into subproblems of smaller size whose recursive solution can be combined to produce the global solution of the problem. Using the power of randomization and geometric cuttings, this technique leads to a complicated algorithm to solve this implicit linear program, and hence ACIT, in expected $O(d^{O(d)}(n+m))$ time [4]. Besides the complexity of this algorithm, the constant hidden by the big O notation resulting from using this technique seems prohibitive [18].

In hope of obtaining a deterministic algorithm for ACIT, one can turn to multidimensional parametric search [19] to try de-randomizing the above algorithm. However, even if all the requirements of this technique can be sorted out, it would lead to a highly complicated algorithm and polylogarithmic overhead.

In the following sections, we present the first deterministic solution for ACIT using a simple algorithm that overcomes the difficulties mentioned above. We achieve this solution by diving into the intrinsic duality of the problem provided by the polar transformation, while exploiting the LP-type-like structure of our problem. The resulting algorithm is quite simple, and a randomized version of it could be written with a few lines of code, provided that one has some LP solver at hand.

[4] Actually, we can allow weights from an arbitrary totally ordered set, but for our purposes, real weights will suffice.

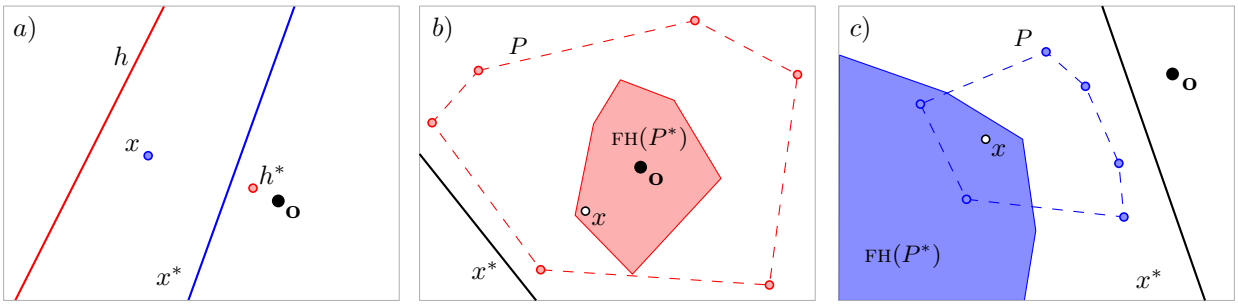

Fig. 1: (a) The situation described in Lemma 3.1. (b) A valid set P of points that is embracing and its polar P^* that is also embracing. (c) A set P that is avoiding and its polar P^* that is also avoiding.

3 Geometric Preliminaries

Let $\mathbf{o}$ denote the *origin* of $\mathbb{R}^d$. A *hyperplane* h is a $(d-1)$-dimensional affine space in $\mathbb{R}^d$ of the form

$$h = \{x \in \mathbb{R}^d \mid \langle z, x \rangle = 1\},$$

for some $z \in \mathbb{R}^d \setminus \{\mathbf{o}\}$, where $\langle \cdot, \cdot \rangle$ represents the scalar product in $\mathbb{R}^d$. We exclude hyperplanes that pass through the origin. A (closed) *halfspace* is the closure of the point set on either side of a given hyperplane, i.e., a halfspace contains the hyperplane defining its boundary.

3.1 The Polar Transformation

Given a point $p \in \mathbb{R}^d$, we define the *polar* of p to be the hyperplane

$$p^* = \{x \in \mathbb{R}^d \mid \langle p, x \rangle = 1\}.$$

Given a hyperplane h in $\mathbb{R}^d$, we define its *polar* $h^* \in \mathbb{R}^d$ as the point with

$$h = \{x \in \mathbb{R}^d \mid \langle x, h^* \rangle = 1\}.$$

Let $\rho_{\mathbf{o}}(p) = \{x \in \mathbb{R}^d \mid \langle p, x \rangle \leq 1\}$ and $\rho_{\infty}(p) = \{x \in \mathbb{R}^d \mid \langle p, x \rangle \geq 1\}$ be the two halfspaces supported by p^* such that $\mathbf{o} \in \rho_{\mathbf{o}}(p)$ and $\mathbf{o} \notin \rho_{\infty}(p)$. Similarly, $h_{\mathbf{o}}$ and h_{∞} denote the halfspaces supported by h such that $\mathbf{o} \in h_{\mathbf{o}}$ and $\mathbf{o} \notin h_{\infty}$.

Note that the polar of a point $p \in \mathbb{R}^d$ is a hyperplane whose polar is equal to p, i.e., the polar operation is involutory (for more details, see Section 2.3 in Ziegler's book [25]). The following result is illustrated in Figure 1(a), for $d = 2$.

Lemma 3.1 (Lemma 2.1 of [1])**.** *Let p and h be a point and a hyperplane in $\mathbb{R}^d$, respectively. Then, $p \in h_{\mathbf{o}}$ if and only if $h^* \in \rho_{\mathbf{o}}(p)$. Also, $p \in h_{\infty}$ if and only if $h^* \in \rho_{\infty}(p)$. Finally, $p \in h$ if and only if $h^* \in p^*$.*

Let P be a set of points in $\mathbb{R}^d$. We say that P is *embracing* if $\mathbf{o}$ lies in the interior of $\text{CH}(P)$. We say that P is *avoiding* if $\mathbf{o}$ lies in the complement of $\text{CH}(P)$. Note that we do not consider point sets whose convex hull has $\mathbf{o}$ on its boundary. We say that P is *valid* if it is either embracing or avoiding.

Let $\mathcal{H}$ be a set of halfspaces in $\mathbb{R}^d$ such that $\text{FH}(\mathcal{H}) \neq \emptyset$, and the boundary of no halfspace in $\mathcal{H}$ contains $\mathbf{o}$. We say that $\mathcal{H}$ is *embracing* if $\mathbf{o} \in H$ for all $H \in \mathcal{H}$ (i.e., $\mathbf{o} \in \text{FH}(\mathcal{H})$). We say that $\mathcal{H}$ is *avoiding* if none of its halfspaces contains $\mathbf{o}$, i.e., $\mathbf{o} \notin \bigcup_{H \in \mathcal{H}} H$. We say that $\mathcal{H}$ is *valid* if it is either embracing or avoiding.

We now describe how to polarize convex polytopes defined as convex hulls of valid sets of points or as intersections of valid sets of halfspaces. Let $\mathcal{H}$ be a valid set of halfspaces in $\mathbb{R}^d$. To *polarize* $\mathcal{H}$, consider the set of hyperplanes bounding the halfspaces in $\mathcal{H}$, and let $\mathcal{H}^*$ be the set consisting of all the points being the polars of these hyperplanes.

Lemma 3.2. *Let $\mathcal{H}$ be a valid set of halfspaces in $\mathbb{R}^d$. Then, $\mathcal{H}^*$ is embracing if and only if $\mathcal{H}$ is embracing.*

Proof. Recall that $\mathcal{H}$ is embracing if and only if $\textsc{fh}(\mathcal{H})$ is bounded and contains $\mathbf{o}$.

$\Rightarrow$). Assume that $\mathcal{H}^*$ is embracing. Thus, $\mathbf{o} \in \textsc{ch}(\mathcal{H}^*)$. In this case, there is a subset Q of $d+1$ points of $\mathcal{H}^*$ whose convex hull contains $\mathbf{o}$, by Carathéodory's theorem [20, Theorem 1.2.3]. Consider all halfspaces of $\mathcal{H}$ whose boundary polarizes to a point in Q. If none of these halfspaces contains the origin, then their intersection has to be empty. This is not allowed by the validity of $\mathcal{H}$. Thus, as $\mathcal{H}$ is valid, and as $\mathcal{H}$ cannot avoid the origin, we conclude that $\mathcal{H}$ is embracing.

$\Leftarrow$). For the other direction, assume that $\mathbf{o} \notin \textsc{ch}(\mathcal{H}^*)$. We want to prove that $\mathcal{H}$ is not embracing. For this, let h be a hyperplane that separates $\mathbf{o}$ from $\textsc{ch}(\mathcal{H}^*)$. That is, $\mathcal{H}^* \subset h_\infty$. Lemma 3.1 implies that the segment $\mathbf{o}h^*$ intersects the boundary of each plane in $\mathcal{H}$. Therefore, since the ray shooting from $\mathbf{o}$ in the direction of the vector $-h^*$ intersects no plane bounding a halfspace in $\mathcal{H}$, the polytope $\textsc{fh}(\mathcal{H})$ either does not contain the origin or is not bounded. Consequently, $\mathcal{H}$ is not embracing. □

Let P be a valid set of points in $\mathbb{R}^d$. To *polarize* P, let $\Pi(P)$ be the set of hyperplanes polar to the points of P. We have two natural ways of polarizing P, depending on whether $\mathbf{o}$ lies in the interior of $\textsc{ch}(P)$, or in its complement (recall that $\mathbf{o}$ cannot lie on the boundary of $\textsc{ch}(P)$). If $\mathbf{o} \in \textsc{ch}(P)$, then

$$P^* = \{h_\circ \mid h \in \Pi(P)\}$$

is the *polarization* of P. Otherwise, if $\mathbf{o} \notin \textsc{ch}(P)$, then

$$P^* = \{h_\infty \mid h \in \Pi(P)\}.$$

Lemma 3.3. *Let P be a valid set of points in $\mathbb{R}^d$. Then P^* is valid, i.e., $\textsc{fh}(P^*) \neq \emptyset$ and P^* is either embracing or avoiding.*

Proof. If $\mathbf{o} \notin \textsc{ch}(P)$, then there is a hyperplane h such that $P \subset h_\infty$. Thus, h^* belongs to p^* for every $p \in P$, i.e., $h^* \in \textsc{fh}(P^*)$. Thus, $\textsc{fh}(P^*)$ is nonempty, and none of its halfspaces contains the origin by definition. That is, P^* is avoiding. If $\mathbf{o} \in \textsc{ch}(P)$, then $\mathbf{o} \in \textsc{fh}(P^*)$ by definition. Thus, to show that P^* is embracing, it remains only to show that it is bounded. To this end, assume for a contradiction that $\textsc{fh}(P^*)$ is unbounded. Then, we can take a point $x \in \textsc{fh}(P^*)$ at arbitrarily large distance from $\mathbf{o}$. Thus, x^* is a plane arbitrarily close to $\mathbf{o}$ such that $P \subset x^*$. Therefore, all points of P must lie on a single halfspace that contains $\mathbf{o}$ on its boundary. Because P is valid, we know that $\mathbf{o}$ cannot lie on the boundary of $\textsc{ch}(P)$ and hence, $\mathbf{o} \notin \textsc{ch}(P)$—a contradiction with our assumption that $\mathbf{o} \in \textsc{ch}(P)$. Therefore, $\textsc{fh}(P^*)$ is bounded and hence P^* is embracing. □

Lemma 3.4. *Let $P \subset \mathbb{R}^d$ be a valid finite point set in d dimensions, and let $\mathcal{H}$ be a valid finite set of halfspaces in d dimensions. Then the polar operator is involutory: $P = (P^*)^*$ and $\mathcal{H} = (\mathcal{H}^*)^*$.*

Proof. The equality $P = (P^*)^*$ follows directly from the definition, because the polar operator for points and hyperplanes is involutory. For equality $\mathcal{H} = (\mathcal{H}^*)^*$, we must check that the orientation of the halfspaces is preserved. First, if $\mathcal{H}$ is embracing, i..e, $\mathbf{o} \in \textsc{fh}(\mathcal{H})$, then every $H \in \mathcal{H}$ is of the form $H = h_\circ$, for some $(d-1)$-dimensional hyperplane h. Moreover, Lemma 3.2 implies that $\mathbf{o} \in \textsc{ch}(\mathcal{H}^*)$. Thus, we have $\mathcal{H} = (\mathcal{H}^*)^*$ in this case. Similarly, if $\mathcal{H}$ is avoiding, i.e., $\mathbf{o} \notin \bigcup_{H \in \mathcal{H}} H$, then every $H \in \mathcal{H}$ is of the form $H = h_\infty$ for some $(d-1)$-dimensional hyperplane h, and by Lemma 3.2, we have $\mathbf{o} \notin \textsc{ch}(\mathcal{H}^*)$. Thus, we have again $\mathcal{H} = (\mathcal{H}^*)^*$. □

Corollary 3.5. *Let P be a valid set of points in $\mathbb{R}^d$. Then, P^* is embracing if and only if P is embracing. Moreover, P^* is avoiding if and only if P is avoiding.*

Proof. Because P is valid, P^* is valid by Lemma 3.3. Therefore, Lemma 3.2 implies that P^* is embracing if and only if $(P^*)^*$ is embracing. Because $P = (P^*)^*$ by Lemma 3.4, we conclude that P^* is embracing if and only if P is embracing. Note that if a valid set P is not embracing, then it is avoiding, yielding the second part of the result. □

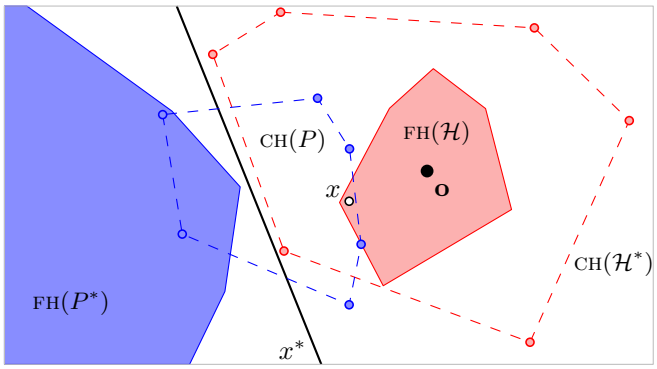

Fig. 2: An example of Theorem 3.6 in dimension 2, where a point x lies in the intersection of $\text{CH}(P)$ and $\text{FH}(\mathcal{H})$ if and only if x^* separates $\text{CH}(\mathcal{H}^*)$ from $\text{FH}(P^*)$.

The following result is illustrated in Figure 2, for $d = 2$.

Theorem 3.6 (Consequence of Theorem 3.1 of [1])**.** *Let P be a finite set of points and let $\mathcal{H}$ be a valid finite set of halfspaces in $\mathbb{R}^d$ such that either* (1) *P is avoiding while $\mathcal{H}$ is embracing, or* (2) *P is embracing while $\mathcal{H}$ is avoiding. Then, a point x lies in the intersection of $\text{CH}(P)$ and $\text{FH}(\mathcal{H})$ if and only if the hyperplane x^* separates $\text{FH}(P^*)$ from $\text{CH}(\mathcal{H}^*)$. Also a hyperplane h separates $\text{CH}(P)$ from $\text{FH}(\mathcal{H})$ if and only if the point h^* lies in the intersection of $\text{FH}(P^*)$ and $\text{CH}(\mathcal{H}^*)$.*

Conditions (1) and (2) will be crucial in our algorithm. Note that by Corollary 3.5, we have that P and $\mathcal{H}$ satisfy condition (1), then the point set $\mathcal{H}^*$ and the set P^* of halfspaces satisfy condition (2), and vice versa.

3.2 Conflict Sets, ε-nets, and Closest Pairs

Let $P \subseteq \mathbb{R}^d$ be a finite point set in d dimensions, and let H be a halfspace in $\mathbb{R}^d$. We say that a point $p \in P$ *conflicts* with H if $p \in H$. The *conflict set* of P and H, denoted $V_H(P)$, consists of all points $p \in P$ that are in conflict with H, i.e., $V_H(P) = P \cap H$. Let $\varepsilon \in (0, 1)$ be a parameter. A set $N \subseteq P$ is called an *ε-net* for P if for every halfspace H in $\mathbb{R}^d$, we have

$$V_H(N) = \emptyset \Rightarrow \big|V_H(P)\big| < \varepsilon|P|. \tag{1}$$

By the classic *ε-net theorem* [15, Theorem 5.28], a random subset $N \subset P$ of size $\Theta\Big(\varepsilon^{-1} \log\left(\varepsilon^{-1} + \alpha^{-1}\right)\Big)$ is an ε-net for P with probability at least $1 - \alpha$. For a deterministic algorithm running in linear time, we can compute such a net using the complicated algorithm of Chazelle and Matoušek [7, Chapter 4.3] or the much simpler algorithm introduced by Chan [3]. See the textbooks of Matoušek [20], Chazelle [7], or Har-Peled [15] for more details on ε-nets and their uses in computational geometry. The following observation shows the usefulness of conflict sets for our problem.

Lemma 3.7. *Let $P \subseteq \mathbb{R}^d$ be a finite point set and $\mathcal{H}$ a finite set of halfspaces in d dimensions. Let $N \subseteq P$ such that $\text{FH}(\mathcal{H})$ and $\text{CH}(N)$ are disjoint, and let x, y be the closest pair between them, such that $x \in \text{FH}(\mathcal{H})$ and $y \in \text{CH}(N)$. Let H_y be the halfspace through y perpendicular to the segment xy, containing $\text{FH}(\mathcal{H})$. Then, we have $d\big(\text{FH}(\mathcal{H}), P\big) < d\big(\text{FH}(\mathcal{H}), N\big)$ if and only if $V_{H_y}(P) \neq \emptyset$.*

Proof. Since all points in $\mathbb{R}^d \setminus H_y$ have distance larger than $d\big(\text{FH}(\mathcal{H}), N\big)$ from $\text{FH}(\mathcal{H})$, the implication $V_{H_y}(P) = \emptyset \Rightarrow d\big(\text{FH}(\mathcal{H}), P\big) < d\big(\text{FH}(\mathcal{H}), N\big)$ is immediate.

Now assume that $V_{H_y}(P) \neq \emptyset$, say, $p \in V_{H_y}(P)$. Then, the line segment py is contained in $\text{CH}(P)$, and since $p \in V_{H_y}(P)$ and since p does not lie on the boundary of H_y be our general position assumption, it follows that the angle between the segments py and xy is strictly smaller than $\pi/2$. Hence, we have

$$d\big(\text{FH}(\mathcal{H}), P\big) \leq d(x, py) < \big(\text{FH}(\mathcal{H}), N\big),$$

as claimed. □

Similarly, let $\mathcal{H}$ be a finite set of halfspaces in d dimensions, and let $p \in \mathbb{R}^d$ be a point. The *conflict set* $V_p(\mathcal{H})$ of $\mathcal{H}$ and p consists of all halfspaces that do not contain p, i.e. $V_p(\mathcal{H}) = \{H \in \mathcal{H} \mid p \notin H\}$. We have the following polar version of Lemma 3.7:

Lemma 3.8. *Let $P \subseteq \mathbb{R}^d$ be a finite point set and $\mathcal{H}$ a finite set of halfspaces in d dimensions. Let $\mathcal{H}' \subseteq \mathcal{H}$ such that $\textsc{fh}(\mathcal{H}')$ and $\textsc{ch}(P)$ are disjoint, and let x, y be the closest pair between them, such that $x \in \textsc{fh}(\mathcal{H}')$ and $y \in \textsc{ch}(P)$. Then, we have $d\big(\textsc{fh}(\mathcal{H}), P\big) > d\big(\textsc{fh}(\mathcal{H}'), P\big)$ if and only if $V_x(\mathcal{H}) \neq \emptyset$.*

Proof. First, if $V_x(\mathcal{H}) = \emptyset$, then $x \in \textsc{fh}(\mathcal{H})$, and since $\textsc{fh}(\mathcal{H}) \subseteq \textsc{fh}(\mathcal{H}')$, we have $d\big(\textsc{fh}(\mathcal{H}), P\big) = d\big(\textsc{fh}(\mathcal{H}'), P\big)$.

Second, suppose that $V_x(\mathcal{H}) \neq \emptyset$, say, $H \in V_x(\mathcal{H})$. Then, $x \notin H$, and since, by gneral position, x is the unique point in $\textsc{fh}(\mathcal{H}')$ with $d(x, \textsc{ch}(P)) = d(\textsc{fh}(\mathcal{H}'), \textsc{ch}(P))$, we have

$$d\big(\textsc{fh}(\mathcal{H}), P\big) \geq d\big(\textsc{fh}(\mathcal{H}' \cup \{H\}), P\big) > d\big(\textsc{fh}(\mathcal{H}'), P\big),$$

as claimed. □

The following lemma gives a polar meaning to the notion of ε-nets.

Lemma 3.9. *Let $N \subseteq P$ be an ε-net for P such that if $\mathbf{o} \in \textsc{ch}(P)$, then also $\mathbf{o} \in \textsc{ch}(N)$. For any point $x \in \mathbb{R}^d$, it holds that if $x \in \textsc{fh}(N^*)$, then $\big|V_x\big(P^*\big)\big| \leq \varepsilon |P|$.*

Proof. First, suppose that $\mathbf{o} \in \textsc{ch}(P)$, then, we have $\mathbf{o} \in \textsc{ch}(N)$, and hence $\mathbf{o} \in \textsc{fh}(N^*)$. Since $x \in \textsc{fh}(N^*)$, we have $x \in \rho_{\mathbf{o}}(p)$, for all $p \in N$. By Lemma 3.1, we get $p \in \rho_{\mathbf{o}}(x)$, for all $p \in N$, so $N \cap \rho_\infty(x) = \emptyset$. Since N is an ε-net for P, we conclude $|P \cap \rho_\infty(x)| \leq \varepsilon |P|$. The claim now follows, because by Lemma 3.1, we have $|P \cap \rho_\infty(x)| = |V_x\big(P^*\big)|$.

Second, suppose that $\mathbf{o} \notin \textsc{ch}(P)$, then, we also get $\mathbf{o} \notin \textsc{ch}(N)$, and hence $\mathbf{o} \notin \bigcup_{H \in N^*} H$. Since $x \in \textsc{fh}(N^*)$, we have $x \in \rho_\infty(p)$, for all $p \in N$. By Lemma 3.1, we get $p \in \rho_\infty(x)$, for all $p \in N$, so $N \cap \rho_{\mathbf{o}}(x) = \emptyset$. Since N is an ε-net for P, we conclude $|P \cap \rho_{\mathbf{o}}(x)| \leq \varepsilon |P|$. The claim now follows, because by Lemma 3.1, we have $|P \cap \rho_{\mathbf{o}}(x)| = |V_x\big(P^*\big)|$. □

4 A Simple Algorithm

Let P be a valid set of n points and let $\mathcal{H}$ be a valid set of m halfspaces in $\mathbb{R}^d$ such that either (1) P is avoiding while $\mathcal{H}$ is embracing, or (2) P is embracing while $\mathcal{H}$ is avoiding. We first present a slightly more restrictive algorithm that requires conditions (1) or (2) to hold. We spend the next few sections proving its correctness and running time, and then we extend it to a general algorithm for the ACIT problem.

4.1 Description of the Algorithm

Our algorithm $\textsc{Test}(P, \mathcal{H})$ takes P and $\mathcal{H}$ as input, such that either (1) or (2) is satisfied, and it computes either the closest pair between $\textsc{ch}(P)$ and $\textsc{fh}(\mathcal{H})$, if $\textsc{ch}(P)$ and $\textsc{fh}(\mathcal{H})$ are disjoint, or the closest pair between $\textsc{ch}(H^*)$ and $\textsc{fh}(P^*)$, if $\textsc{ch}(P)$ and $\textsc{fh}(\mathcal{H})$ intersect. By Theorem 3.6, this is always possible.

The algorithm is recursive. Let $\alpha = c\, d^4 \log d$, for some approriate constant $c > 0$. For the base case, if both $|P|, |\mathcal{H}| \leq \alpha$, we apply the brute force algorithm: we explicity compute the polytope $\textsc{ch}(P)$ to obtain the set $\mathcal{H}_P$ of hyperplanes with $\textsc{ch}(P) = \textsc{fh}(\mathcal{H}_P)$, and we use a classic LP-type algorithm to find the closest pair between $\textsc{ch}(P)$ and $\textsc{fh}(\mathcal{H})$ or between $\textsc{fh}(P^*)$ and $\textsc{ch}(\mathcal{H}^*)$. Otherwise, we compute a $(1/d^4)$-net $N \subseteq P$, and if necessary, we add $d+1$ points to N such that if $\mathbf{o} \in \textsc{ch}(P)$, then $\mathbf{o} \in \textsc{ch}(N)$. These $d+1$ points can be found in $O(n)$ time using basic linear algebra. Then, we execute the following loop.

Repeat $2d+1$ **times:** Recursively call $\textsc{Test}(\mathcal{H}^*, N^*)$; there are two possibilities.

Case 1: $\textsc{ch}(\mathcal{H}^*)$ and $\textsc{fh}(N^*)$ are disjoint. Then, $\textsc{Test}(\mathcal{H}^*, N^*)$ returns the closest pair x, y, with $x \in \textsc{ch}(\mathcal{H}^*)$ and $y \in \textsc{fh}(N^*)$ (unique by our general position assumption). Let $V_y \subset P^*$ be conflict set of P^* and y. If

$V_y = \emptyset$, then report that $\text{CH}(P)$ and $\text{FH}(\mathcal{H})$ intersect, and output x, y as the polar witness. Otherwise, add to N all elements of ${V_y}^*$, and continue with the next iteration.
Case 2: $\text{CH}(\mathcal{H}^*)$ and $\text{FH}(N^*)$ intersect. Then, $\textsc{Test}(\mathcal{H}^*, N^*)$ returns the closest pair x, y between $\text{FH}((\mathcal{H}^*)^*) = \text{FH}(\mathcal{H})$ and $\text{CH}((N^*)^*) = \text{CH}(N)$, with $x \in \text{FH}(\mathcal{H})$ and $y \in \text{CH}(N)$. Let H be the halfspace that contains $\text{FH}(\mathcal{H})$ supported by the normal hyperplane of xy through y. Let V_H be the conflict set of P and H. If $V_H = \emptyset$, then report that $\text{CH}(P)$ and $\text{FH}(\mathcal{H})$ are disjoint, and output x, y as the witness. Otherwise, add to N all elements of V_H and continue with the next iteration.

If the loop terminates without a result, the algorithm finishes and returns an Error.

4.2 Running Time

While at this point we have no idea why $\textsc{Test}(P, \mathcal{H})$ works, we can start by analyzing its running time. In the base case, when both P and $\mathcal{H}$ have of at most α elements, we can compute $\text{CH}(P)$ explicitly to obtain the $O\Big(\alpha^{\lfloor d/2 \rfloor}\Big)$ halfspaces of $\mathcal{H}_P$. We can do this in a brute force manner by trying all d-tuples of P^d and checking whether all of P is on the same side of the hyperplane spanned by a given tuple, or we can run a convex-hull algorithm [6, 12]. The former approach has a running time $O\big(\alpha^{d+1}\big)$, while the latter needs $O\big(\alpha^{\lfloor d/2 \rfloor}\big)$ time [6, 12]. Once we have $\mathcal{H}_P$ at hand, we can run standard LP-type algorithms with $O(\alpha^{\lfloor d/2 \rfloor})$ constraints to determine the closest pair either between $\text{CH}(P)$ and $\text{FH}(\mathcal{H})$, or between $\text{CH}(\mathcal{H}^*)$ and $\text{FH}(P^*)$. The running time of such algorithms is $O(d^{O(d)}\alpha^{\lfloor d/2 \rfloor}) = O(d^{O(d)})$ [3].

To see what happens in the main loop of the algorithm, we apply the theory of ε-nets, as described in the Section 3. As mentioned there, the initial set N is a $(1/d^4)$-net for P. Thus, the size of each V_H added to N in Case 2 of the main loop of our algorithm is at most n/d^4. Using Lemma 3.9, the same holds for any set V_y added in Case 1. Thus, regardless of the case, the size of N at the beginning of the i-th loop iteration is at most $\max\{in/d^4, \alpha\}$.

The main loop runs for at most $2d+1$ iterations. Thus, the size of N never exceeds $(2d+1)n/d^4 \leq \beta n/d^3$, for some constant $\beta > 0$. Since the algorithm to compute the $(1/d^4)$-net N for P runs in time $O(d^{O(d)}n)$ [3,7], we obtain the following recurrence for the running time:

$$T(n,m) \leq \begin{cases} \text{LP}\Big(\alpha + \alpha^{\lfloor d/2 \rfloor}, d\Big) + O\big(\alpha^{\lfloor d/2 \rfloor}\big), & \text{if } n, m \leq \alpha, \\ (2d+1) \cdot T\Big(m, \max\big\{\beta n/d^3, \alpha\big\}\Big) + O(d^{O(d)}n), & \text{otherwise.} \end{cases}$$

We look further into $T\big(m, \max\big\{\beta n/d^3, \alpha\big\}\big)$ and notice that if we do not reach the base case, then unfolding the recursion by one more step yields

$$T\big(m, \max\{\beta n/d^3, \alpha\}\big) \leq (2d+1) \cdot T\big(\max\{\beta n/d^3, \alpha\}, \max\{\beta m/d^3, \alpha\}\big) + O(d^{O(d)}m).$$

Thus, by contracting two steps into one, we get the following more symmetric relation:

$$T(n,m) \leq (2d+1)^2 \cdot T\big(\max\{\beta n/d^3, \alpha\}, \max\{\beta m/d^3, \alpha\}\big) + O\big(d^{O(d)}(n+m)\big),$$

for sufficiently large n and m. Together with the base case, one can show by induction that this yields a running time of $O(d^{O(d)}(n+m))$.

Remark. Because the best deterministic algorithm know for LP-type problems with n constraints runs also in time $O(d^{O(d)}n)$ [3], substantial improvements on the running time of our problem seem out of reach. If we allow randomization however, then we can improve in two places. First of all, by randomly sampling $O(\alpha \log n)$ elements of P, we obtain a $(1/d^4)$-net of P with high probability. Secondly, the base case could be solved with faster algorithms. The best known randomized algorithms for LP-type problems with n constraints have a running time of $O(d^2 n + 2^{O(\sqrt{d \log d})})$, which substantially improves the dependency on d. Alternatively, we could use methods to solve convex quadratic programs in the base case to find the closest pair between two H-polytopes.

4.3 Correctness

We show that $\textsc{Test}(P, \mathcal{H})$ indeed tests whether $\textsc{ch}(P)$ and $\textsc{fh}(\mathcal{H})$ intersect. First, we verify that the input to each recursive call $\textsc{Test}(\cdot, \cdot)$ satisfies either condition (1) or (2).

Lemma 4.1. *Let $P \subset \mathbb{R}^d$ be a finite point set and $\mathcal{H}$ a finite set of halfspaces in d dimensions, such that either (1) P is avoiding while $\mathcal{H}$ is embracing, or (2) P is embracing while $\mathcal{H}$ is avoiding. Consider a call of $\textsc{Test}(P, \mathcal{H})$. Then, the input to each recursive call satisfies either condition (1) or condition (2).*

Proof. We do induction on the recursion depth. The base case holds by assumption. For the inductive step, we note that if the input $(P, \mathcal{H})$ satisfies condition (1), then $(N, \mathcal{H})$ also satisfies condition (1), for any subset $N \subseteq P$. For condition (2), first note that our algorithm ensures that if $\mathbf{o} \in \textsc{ch}(P)$, then also $\mathbf{o} \in \textsc{ch}(N)$. This implies that if $(P, \mathcal{H})$ satisfies condition (2), then $(N, \mathcal{H})$ also satisfies condition (2). Finally, if $(P, \mathcal{H})$ satisfies condition (1), then by Corollary 3.5 $\left(\mathcal{H}^*, P^*\right)$ satisfies condition (2), and vice-versa. The claim follows. □

We are now ready for the correctness proof. We show that $\textsc{Test}(P, \mathcal{H})$, with P and $\mathcal{H}$ satisfying either (1) or (2), computes either the closest pair between $\textsc{ch}(P)$ and $\textsc{fh}(\mathcal{H})$, if they are disjoint, or the closest pair between $\textsc{ch}(\mathcal{H}^*)$ and $\textsc{fh}(P^*)$, if the polytopes intersect.

We use induction on $\max\{|P|, |\mathcal{H}|\}$. For the base case, when the maximum is at most α, our algorithm uses the brute-force method. This certainly provides a correct answer, by our assumptions on P and $\mathcal{H}$ and by Theorem 3.6.

For the inductive set, we may assume that each recursive call to $\textsc{Test}(\cdot, \cdot)$ provides a correct answer. It remains to show (i) that the main loop succeeds in at most $2d + 1$ iterations; and (ii) if the main loop succeeds, the algorithm returns a valid closest pair.

Number of Iterations. We show that the algorithm will never return an Error, i.e., that the loop will succeed in at most $2d + 1$ iterations. To start, we observe that the cases in the algorithm cannot alternate: first, we encounter only Case 2, then, we encounter only Case 1.

Lemma 4.2. *It the main loop in algorithm $\textsc{Test}(P, \mathcal{H})$ encounters Case 1, it will never again encounter Case 2.*

Proof. In each unsuccessful iteration, the set N grows by at least one element, so the convex polytope $\textsc{fh}(N^*)$ becomes smaller. Once $\textsc{fh}(N^*)$ and $\textsc{ch}(\mathcal{H}^*)$ are disjoint, they will remain disjoint for the rest of the algorithm, and by our inductive hypothesis, this will be reported correctly by the recursive calls to $\textsc{Test}(\cdot, \cdot)$. □

We now bound the number of iterations in Case 2.

Lemma 4.3. *The algorithm can have at most $d + 1$ iterations in Case 2. If there are $d + 1$ iterations in Case 2, then the last iteration is successful and the algorithm terminates.*

Proof. Suppose there are at least $d + 2$ iterations in Case 2. By Lemma 4.2, each iteration until this point encounters Case 2. Let $N_1 \subset N_2 \subset \cdots \subset N_{d+2}$ be the set N at the beginning of the first $d + 2$ iterations in Case 2. By Lemma 3.7 and the inductive hypothesis, each time we run into Case 2 unsuccessfully, the distance between $\textsc{ch}(N)$ and $\textsc{fh}(\mathcal{H})$ decreases strictly. Since the first $d + 1$ iterations in Case 2 are not successful, this means $d(\textsc{ch}(N_i), \textsc{fh}(\mathcal{H})) > d(\textsc{ch}(N_{i+1}), \textsc{fh}(\mathcal{H}))$, for $i = 1, \ldots, d + 1$.

Because the $(d + 2)$-th iteration runs into Case 2, it follows that $\textsc{ch}(N_{d+2})$ does not intersect $\textsc{fh}(\mathcal{H})$. Let x, y be the closest pair between $\textsc{fh}(\mathcal{H})$ and $\textsc{ch}(N_{d+2})$, with $y \in \textsc{ch}(N_{d+2})$. Then, y must lie on a face of $\textsc{ch}(N_{d+2})$, and by Carathéodory's theorem [20, Theorem 1.2.3], there is a set $B \subseteq N_{d+2}$ with at most d elements such that $y \in \textsc{ch}(B_{d+2})$. We claim that in each prior iteration $i = 1, \ldots, d + 1$, the conflict set V_H must contain at least one new element of B. Otherwise, if all the elements of B were already in some N_i, with $i \leq d + 1$, then $\textsc{ch}(N_i)$ would contain y and hence have a distance to $\textsc{fh}(\mathcal{H})$ smaller or equal than $\textsc{ch}(N_{d+2})$, leading to a contradiction. Similarly, if in an iteration $i \leq d + 1$, all elements of B were contained in H, then the distance between $\textsc{ch}(N_i)$ and $\textsc{fh}(\mathcal{H})$ could not strictly decrease, by Lemma 3.7. However, B

contains only d elements, and we have $d+1$ iterations, so we cannot add a new element of B at the end of each iteration Thus, the main loop can have at most d unsuccessful iterations in Case 2 before either encountering Case 2 successfully or reaching Case 1. □

Lemma 4.4. *The algorithm can have at most $d+1$ iterations in Case 1.*

Proof. Similar to the proof of Lemma 4.3, assume that we have at least $d+2$ iterations in Case 1. Let $N_1 \subset N_2 \subset \cdots \subset N_{d+2}$ be the set N at the beginning of each such iteration. By Lemma 3.8 and the inductive hypothesis, each time we encounter Case 1 unsuccessfully, we strictly increase the distance between $\text{CH}(\mathcal{H}^*)$ and $\text{FH}(N^*)$. That is, $d(\text{CH}(\mathcal{H}^*), \text{FH}({N_i}^*)) > d(\text{CH}(\mathcal{H}^*), \text{FH}({N_{i+1}}^*))$, for $i = 1, \ldots, d+1$.

Because we run into Case 1 in the $(d+2)$-th iteration, it follows that $\text{CH}(\mathcal{H}^*)$ and $\text{FH}({N_{d+2}}^*)$ do not intersect. Let x, y be the closest pair between $\text{CH}(\mathcal{H}^*)$ and $\text{FH}({N_{d+2}}^*)$, with $y \in \text{FH}({N_{d+2}}^*)$. Let B be the at most d elements in N_{d+2} such that x, y is the closest pair of $\text{CH}(\mathcal{H}^*)$ and $\text{FH}(B^*)$. Note that y could either be a vertex of $\text{FH}(B^*)$, or lie in the relative interior of one of its faces. Observe that in each unsuccessful iteration in Case 1, V_y must include a new member of B. Otherwise, if all the elements of B were already in some N_i with $i \leq d+1$, then $\text{FH}({N_i}^*)$ would have a distance to $\text{CH}(\mathcal{H}^*)$ larger or equal than $\text{FH}({N_{d+2}}^*)$, leading to a contradiction. Similarly, if in an iteration $i \leq d+1$, all elements of B^* were not in conflict with y, then the distance between $\text{FH}({N_i}^*)$ and $\text{CH}(\mathcal{H}^*)$ could not strictly decrease, by Lemma 3.8. However, this is impossible, because B has d elements and we have at least $d+1$ unsuccessful iterations. Thus, in the $(d+1)$-th iteration at the latest, we would observe that V_y is empty and the algorithm would finish. That is, the main loop can run for at most d unsuccessful iterations in Case 1. □

Lemmas 4.2, 4.3, and 4.4 guarantee that the algorithm will finish successfully within $2d+1$ iterations and that it will never return an ERROR. It remains to argue that the algorithm reports a correct closest pair if one of the two cases is encountered successfully.

Correctness of the Closest Pair. We first analyze the success condition of Case 1, i.e., when $\text{CH}(\mathcal{H}^*)$ and $\text{FH}(N^*)$ are disjoint. This condition is triggered when we have a set $N \subseteq P$ and a point $y \in \text{FH}(N^*)$ such that $V_y = \emptyset$. By Lemma 3.8, the closest pair x, y between $\text{CH}(\mathcal{H}^*)$ and $\text{FH}(N^*)$ then coincides with the closest pair between $\text{CH}(\mathcal{H}^*)$ and $\text{FH}(P^*)$. In particular, this implies that $\text{CH}(\mathcal{H}^*)$ and $\text{FH}(P^*)$ are disjoint. Because the recursive call returns correctly the closest pair x, y between $\text{CH}(\mathcal{H}^*)$ and $\text{FH}(N^*)$ by induction, it follows that the algorithm correctly returns the closest pair between $\text{CH}(\mathcal{H}^*)$ and $\text{FH}(P^*)$.

Next, we analyze the success condition of Case 2, i.e., when $\text{CH}(\mathcal{H}^*)$ and $\text{FH}(N^*)$ intersect. This implies by Theorem 3.6 that $\text{CH}(N)$ and $\text{FH}(\mathcal{H})$ are disjoint. Let x, y be the closest pair between $\text{CH}(N)$ and $\text{FH}(\mathcal{H})$, with $y \in \text{CH}(N)$. The success condition of Case 2 is triggered when $V_H = \emptyset$. By Lemma 3.7, this means that x, y coincides with the closest pair between $\text{CH}(P)$ and $\text{FH}(\mathcal{H})$. In particular, $\text{CH}(P)$ and $\text{FH}(\mathcal{H})$ are disjoint. Because the recursive call returns correctly the closest pair x, y between $\text{CH}(N)$ and $\text{FH}(\mathcal{H})$ by induction, the algorithm correctly returns the closest pair between $\text{CH}(P)$ and $\text{FH}(\mathcal{H})$. This now shows that $\text{TEST}(P, \mathcal{H})$ is indeed correct.

4.4 The Final Algorithm

Finally, we show how to remove the initial assumption that $(P, \mathcal{H})$ satisfies either condition (1) or condition (2).

Theorem 4.5. *Let P be a set of n points in $\mathbb{R}^d$ and let $\mathcal{H}$ be a set of m halfspaces in $\mathbb{R}^d$. We can test if $\text{CH}(P)$ and $\text{FH}(\mathcal{H})$ intersect in $O(d^{O(d)}(n+m))$ time. If they do, then we compute a point in their intersection; otherwise, we compute a separating plane.*

Proof. Recall that our algorithm requires that either (1) $\mathbf{o} \notin \text{CH}(P)$ and $\mathbf{o} \in \text{FH}(\mathcal{H})$, or (2) $\mathbf{o} \in \text{CH}(P)$ and $\mathbf{o} \notin \bigcup_{H \in \mathcal{H}} H$ to work, which might not hold for the given P and $\mathcal{H}$. Thus, before running $\text{TEST}(P, \mathcal{H})$, we first compute a point in $\text{FH}(\mathcal{H})$ using standard linear programming and change the coordinate system so that this point coincides with $\mathbf{o}$. Then, we test using standard linear programming if $\mathbf{o} \in \text{CH}(P)$. If it is, we are

done. Otherwise, we guarantee that condition (1) is satisfied, and we can run $\textsc{Test}(P, \mathcal{H})$ in $O(d^{O(d)}(n+m))$ time. □

Acknowledgments. This work was initiated at the *Sixth Annual Workshop on Geometry and Graphs*, that took place March 11–16, 2018, at the Bellairs Research Institute. We would like to thank the organizers and all participants of the workshop for stimulating discussions and for creating a conducive research environment. We would also like Timothy M. Chan for answering our questions about LP-type problems and for pointing us to several helpful references.

References

[1] L. Barba and S. Langerman. Optimal detection of intersections between convex polyhedra. In *Proc. 26th Annu. ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 1641–1654, 2015.

[2] T. M. Chan. An optimal randomized algorithm for maximum Tukey depth. In *Proc. 15th Annu. ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 430–436, 2004.

[3] T. M. Chan. Improved deterministic algorithms for linear programming in low dimensions. *ACM Trans. Algorithms*, 14(3):30, 2018.

[4] T. M. Chan. Personal communication. 2018.

[5] B. Chazelle. An optimal algorithm for intersecting three-dimensional convex polyhedra. *SIAM J. Comput.*, 21:586–591, 1992.

[6] B. Chazelle. An optimal convex hull algorithm in any fixed dimension. *Discrete Comput. Geom.*, 10:377–409, 1993.

[7] B. Chazelle. *The Discrepancy Method - Randomness and Complexity.* Cambridge University Press, 2001.

[8] B. Chazelle and D. Dobkin. Detection is easier than computation (extended abstract). In *Proc. 12th Annu. ACM Sympos. Theory Comput. (STOC)*, pages 146–153, 1980.

[9] B. Chazelle and D. Dobkin. Intersection of convex objects in two and three dimensions. *J. ACM*, 34(1):1–27, January 1987.

[10] V. Chvátal. *Linear programming.* A Series of Books in the Mathematical Sciences. W. H. Freeman, 1983.

[11] K. L. Clarkson. A Las Vegas algorithm for linear programming when the dimension is small. In *Proc. 29th Annu. IEEE Sympos. Found. Comput. Sci. (FOCS)*, pages 452–456, 1988.

[12] K. L. Clarkson and P. W. Shor. Application of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4:387–421, 1989.

[13] D. Dobkin and D. Kirkpatrick. A linear algorithm for determining the separation of convex polyhedra. *J. Algorithms*, 6(3):381–392, 1985.

[14] H. Edelsbrunner. Computing the extreme distances between two convex polygons. *J. Algorithms*, 6(2):213–224, 1985.

[15] S. Har-Peled. *Geometric approximation algorithms.* American Mathematical Society, 2011.

[16] S. Kapoor and P. M. Vaidya. Fast algorithms for convex quadratic programming and multicommodity flows. In *Proc. 18th Annu. ACM Sympos. Theory Comput. (STOC)*, pages 147–159, 1986.

[17] M. K. Kozlov, S. P. Tarasov, and L. G. Hačijan. The polynomial solvability of convex quadratic programming. *Zh. Vychisl. Mat. i Mat. Fiz.*, 20(5):1319–1323, 1359, 1980.

[18] G. Louchard, S. Langerman, and J. Cardinal. Randomized optimization: a probabilistic analysis. *Discrete Mathematics & Theoretical Computer Science*, 2007.

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

[20] J. Matoušek. *Lectures on Discrete Geometry*. Springer-Verlag, 2002.

[21] D. E. Muller and F. P. Preparata. Finding the intersection of two convex polyhedra. *Theoret. Comput. Sci.*, 7(2):217–236, 1978.

[22] F. P. Preparata and M. I. Shamos. *Computational Geometry–An Introduction*. Springer-Verlag, 1985.

[23] R. Seidel. Small-dimensional linear programming and convex hulls made easy. *Discrete Comput. Geom.*, 6:423–434, 1991.

[24] M. Sharir and E. Welzl. A combinatorial bound for linear programming and related problems. *Proc. 9th Sympos. Theoret. Aspects Comput. Sci. (STACS)*, pages 567–579, 1992.

[25] G. M. Ziegler. *Lectures on Polytopes*. Springer-Verlag, 1995.