An Efficient Algorithm for Generalized Polynomial Partitioning and Its Applications[*]

Pankaj K. Agarwal [†] Boris Aronov[‡] Esther Ezra[§] Joshua Zahl[¶]

June 2, 2019

Abstract

In 2015, Guth proved that if $\mathcal{S}$ is a collection of n g-dimensional semi-algebraic sets in $\mathbb{R}^d$ and if $D \geq 1$ is an integer, then there is a d-variate polynomial P of degree at most D so that each connected component of $\mathbb{R}^d \setminus Z(P)$ intersects $O(n/D^{d-g})$ sets from $\mathcal{S}$. Such a polynomial is called a *generalized partitioning polynomial*. We present a randomized algorithm that computes such polynomials efficiently—the expected running time of our algorithm is linear in $|\mathcal{S}|$. Our approach exploits the technique of *quantifier elimination* combined with that of *ε-samples*.

We present five applications of our result. The first is a data structure for answering point-enclosure queries among a family of semi-algebraic sets in $\mathbb{R}^d$ in $O(\log n)$ time, with storage complexity and expected preprocessing time of $O(n^{d+\varepsilon})$. The second is a data structure for answering range search queries with semi-algebraic ranges in $O(\log n)$ time, with $O(n^{t+\varepsilon})$ storage and expected preprocessing time, where $t > 0$ is an integer that depends on d and the description complexity of the ranges. The third is a data structure for answering vertical ray-shooting queries among semi-algebraic sets in $\mathbb{R}^d$ in $O(\log^2 n)$ time, with $O(n^{d+\varepsilon})$ storage and expected preprocessing time. The fourth is an efficient algorithm for cutting algebraic planar curves into pseudo-segments. The fifth application is for eliminating depth cycles among triangles in 3-space, in which case we show a nearly-optimal algorithm to cut n pairwise disjoint non-vertical triangles in $\mathbb{R}^3$ into pieces that form a depth order.

1 Introduction

In 2015, Guth [11] proved that if $\mathcal{S}$ is a collection of n g-dimensional semi-algebraic sets[1] in $\mathbb{R}^d$ and if $D \geq 1$ is an integer, then there is a d-variate polynomial P of degree at most D so that each

[*]P. Agarwal was supported by NSF under grants CCF-15-13816, CCF-15-46392, and IIS-14-08846, by an ARO grant W911NF-15-1-0408, and by BSF Grant 2012/229 from the U.S.-Israel Binational Science Foundation. B. Aronov was supported by NSF grants CCF-12-18791 and CCF-15-40656, and by grant 2014/170 from the US-Israel Binational Science Foundation. E. Ezra was supported by NSF CAREER under grant CCF:AF 1553354 and by Grant 824/17 from the Israel Science Foundation. J. Zahl was supported by a NSERC Discovery grant.

[†]Department of Computer Science, Duke University, Box 90129, Durham, NC 27708-0129 USA pankaj@cs.duke.edu

[‡]Department of Computer Science and Engineering, Tandon School of Engineering, New York University, Brooklyn, NY 11201, USA. boris.aronov@nyu.edu.

[§]Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel and School of Math, Georgia Institute of Technology, Atlanta, Georgia 30332, USA. eezra3@math.gatech.edu.

[¶]Department of Mathematics, University of British Columbia, Vancouver, BC V6T 1Z2, Canada. jzahl@math.ubc.ca.

[1]Roughly speaking, a semi-algebraic set in $\mathbb{R}^d$ is the set of points in $\mathbb{R}^d$ that satisfy a Boolean formula over a set of polynomial inequalities. See [14, Chapter 2] for formal definitions of a semialgebraic set and its dimension.

connected component of $\mathbb{R}^d \setminus Z(P)$ intersects $O(n/D^{d-g})$ sets from $\mathcal{S}$ [2], where the implicit constant in the $O(\cdot)$ notation depends on d and on the degree and number of polynomials required to define each semi-algebraic set. We refer to such a polynomial P as a *generalized partitioning polynomial*. Guth's proof established the existence of a generalized partitioning polynomial, but it did not offer an efficient algorithm to compute such a polynomial for a given collection of semi-algebraic sets. In this paper we study the problem of computing a generalized partitioning polynomial efficiently and present a few applications of such an algorithm.

Related work. In 2010, Guth and Katz [12] resolved the Erdős distinct distances problem in the plane. A major ingredient in their proof was a partitioning theorem for points in $\mathbb{R}^d$. Specifically, they proved that given a set of n points in $\mathbb{R}^d$ and an integer $D \geq 1$, there is a d-variate *partitioning polynomial* P of degree at most D so that each connected component of $\mathbb{R}^d \setminus Z(P)$ contains $O(n/D^d)$ points from the set. Their polynomial partitioning theorem led to a flurry of new results in combinatorial and incidence geometry, harmonic analysis, and theoretical computer science.

The Guth-Katz result established the existence of a partitioning polynomial, but it did not provide an efficient algorithm to compute such a polynomial. In [3], Agarwal, Matoušek, and Sharir developed an efficient algorithm to compute partitioning polynomials, matching the degree bound obtained in [12] up to a constant factor. They used this algorithm to construct a linear-size data structure that can answer semialgebraic range queries amid a set of n points in $\mathbb{R}^d$ in time $O(n^{1-1/d}\,\mathrm{polylog}(n))$, which is near optimal.

A major open question in geometric searching is whether a semialgebraic range query can be answered in $O(\log n)$ time using a data structure of size roughly n^d; such a data structure is known only in very special cases, e.g., when query ranges are simplices [1]. If t is the number of (real valued) parameters needed to represent query ranges, then the best known data structure for semialgebraic range searching uses $O(n^{t+\varepsilon})$ space for $t \leq 4$ and $O(n^{2t-4+\varepsilon})$ space for $t > 4$ [1]. As we show below, an efficient algorithm for computing a generalized partitioning polynomial leads to a semialgebraic range searching data structure with $O(\log n)$ query time and $O(n^{t+\varepsilon})$ space.

In [4], the last three authors developed an efficient algorithm for constructing a partition of $\mathbb{R}^3$ adapted to a set of space curves. This partition is not given by a partitioning polynomial, but it shares many of the same properties. For other settings, however, no effective method is known for computing a partitioning polynomial.

Our results. Our main result is an efficient algorithm for computing a generalized partitioning polynomial for a family of semi-algebraic sets (Theorem 4): Given a set $\mathcal{S}$ of n semi-algebraic sets in $\mathbb{R}^d$, each of complexity at most b for some constant $b > 0$ (see Section 2 for the definition of the complexity of a semi-algebraic set), our algorithm computes a generalized partitioning polynomial of given degree D in expected time $O(ne^{\mathrm{Poly}(D)})$.

In Section 4 we present five applications of our algorithm:

(i) Let $\mathcal{S}$ be a family of n semi-algebraic sets in $\mathbb{R}^d$, each of complexity at most b for some constant $b > 0$. Each set in $\mathcal{S}$ is assigned a weight that belongs to a semigroup. We present a data structure of size $O(n^{d+\varepsilon})$, for any constant $\varepsilon > 0$, that can compute, in $O(\log n)$ time, the cumulative weight of the sets in $\mathcal{S}$ containing a query point. We refer to the latter as a *point-enclosure* query. The data structure can be constructed in $O(n^{d+\varepsilon})$ randomized expected time.

[2] Guth stated his result for the special case where the semi-algebraic sets are real algebraic varieties, but his proof in fact holds in the more general setting of semi-algebraic sets.

This is a significant improvement over the best known data structure by Koltun [16], for $d > 4$, that uses $O(n^{2d-4+\varepsilon})$ space.

(ii) Let P be a set of n points in $\mathbb{R}^d$, each of which is assigned a weight in a semigroup, and let $\mathcal{R}$ be a (possibly infinite) family of semi-algebraic sets in $\mathbb{R}^d$. Suppose that there exists a positive integer t and an injection $f\colon \mathcal{R} \to \mathbb{R}^t$ so that for each $p \in P$, the set $f(\{R \in \mathcal{R} \mid p \in R\})$ is a semi-algebraic set in $\mathbb{R}^t$ of complexity at most b. We can construct in $O(n^{t+\varepsilon})$ randomized expected time a data structure of size $O(n^{t+\varepsilon})$, for any constant $\varepsilon > 0$, that can compute in $O(\log n)$ time the cumulative weight of $P \cap \gamma$ for a query range $\gamma \in \mathcal{R}$.

(iii) Given a family $\mathcal{S}$ of n semi-algebraic sets in $\mathbb{R}^d$, we present a data structure of size $O(n^{d+\varepsilon})$, for any constant $\varepsilon > 0$, that can answer vertical ray shooting queries in $O(\log^2 n)$ time. The data structure can be constructed in $O(n^{d+\varepsilon})$ randomized expected time.

(iv) We follow the technique of Sharir and Zahl [18] to cut n algebraic planar curves into a collection of $O(n^{3/2+\varepsilon})$ *pseudo-segments* (that is, a collection of Jordan arcs, each pair of which intersects at most once), where the constant of proportionality depends on the degree of the curves. By exploiting Theorem 4, we show that this collection can be constructed in comparable time bound. We comment that applying the algorithm in the previous work of Aronov *et al.* [4] results in the same asymptotic bound on the number of pseudo-segments, however, its running time is quadratic in the number of curves. Thus our algorithm yields a considerable improvement in the running time.

(v) Finally, we follow the mechanism of Aronov *et al.* [5] to eliminate depth cycles among triangles in 3-space. Specifically, Aronov *et al.* [5] used the generalized partitioning polynomial of Guth [11] in order to show that n pairwise disjoint non-vertical triangles in $\mathbb{R}^3$ can be cut into $O(n^{3/2+\varepsilon})$ pieces that do not form cycles (and thus form the so-called *depth order*), for any $\varepsilon > 0$. Using Theorem 4 we show an efficient algorithm to produce these pieces in comparable time bound.

Throughout this paper we assume that d, b, and ε are constants. Whenever big-O notation is used, the implicit constant may depend on d, b, and ε.

2 Preliminaries

In what follows, the *complexity* of a semi-algebraic set S in $\mathbb{R}^d$ is the minimum value b so that S can be represented as the set of points $x \in \mathbb{R}^d$ satisfying a Boolean formula with at most b atoms of the form $P(x) < 0$ or $P(x) = 0$, with each P being a d-variate polynomial of degree at most b.

Our analysis makes extensive use of concepts and results from real algebraic geometry and random sampling. We review them below.

2.1 Polynomials, partitioning, and quantifier elimination

Sign conditions. Let $\mathbb{R}[x_1, \ldots, x_d]$ denote the set of all d-variate polynomials with real coefficients. For $P_1, \ldots, P_\ell \in \mathbb{R}[x_1, \ldots, x_d]$, a *sign condition* on $(P_1, \ldots, P_\ell)$ is an element of $\{-1, 0, 1\}^\ell$. A *strict sign condition* on $(P_1, \ldots, P_\ell)$ is an element of $\{-1, 1\}^\ell$. A sign condition $(\nu_1, \ldots, \nu_\ell) \in \{-1, 0, 1\}^\ell$ is *realizable* if the set

$$\{x \in \mathbb{R}^d \mid \operatorname{sign}(P_j(x)) = \nu_j \text{ for each } j = 1, \ldots, \ell\} \tag{1}$$

is non-empty. A realizable strict sign condition is defined analogously. The set (1) is called the *realization* of the sign condition. The set of *realizations of sign conditions* (resp., *realizations of strict sign conditions*) corresponding to the tuple $(P_1, \ldots, P_\ell)$ is the collection of all non-empty sets of the above form. These sets are pairwise disjoint and partition $\mathbb{R}^d$, by definition.

While a tuple of ℓ polynomials has 3^ℓ sign conditions and 2^ℓ strict sign conditions, Milnor and Thom (see, e.g., [17, 15]) showed that any ℓ polynomials in $\mathbb{R}[x_1, \ldots, x_d]$ of degree at most s (with $2 \le d \le \ell$) have at most $(50s\ell/d)^d$ realizable sign conditions.

Polynomials and partitioning. The set of polynomials in $\mathbb{R}[x_1, \ldots, x_d]$ of degree at most b is a real vector space of dimension $\binom{b+d}{d}$; we identify this vector space with $\mathbb{R}^{\binom{b+d}{d}}$. For a point $q \in \mathbb{R}^{\binom{b+d}{d}}$, let $P_q \in \mathbb{R}[x_1, \ldots, x_d]$ be the corresponding polynomial of degree at most b.

Remark 1. Consider the polynomial $Q(q,x) \in \mathbb{R}[q_1, \ldots, q_{\binom{b+d}{d}}, x_1, \ldots, x_d]$ given by $Q(q,x) \coloneqq P_q(x)$. Since we can write $Q(q,x) = \sum_{i=1}^{\binom{b+d}{d}} q_i H_i(x)$, where H_i is a monomial of degree at most b, Q has degree $b+1$.

For each positive integer j, let D_j be the smallest positive integer so that $\binom{D_j+d}{d} > 2^{j-1}$; we have $D_j = O(2^{j/d})$. Let k be a positive integer. For each $j = 1, \ldots, k$, pick a 2^{j-1}-dimensional subspace $\mathbb{V}_j$ of $\mathbb{R}^{\binom{D_j+d}{d}}$. These subspaces $\mathbb{V}_j$ will be fixed hereafter. Define the product space

$$\mathbb{Y}_k \coloneqq \bigtimes_{j=1}^{k} \mathbb{V}_j. \tag{2}$$

We identify each point $y = (y_1, \ldots, y_k) \in \mathbb{Y}_k$, where $y_j \in \mathbb{V}_j$, with a k-tuple of polynomials $\mathbf{P}_y = (P_{y_1}, \ldots, P_{y_k})$. For each $j = 1, \ldots, k$, $\deg(P_j) = O(2^{j/d})$ and thus $\deg\left(\prod_{j=1}^{k} P_j\right) = O(2^{k/d})$.

Let $\mathbf{P}_y \in \mathbb{Y}_k$, let $\mathcal{S}$ be a collection of semi-algebraic sets in $\mathbb{R}^d$, and let $\alpha \ge 1$. We say that $\mathbf{P}_y$ is a *(k,α)-partitioning tuple* for $\mathcal{S}$ if $\mathbf{P}_y$ has exactly 2^k realizable strict sign conditions and the realization of each of them intersects at most $|\mathcal{S}|/\alpha$ sets from $\mathcal{S}$.[3] Guth [11] proved that, for an appropriate choice of α, a (k,α)-partitioning tuple is guaranteed to exist:

Proposition 1 (Generalized Polynomial Partitioning [11])**.** *Let $\mathcal{S}$ be a family of semi-algebraic sets in $\mathbb{R}^d$, each of dimension at most g and complexity at most b. For each $k \ge 1$, there exists a (k,α)-partitioning tuple for $\mathcal{S}$, with $\alpha = \Omega_{b,d}(2^{k(1-g/d)})$.*

We also recall Theorem 2.16 from [8]:

Proposition 2. *Let $\mathcal{P}$ be a set of s polynomials in $\mathbb{R}[x_1, \ldots, x_d]$ of degree at most t. Then there is an algorithm that computes a set of points meeting every connected component of every realizable sign condition on $\mathcal{P}$ in time $O(s^d t^{O(d)})$. There is also an algorithm providing the list of signs of all the polynomials of $\mathcal{P}$ at each of these points in time $O(s^{d+1} t^{O(d)})$.*

Singly exponential quantifier elimination. Let h and ℓ be non-negative integers and let $\mathcal{P} = \{P_1, \ldots, P_s\} \subset \mathbb{R}[x_1, \ldots, x_h, y_1, \ldots, y_\ell]$. Let $\Phi(y)$ be a first-order formula given by

$$(\exists x_1, \ldots, x_h) F(P_1(x,y), \ldots, P_s(x,y)), \tag{3}$$

where $y = (y_1, \ldots, y_\ell)$ is a block of ℓ free variables; x is a block of h variables, and $F(P_1, \ldots, P_s)$ is a quantifier-free Boolean formula with atomic predicates of the form $\operatorname{sign}(P_i(x_1, \ldots, x_h, y)) = \sigma$, with $\sigma \in \{-1, 0, 1\}$.

[3]As in [11], we work with a k-tuple of polynomials instead of a single polynomial so that we can bound the number of sets intersected by the realization of a sign condition rather than by a connected component of a realization.

The Tarski-Seidenberg theorem states that the set of points $y \in \mathbb{R}^\ell$ satisfying the formula $\Phi(y)$ is semi-algebraic. The next proposition is a quantitative version of this result that bounds the number and degree of the polynomial equalities and inequalities needed to describe the set of points satisfying $\Phi(y)$. This proposition is known as a "singly exponential quantifier elimination," and its more general form (where $\Phi(y)$ may contain a mix of $\forall$ and $\exists$ quantifiers) can be found in [8, Theorem 2.27].

Proposition 3. Let $\mathcal{P}$ be a set of at most s polynomials, each of degree at most t in $h + \ell$ real variables. Given a formula $\Phi(y)$ of the form (3), there exists an equivalent quantifier-free formula

$$\Psi(y) = \bigvee_{i=1}^{I} \bigwedge_{j=1}^{J_i} \Big(\bigvee_{n=1}^{N_{i,j}} \operatorname{sign}(P_{ijn}(y)) = \sigma_{ijn} \Big), \tag{4}$$

where P_{ijn} are polynomials in the variables y, $\sigma_{ijn} \in \{-1, 0, 1\}$,

$$\begin{aligned} I &\le s^{(h+1)(\ell+1)} t^{O(h\cdot\ell)}, \\ J_i &\le s^{(h+1)} t^{O(h)}, \\ N_{ij} &\le t^{O(h)}, \end{aligned} \tag{5}$$

and the degrees of the polynomials $P_{ijn}(y)$ are bounded by $t^{O(h)}$. Moreover, there is an algorithm to compute $\Psi(Y)$ in time $s^{(h+1)(\ell+1)} t^{O(h\cdot\ell)}$.

2.2 Range spaces, VC dimension, and ε-samples

We first recall several standard definitions and results from [13, Chapter 5]. A *range space* is a pair $\Sigma = (X, \mathcal{R})$, where X is a set and $\mathcal{R}$ is a collection of subsets of X. Let $(X, \mathcal{R})$ be a range space and let $A \subset X$ be a set. We define the *restriction* of Σ to A, denoted by Σ_A to be $(A, \mathcal{R}_A)$, where $\mathcal{R}_A := \{R \cap A \mid R \in \mathcal{R}\}$. If A is finite, then $|\mathcal{R}_A| \le 2^{|A|}$. If equality holds, then we say A is *shattered.* We define the *shatter function* by $\pi_{\mathcal{R}}(z) := \max_{|A|=z} |\mathcal{R}_A|$. The *VC dimension* of Σ is the largest cardinality of a set shattered by $\mathcal{R}$. If arbitrarily large finite subsets can be shattered, we say that the VC dimension of Σ is infinite.

Let Σ be a range space, A a finite subset of X, and $0 \le \varepsilon \le 1$. A set $B \subset A$ is an *ε-sample* (also known as *ε-approximation*) of Σ_A if

$$\Big| \frac{|A \cap R|}{|A|} - \frac{|B \cap R|}{|B|} \Big| \le \varepsilon \quad \forall R \in \mathcal{R}.$$

The following classical theorem of Vapnik and Chervonenkis [19] guarantees that, if the VC-dimension of Σ is finite, then for each positive $\varepsilon > 0$, a sufficiently large random sample of A is likely to be an ε-sample.[4]

Proposition 4 (ε-Sample Theorem)**.** Let $\Sigma = (X, \mathcal{R})$ be a range space of VC dimension at most d and let $A \subset X$ be finite. Let $0 < \varepsilon, \delta < 1$. Then a random subset $B \subset A$ of cardinality $\frac{8d}{\varepsilon^2} \log \frac{1}{\varepsilon\delta}$ is an ε-sample for Σ_A with probability at least $1 - \delta$.

Proposition 5 ([15, 13])**.** Let $\Sigma = (X, \mathcal{R})$ be a range space whose shatter function $\pi_{\mathcal{R}}(z)$ satisfies the bound $\pi_{\mathcal{R}}(z) \le C z^\rho$, for all positive integers z, where $\rho > 0$ is a real parameter. Then Σ has VC dimension at most $4\rho \log(C\rho)$.

[4]The stated bound is not the strongest possible (see, e.g., [13, Chapter 7] for an improved bound), but is sufficient for our purposes.

We next closely follow the arguments in the proof of Corollary 2.3 from [15], and show the following theorem:

Theorem 1. Let $Z \subset \mathbb{R}^d \times \mathbb{R}^\ell$ be a semi-algebraic set of complexity b. For each $y \in \mathbb{R}^\ell$, define $R_y = \{x \in \mathbb{R}^d \mid (x, y) \in Z\}$. Then the range space $(\mathbb{R}^d, \{R_y \mid y \in \mathbb{R}^\ell\})$ has VC dimension at most $200\ell^2 \log b$.

Proof. By assumption, there are polynomials $f_1, \dots, f_b$ and a Boolean formula Φ, so that, for $(x, y) \in \mathbb{R}^m \times \mathbb{R}^n$, $(x, y) \in Z$ if and only if $\Phi(f_1(x, y) \geq 0, \dots, f_b(x, y) \geq 0) = 1$.

Put $\mathcal{R} = \{R_y \mid y \in \mathbb{R}^n\}$. Fix a positive integer z and let $p_1, \dots, p_z \in \mathbb{R}^m$. Our goal is to bound

$$\left|\{R \cap \{p_1, \dots, p_z\} \mid R \in \mathcal{R}\}\right| = \left|\{R_y \cap \{p_1, \dots, p_z\} \mid y \in \mathbb{R}^n\}\right|.$$

For each $j = 1, \dots, z$, define

$$W_j = \{y \in \mathbb{R}^n \mid \Phi(f_1(p_j, y) \geq 0, f_2(p_j, y) \geq 0, \dots, f_b(p_j, y) \geq 0) = 1\}.$$

Let $A \subset [z] = \{1, \dots, z\}$ and suppose that there exists $y \in \mathbb{R}^n$ with $R_y \cap \{p_1, \dots, p_z\} = A$. This means $y \in W_j$ for each $j \in A$ and $y \notin W_j$ for each $j \in [z] \setminus A$, i.e., the semi-algebraic set S_A consisting of those points $y \in \mathbb{R}^n$ satisfying the Boolean formula

$$\bigwedge_{j \in A} \big(\Phi(f_1(p_j, y) \geq 0, f_2(p_j, y) \geq 0, \dots, f_b(p_j, y) \geq 0) = 1\big)$$
$$\wedge \bigwedge_{j \in [z] \setminus A} \big(\Phi(f_1(p_j, y) \geq 0, f_2(p_j, y) \geq 0, \dots, f_b(p_j, y) \geq 0) = 0\big)$$

is non-empty. Observe that if A and A' are distinct subsets of $[z]$, then S_A and $S_{A'}$ are disjoint and, in fact,

$$\left|\{R_y \cap \{p_1, \dots, p_z\} \mid y \in \mathbb{R}^n\}\right| = \left|\{A \subset [z] \mid S_A \neq \emptyset\}\right|.$$

Each of the non-empty sets S_A contains at least one realization of a sign condition of the bz polynomials

$$\{f_i(p_j, y) \mid i = 1, \dots, b;\ j = 1, \dots, z\},$$

each of degree at most b. By a result of Milnor and Thom stated in Section 2.1, these polynomials determine at most $(50 \cdot b \cdot bz/n)^n \leq (50b^2z)^n$ realizable sign conditions. Thus

$$|\{R_y \cap \{p_1, \dots, p_z\} \mid y \in \mathbb{R}^n\}| \leq (50b^2z)^n. \tag{6}$$

Since (6) holds for every choice of $p_1, \dots, p_z \in \mathbb{R}^m$, we conclude that

$$\pi_{\mathcal{R}}(z) \leq (50b^2z)^n.$$

By Proposition 5, $(\mathbb{R}^m, \{R_y \mid y \in \mathbb{R}^n\})$ has VC dimension at most $4n \log((50b^2)^n n) \leq 200n^2 \log b$. □

3 Computing Generalized Polynomial Partition

In this section we obtain the main result of the paper: given a collection $\mathcal{S}$ of semi-algebraic sets in $\mathbb{R}^d$, each of dimension at most g and complexity at most b, a $(k, \Omega_{b,d}(D^{d-g}))$-partitioning tuple for $\mathcal{S}$ can be computed efficiently. We obtain this result in several steps. Given a semialgebraic set

S, a sign condition $\sigma \in \{-1,+1\}^k$ and a real value $b > 0$, we first show that the set of k-tuples of degree-b polynomials whose realization of σ intersects S is a semialgebraic set. This in turn implies that, if $S_1, \ldots, S_n$ are semi-algebraic sets and if $m \leq n$, then the set of k-tuples of degree-b polynomials whose realization intersects at most m of the sets $S_1, \ldots, S_n$ is semi-algebraic. We use a quantifier-elimination algorithm to find a desired k-tuple. Unfortunately, the running time of the algorithm is exponential in n. We reduce the running time of the algorithm by using a random sampling technique — we show that it suffices to compute a partitioning tuple with respect to a small-size random subset of $\mathcal{S}$.

3.1 The parameter space of semi-algebraic sets

Fix positive integers b, d, g, and k, and let $D = 2^{k/d}$. Hereafter we assume that $D = \Omega(2^b)$, which can be enforced by choosing k sufficiently large.

As above, let $\mathcal{S}$ be a family of semi-algebraic sets in $\mathbb{R}^d$, each of dimension at most g and complexity at most b. Let $G\colon \{0,1\}^b \to \{0,1\}$ be a Boolean function. Let $\mathbb{X} = \left(\mathbb{R}^{\binom{b+d}{d}}\right)^b$. We identify a point $x = (q_1, \ldots, q_b) \in \mathbb{X}$ with the semi-algebraic set

$$Z_{x,G} = \{v \in \mathbb{R}^d \mid G(P_{q_1}(v) \geq 0, \ldots, P_{q_b}(v) \geq 0) = 1\} \subset \mathbb{R}^d.$$

Observe that each semi-algebraic set in $\mathcal{S}$ is of the form $Z_{x,G}$ for some choice of $x \in \mathbb{X}$ and a Boolean function G. Let $\mathbb{Y} = \mathbb{Y}_k$. For each $y \in \mathbb{Y}$, define $S_y \coloneqq \{u \in \mathbb{R}^d \mid P_1(u) > 0, \ldots, P_k(u) > 0\}$, where $(P_1, \ldots, P_k)$ is the tuple associated with y. Define

$$W_G \coloneqq \{(x,y) \in \mathbb{X} \times \mathbb{Y} \mid Z_{x,G} \cap S_y \neq \emptyset\}.$$

Theorem 2. The set W_G is semi-algebraic; it is defined by $O(e^{\mathrm{Poly}(D)})$ polynomials, each of degree $D^{O(d)}$.

Proof. Define $V = \{(x,y,v) \in \mathbb{X} \times \mathbb{Y} \times \mathbb{R}^d \mid v \in Z_{x,G} \cap S_y\}$. The condition $v \in Z_{x,G}$ is a Boolean condition on b polynomial inequalities. By Remark 1, each of these polynomials has degree at most $b+1$. Similarly, the condition $v \in S_y$ consists of k polynomial inequalities, each of degree at most $D+1$. This means that there exists a set of polynomials $\mathcal{Q} = \{Q_1, \ldots, Q_{b+k}\}$ of degree $b+D+1$ in the variables x, y, v, and a Boolean function $F(z_1, \ldots, z_{b+k})$ so that

$$V = \{(x,y,v) \in \mathbb{X} \times \mathbb{Y} \times \mathbb{R}^d \mid F(Q_1(x,y,v) \geq 0, \ldots, Q_{b+k}(z,y,v) \geq 0) = 1\}.$$

With the above definitions

$$W_G = \{(x,y) \mid \exists (v_1, \ldots, v_d)\colon F(Q_1(x,y,v), \ldots, Q_{b+k}(x,y,v)) = 1\}.$$

We now apply Proposition 3. We have a set $\mathcal{Q}$ of $s = b+k$ polynomials, each of degree at most $t = b+D+1$. The variables h and ℓ from the hypothesis of Proposition 3 are set to $h = d$ and $\ell = O\left(\binom{b+d}{d}^b + D^d\right) = \mathrm{Poly}(D)$; recall that D is sufficiently larger than b, and thus ℓ is a suitably chosen polynomial function of D. With these assignments, Proposition 3 says that W_G can be expressed as a quantifier-free formula of the form

$$\bigvee_{i=1}^{I} \bigwedge_{j=1}^{J_i} \Big(\bigvee_{n=1}^{N_{i,j}} \mathrm{sign}(P_{ijn}(x,y)) = \sigma_{ijn} \Big), \tag{7}$$

where P_{ijn} are polynomials in the variables (x, y), $\sigma_{ijn} \in \{-1, 0, 1\}$,

$$\begin{aligned} I &\le (b+k)^{\mathrm{Poly}(D)}(b+D)^{\mathrm{Poly}(D)} = O(e^{\mathrm{Poly}(D)}), \\ J_i &\le (b+k)^{d+1}(b+D)^{O(d)} = O(D^{O(d)}), \\ N_{ij} &\le (b+D)^{O(d)} = O(D^{O(d)}), \end{aligned} \tag{8}$$

where the degrees of the polynomials $P_{ijn}(y)$ are bounded by $(b+D)^{O(d)} = D^{O(d)}$.

Summarizing, the quantifier-free formula (7) for W_G is a Boolean combination of $O(e^{\mathrm{Poly}(D)})$ polynomial inequalities, each of degree $D^{O(d)}$, as claimed. $\square$

3.2 A singly-exponential algorithm

In this section, we discuss how to compute a (k, α)-partitioning tuple (for an appropriate value of α) for a small number m of semi-algebraic sets.

Theorem 3. Let $\mathcal{S}$ be a family of m semi-algebraic sets in $\mathbb{R}^d$, each of dimension at most g and complexity at most b. Let $1 \le k \le \log m$ and let $D = 2^{k/d}$. Then a $(k, \Omega_{b,d}(D^{d-g}))$-partitioning tuple for $\mathcal{S}$ can be computed in $O(e^{\mathrm{Poly}(m)})$ time.

Proof. Set $\mathbb{Y} = \mathbb{Y}_k$. As above, we identify points in $\mathbb{Y}$ with k tuples $(P_1, \dots, P_k)$ of polynomials. The argument in Theorem 2, as well as the fact that the class of semi-algebraic sets is closed under the operation of taking a projection, show that for each $S \in \mathcal{S}$ and each $\sigma \in \{-1, 1\}^k$,

$$I_{S,\sigma} := \{y \in \mathbb{Y} \mid S \cap \{\sigma_1 P_1 > 0, \sigma_2 P_2 > 0, \dots, \sigma_k P_k > 0\} \neq \emptyset\}$$

is a semi-algebraic set in $\mathbb{Y}$ that can be expressed as a Boolean combination of $O(e^{\mathrm{Poly}(D)})$ polynomials, each of degree $D^{O(d)}$. Moreover, it can be computed in time $O(e^{\mathrm{Poly}(D)})$ (see once again Proposition 3).

Let $C_{b,d}$ be a constant to be specified later (the constant will depend only on b and d) and let $N = C_{b,d}|\mathcal{S}|D^{g-d} + 1$; observe that $N = O(m)$. For each $\sigma \in \{-1, 1\}^k$ and for each set $\mathcal{S}' \subset \mathcal{S}$ of cardinality $|\mathcal{S}'| \ge N$, the set $\{y \in \mathbb{Y} \mid y \in I_{S,\sigma}$ for every $S \in \mathcal{S}'\}$ is a semi-algebraic set in $\mathbb{Y}$ that can be expressed as a Boolean combination of $O(N' e^{\mathrm{Poly}(D)}) = O(m e^{\mathrm{Poly}(D)})$ polynomials, each of degree $D^{O(d)}$, where $N' = |\mathcal{S}'|$. Therefore

$$\mathcal{K} := \bigcup_{\substack{S' \subset \mathcal{S} \\ |\mathcal{S}'| \ge N}} \{y \in \mathbb{Y} \mid y \in I_{S,\sigma} \text{ for every } S \in \mathcal{S}'\} \tag{9}$$

is a semi-algebraic set in $\mathbb{Y}$ that can be expressed as a Boolean combination of

$$\sum_{\substack{S' \subset \mathcal{S} \\ |\mathcal{S}'| \ge N}} O\left(\binom{m}{|S'|} m e^{\mathrm{Poly}(D)}\right) = O(e^{m + \mathrm{Poly}(D)}) = O(e^{\mathrm{Poly}(m)})$$

polynomials, each of degree $D^{O(d)}$. This and the fact that the class of semi-algebraic sets is closed under the operation of taking complement imply that

$$\mathrm{Good}(\sigma) := \mathbb{Y} \setminus \mathcal{K} = \{y \in \mathbb{Y} \mid y \in I_{S,\sigma} \text{ for at most } C_{b,d}|\mathcal{S}|D^{g-d} \text{ sets } S \in \mathcal{S}\}$$

is a semi-algebraic set in $\mathbb{Y}$ that can be expressed as a Boolean combination of $O(e^{\mathrm{Poly}(m)})$ polynomials, each of degree $D^{O(d)}$. This means that the set

$$\bigcap_{\sigma \in \{-1,1\}^k} \mathrm{Good}(\sigma) \tag{10}$$

is a semi-algebraic set in $\mathbb{Y}$ that can be expressed as a Boolean combination of $O(e^{\mathrm{Poly}(m)})$ polynomials, each of degree $D^{O(d)}$. Recall that by assumption $1 \le k \le \log m$ and $D = 2^{k/d}$. It thus follows that the degree is bounded by $\mathrm{Poly}(m)$. Similarly, the dimension of the space $\mathbb{Y}$ is bounded by $\mathrm{Poly}(m)$ as well.

Proposition 1 guarantees that if $C_{b,d}$ is selected sufficiently large, then the set (10) is non-empty. By Proposition 2, it is possible to locate a point in this set in $O(e^{\mathrm{Poly}(m)})$ time, concluding the proof of the theorem.

□

3.3 Speeding up the algorithm using ε-sampling

In this section we first state and prove the following lemma:

Lemma 1. For every choice of positive integers b and d, there is a constant $C = C(b,d)$ so that the following holds. Let C_0 be a positive integer. Let $\mathcal{S}$ be a finite collection of semi-algebraic sets in $\mathbb{R}^d$, each of dimension at most g and complexity at most b. Let k be a positive integer and let $D = 2^{k/d}$. Let $B \subset \mathcal{S}$ be a randomly chosen subset of $\mathcal{S}$ of cardinality at least CD^C and let $(P_1, \ldots, P_k)$ be a $(k, \frac{D^{d-g}}{C_0})$-partitioning tuple for B. Then with probability at least $1/2$, each of the D^d realizable sign conditions of $(P_1, \ldots, P_k)$ intersects $O(|\mathcal{S}|C_0 D^{g-d})$ elements from $\mathcal{S}$.

Proof. Define $\mathbb{X}$ and $\mathbb{Y}$ as above, and let $G\colon \{0,1\}^b \to \{0,1\}$. For each $y \in \mathbb{Y}$, define the range

$$R_{y,G} = \{x \in \mathbb{X} \mid Z_{x,G} \cap S_y \neq \emptyset\}.$$

Define $\mathcal{R}_G = \{R_{y,G} \mid y \in \mathbb{Y}\}$. By Theorems 1 and 2, the range space $(\mathbb{X}, \mathcal{R}_G)$ has VC dimension $O_{b,d}(\mathrm{Poly}(D))$. Define $\mathcal{R} = \bigcup_G \mathcal{R}_G$, where the union is taken over the 2^b Boolean functions $G\colon \{0,1\}^b \to \{0,1\}$. Since the shatter function grows by at most a multiplicative factor of 2^b, the VC dimension of the range space $(\mathbb{X}, \mathcal{R})$ is $O_{b,d}(\mathrm{Poly}(D))$ as well (this is a standard fact, see, e.g., [13, Chapter 5]).

We are now ready to prove the statement of the lemma. Set $\varepsilon = O_{b,d}(C_0 D^{g-d})$. Suppose that B is an ε-sample of $\mathcal{S}$ and that $(P_1, \ldots, P_k)$ is a $(k, \frac{D^{d-g}}{C_0})$-partitioning tuple for B. Then for each range $R \in \mathcal{R}$, we have $|B \cap R|/|B| \le O_{d,b}(C_0 D^{g-d})$. Combining this with ε-sample properties, we obtain:

$$\left| \frac{|\mathcal{S} \cap R|}{|\mathcal{S}|} - \frac{|B \cap R|}{|B|} \right| \le \varepsilon,$$

and by the choice of ε (with an appropriate constant of proportionality) we have:

$$\left| \frac{|\mathcal{S} \cap R|}{|\mathcal{S}|} \right| \le 2\varepsilon,$$

and thus

$$|\mathcal{S} \cap R| = O_{b,d}(|\mathcal{S}|C_0 D^{g-d}).$$

The corresponding cardinality of B is

$$\frac{\text{VC-dim}(\mathcal{R})}{\varepsilon^2} \log(2/\varepsilon) = O_{b,d}(\mathrm{Poly}(D)).$$ □

Note that Lemma 1 states that it is sufficient to consider a random subset B of size polynomial in D in order to obtain an appropriate partitioning tuple for the entire collection $\mathcal{S}$, with reasonable probability.

We next proceed as follows. We select a random sample of $\mathcal{S}$ of cardinality CD^C and use Theorem 3 to compute the corresponding partitioning tuple $(P_1, \dots, P_k)$. This takes $O(e^{\text{Poly}(D)})$ time. By Lemma 1, this tuple will be a $(k, \Omega_{b,d}(D^{d-g}))$-partitioning tuple for $\mathcal{S}$ with probability at least 1/2. We can verify whether the partitioning tuple works in $O(|\mathcal{S}| \operatorname{Poly}(D))$ time. If the tuple does not produce the appropriate partition, we discard it and try again; the expected number of trials is at most 2. The verification step is done as follows. For each semi-algebraic set $S \in \mathcal{S}$ we compute the subset of sign conditions of $(P_1, \dots, P_k)$, with which it has a non-empty intersection. To this end, we restrict each of the polynomials $P_1, \dots, P_k$ to S and apply Proposition 2 on this restricted collection, thereby obtaining a set of points meeting each connected component of each of the realizable sign conditions, as well as the corresponding list of signs of the restricted polynomials for each of these points. This is done in $O(D^{O(d)})$ time for a single semi-algebraic set $S \in \mathcal{S}$, and overall $O(|\mathcal{S}|D^{O(d)})$ time, over all sets. We refer the reader to [7] for further details concerning the complexity of the restriction of $P_1, \dots, P_k$ to S. We have thus shown:

Theorem 4. *Let $\mathcal{S}$ be a finite collection of semi-algebraic sets in $\mathbb{R}^d$, each of which has dimension at most g and complexity at most b. Let $k \geq 1$ and let $D = 2^{k/d}$. Then a $(k, \Omega_{b,d}(D^{d-g}))$-partitioning tuple for $\mathcal{S}$ can be computed in $O(|\mathcal{S}| \operatorname{Poly}(D) + e^{\operatorname{Poly}(D)})$ expected time by a randomized algorithm.*

4 Applications

In this section we describe a few applications of Theorem 4, namely, point-enclosure queries amid semi-algebraic sets, semi-algebraic range searching with logarithmic query time, vertical ray shooting amid semi-algebraic sets, cutting algebraic curves into pseudo-segments, and eliminating depth cycles among disjoint triangles in $\mathbb{R}^3$..

4.1 Point-enclosure queries

Let $\mathcal{S}$ be a set of n semi-algebraic sets in $\mathbb{R}^d$, each of complexity at most b. Each set S is assigned a weight $w(S)$. We assume that the weights belong to a semigroup, i.e., subtractions are not allowed, and that the semigroup operation can be performed in constant time. We wish to preprocess $\mathcal{S}$ into a data structure so that the cumulative weight of the sets in $\mathcal{S}$ that contain a query point can be computed in $O(\log n)$ time; we refer to this query as *point-enclosure* query. Note that if the weight of each set is 1 and the semi-group operation is Boolean $\vee$, then the point-enclosure query becomes a *union-membership query*: determine whether the query point lies in $\bigcup \mathcal{S}$.

We follow a standard hierarchical partitioning scheme of space, e.g., as in [10, 1], but use Theorem 4 at each stage. Using this hierarchical partition, we construct a tree data structure $\mathcal{T}$ of depth $O(\log n)$, and a query is answered by following a path in $\mathcal{T}$. More precisely, we fix sufficiently large positive constants $D = D(b, d)$ and $n_0 = n_0(D)$. If $n \leq n_0$, $\mathcal{T}$ consists of a single node that stores $\mathcal{S}$ itself. So assume that $n > n_0$. Using Theorem 4, we construct a tuple $\mathcal{P} = (P_1, \dots, P_k)$ of d-variate polynomials of degree at most D, which have $2^k = O(D^d)$ realizable sign conditions, each of which with a realization that meets the boundaries of at most $O(|\mathcal{S}|/D)$ sets of $\mathcal{S}$. For each realizable sign condition σ, let $\mathcal{S}_\sigma \subseteq \mathcal{S}$ be the family of sets whose boundaries meet the realization of σ, and let $\mathcal{S}^*_\sigma \subseteq \mathcal{S}$ be the family of sets that contain the realization of σ.

We compute $\mathcal{S}_\sigma$, $\mathcal{S}^*_\sigma$, and $W_\sigma = w(\mathcal{S}^*_\sigma)$, as follows: We first apply Proposition 2 to $\mathcal{P}$ to compute, in $O(D^{O(d)})$ time, a representative point ξ_σ in the realization of every realizable sign condition σ.

We fix a set $S \in \mathcal{S}$, mark every realizable sign condition σ that meets ∂S, and add S to the set $\mathcal{S}_\sigma$. This step is similar to the one described in the proof of Theorem 4, that is, we restrict each of the polynomials $P_1, \dots, P_k$ to the algebraic varieties representing the boundary of S and apply Proposition 2 to this restricted collection. Each remaining sign condition σ is either contained in S or disjoint from it, which can be determined by testing whether its representing point ξ_σ is contained in S. If the answer is yes, we add the weight $w(S)$ to W_σ. This task can be completed in overall $O(nD^{O(d)})$ time over all sets of $\mathcal{S}$.

We create the root v of $\mathcal{T}$ and store the tuple $\mathcal{P}$ at v. We then create a child z_σ for each realizable sign condition σ and store σ and W_σ at z_σ. We recursively construct the data structure for each $\mathcal{S}_\sigma$ and attach it to z_σ as its subtree. Since each node of $\mathcal{T}$ has degree at most $O(D^d)$ and the size of the subproblem reduces by a factor of D at each level of the recursion, a standard analysis shows that the total size of the data structure is $O(n^{d+\varepsilon})$, where $\varepsilon > 0$ is a constant that can be made arbitrarily small by choosing D and n_0 to be sufficiently large. Similarly, the expected preprocessing time is also $O(n^{d+\varepsilon})$.

Given a query point $q \in \mathbb{R}^d$, we compute the cumulative weight of the sets containing q by traversing a path in the tree in a top-down manner: We start from the root and maintain a partial weight W, which is initially set to 0. At each node v, we find the sign condition σ of the polynomial tuple at v whose realization contains q, add W_σ to W, and recursively query the child z_σ of v. The total query time is $O(\log n)$, where the constant of proportionality depends on D (and thus on ε). Putting everything together, we obtain the following:

Theorem 5. Let $\mathcal{S}$ be a set of n semi-algebraic sets in $\mathbb{R}^d$, each of complexity at most b for some constant $b > 0$, and let $w(S)$ be the weight of each set $S \in \mathcal{S}$ that belongs to a semigroup. Assuming that the semigroup operation can be performed in constant time, $\mathcal{S}$ can be preprocessed in $O(n^{d+\varepsilon})$ randomized expected time into a data structure of size $O(n^{d+\varepsilon})$, for any constant $\varepsilon > 0$, so that the cumulative weight of the sets that contain a query point can be computed in $O(\log n)$ time.

4.2 Range searching

Next, we consider range searching with semi-algebraic sets: Let P be a set of n points in $\mathbb{R}^d$. Each point $p \in P$ is assigned a weight $w(p)$ that belongs to a semigroup. Again we assume that the semigroup operation takes constant time. We wish to preprocess P so that, for a query range γ, represented as a semi-algebraic set in $\mathbb{R}^d$, the cumulative weight of $\gamma \cap P$ can be computed in $O(\log n)$ time. Here we assume that the query ranges (semi-algebraic sets) are parameterized as described in Section 3.1. That is, we have a fixed b-variate Boolean function G. A query range is represented as a point $x \in \mathbb{X} = \mathbb{R}^t$, for some $t \le \binom{b+d}{d}^b$, and the underlying semi-algebraic set is $Z_{x,G}$. We refer to t as the *dimension of the query space*, and to the range searching problem in which all query ranges are of the form $Z_{x,G}$ as (G,t)*-semi-algebraic range searching*.

For a point $p \in \mathbb{R}^d$, let $S_p \subseteq \mathbb{X}$ denote the set of semi-algebraic sets $Z_{x,G}$ that contain p, i.e., $S_p = \{x \in \mathbb{X} \mid p \in Z_{x,G}\}$. It can be checked that S_p is a semi-algebraic set whose complexity depends only on b, d, and G. Let $\mathcal{S} = \{S_p \mid p \in P\}$. For a query range $Z_{x,G}$, we now wish to compute the cumulative weight of the sets in $\mathcal{S}$ that contain x. This can be done using Theorem 5. Putting everything together, we obtain the following:

Theorem 6. Let P be a set of n points in $\mathbb{R}^d$, let $w(p)$ be the weight of $p \in P$ that belongs to a semigroup, and let G be a fixed b-variate Boolean function for some constant $b > 0$. Let $t \le \binom{b+d}{d}^b$ be the dimension of the query space. Assuming that the semigroup operation can be performed in constant time, P can be preprocessed in $O(n^{t+\varepsilon})$ randomized expected time into a data structure of size $O(n^{t+\varepsilon})$, for any constant $\varepsilon > 0$, so that a (G, t)-semi-algebraic range query can be answered in $O(\log n)$ time.

Remark. If G is the conjunction of a set of b polynomial inequalities, then the size of the data structure can be significantly improved by using a multi-level data structure, with a slight increase in the query time to $O(\log^b n)$; see, e.g., [1]. Roughly speaking, the value of t will now be the dimension of the parametric space of each polynomial defining the query semi-algebraic set, rather than the dimension of the parametric space of the entire semi-algebraic set (which is the conjunction of b such polynomials).

4.3 Vertical ray shooting

We next present an efficient data structure for answering *vertical ray-shooting* queries: Preprocess a collection $\mathcal{S}$ of n semi-algebraic sets in $\mathbb{R}^d$, each of complexity at most b, into a data structure so that the first set of $\mathcal{S}$ hit by ρ_q, the ray emanating in the $(+x_d)$-direction from a query point q, can be reported quickly. If there is more than one such set, the query procedure returns one of them, arbitrarily.

The overall data structure. Our data structure for vertical ray shooting is similar to the one described in Section 4.1, except that we store an auxiliary data structure at each node of the tree to determine which of its children the query procedure should visit recursively.

Again we fix two constants D and n_0. If $n \le n_0$, the tree data structure $\mathcal{T}$ consists of a single node that stores $\mathcal{S}$. So assume that $n > n_0$. We compute a partitioning polynomial tuple $\mathcal{P} = (P_1, \ldots, P_k)$ for $\mathcal{S}$ of degree D. For each realizable sign condition σ, we compute the set $\mathcal{S}_\sigma \subseteq \mathcal{S}$ whose boundaries meet the realization of σ. We create the root node v of $\mathcal{T}$ and create a child z_σ for each realizable sign condition σ. We store two auxiliary data structures $\mathsf{DS}_1(v)$ and $\mathsf{DS}_2(v)$ at v, described below, each of which can be constructed in $O(n^{d+\varepsilon})$ randomized expected time and requires $O(n^{d+\varepsilon})$ space. Given a query point $q \in \mathbb{R}^d$, $\mathsf{DS}_1(v), \mathsf{DS}_2(v)$ together determine, in $O(\log n)$ time, the sign condition σ whose realization contains the first intersection point of ρ_q with a set of $\mathcal{S}$. We recursively construct the data structure for each subset $\mathcal{S}_\sigma$ and attach it to z_σ as its subtree.

A standard analysis of multi-level data structures (see e.g. [1]) shows that the total size of $\mathcal{T}$ is $O(n^{d+\varepsilon})$, for any constant $\varepsilon > 0$, and that it can be constructed in $O(n^{d+\varepsilon})$ randomized expected time.

For a query point $q \in \mathbb{R}^d$, the first set hit by ρ_q can be computed by traversing a root-to-leaf path in $\mathcal{T}$. Suppose we are at a node v. If v is a leaf, then we naively check all sets in $\mathcal{S}_v$ to find the first among them hit by ρ_q. Otherwise, we use the auxiliary data structures $\mathsf{DS}_1(v)$ and $\mathsf{DS}_2(v)$ to determine in $O(\log n)$ time the sign condition σ whose realization contains the first intersection point of ρ_q and a set of $\mathcal{S}$. We recursively visit the child z_σ of v. Since the depth of $\mathcal{T}$ is $O(\log n)$, the total query time is $O(\log^2 n)$.

This completes the description of the overall algorithm. What remains is to describe the auxiliary data structures $\mathsf{DS}_1, \mathsf{DS}_2$.

Auxiliary data structures. Recall that the auxiliary data structures are used to determine the sign condition of $\mathcal{P}$ whose realization contains the first intersection point of a vertical ray with a set of $\mathcal{S}$. We first refine the realizations of sign conditions of $\mathcal{P}$ into "cylindrical" cells, as follows. Let $f = \prod_{i=1}^{k} P_i$; by construction, the degree of f is $O(D)$. By Warren [20, Theorem 2], the number of connected components of $\mathbb{R}^d \setminus Z(f)$ is at most $O(D)^d$; from now on we refer to these components as *cells*.[5] We refine the cells of $\mathbb{R}^d \setminus Z(f)$ using the so-called *first-stage CAD (cylindrical algebraic decomposition)*; see, e.g., [17, Chapter 5] for a detailed overview of standard CAD. That is, this is a simplified version of CAD, presented in [3].

Roughly speaking, the first-stage CAD for f is obtained by constructing a collection $\mathcal{G}$ of polynomials in the variables $x_1, \ldots, x_{d-1}$, whose zero sets contain the projection onto $\mathbb{R}^{d-1}$ of the set of points in $Z(f)$ of vertical tangency, self-intersection of zeros sets (roots with multiplicity), or a singularity of some other kind. Having constructed $\mathcal{G}$, the first-stage CAD is obtained as the arrangement $\mathcal{A}(\{f\}\cup\mathcal{G})$ in $\mathbb{R}^d$, where the polynomials in $\mathcal{G}$ are now regarded as d-variate polynomials, that is, we lift them in the x_d-direction; the geometric interpretation of the lifting operation is to erect a "vertical wall" in $\mathbb{R}^d$ over each zero set within $\mathbb{R}^{d-1}$ of a $(d-1)$-variate polynomial from $\mathcal{G}$, and the first-stage CAD is the arrangement of these vertical walls plus $Z(f)$. It follows by construction that the cells of $\mathcal{A}(\{f\}\cup\mathcal{G})$ are vertical stacks of "cylindrical" cells. In more detail, for each cell τ of $\mathcal{A}(\{f\}\cup\mathcal{G})$, there is unique cell φ of the $(d-1)$-dimensional arrangement $\mathcal{A}(\mathcal{G})$ in $\mathbb{R}^{d-1}$ such that one of the following two cases occur: (i) $\tau = \{(x, \xi(x)\} \mid x \in \varphi\}$, where $\xi\colon \varphi \to \mathbb{R}$ is a continuous semi-algebraic function (i.e., τ is the graph of ξ over φ); or (ii) $\tau = \{(x,t) \mid x \in \varphi, t \in (\xi_1(x), \xi_2(x))\}$, where ξ_i, $i = 1, 2$ is a continuous semi-algebraic function on φ, the constant function $\varphi \to \{-\infty\}$, or the constant function $\varphi \to \{+\infty\}$, and $\xi_1(x) < \xi_2(x)$ for all $x \in \varphi$ (i.e., τ is a cylindrical cell over φ bounded from below (resp. above) by the graph of ξ_1 (resp. ξ_2)). As stated in [3], the total number of cells in $\mathcal{A}(\{f\}\cup\mathcal{G})$ is $D^{O(d)}$, and each of them is a semi-algebraic set defined by $D^{O(d^4)}$ polynomials of degree $D^{O(d^3)}$ (this is, in fact, an application of Proposition 2).

For a cell φ of the $(d-1)$-dimensional arrangement $\mathcal{A}(\mathcal{G})$, let $V(\varphi)$ be the stack of cells of $\mathcal{A}(\{f\}\cup\mathcal{G})$ over φ, i.e., the set of cells that project to φ.

We note that the sign condition of $\mathcal{P}$ is the same for all points in a cell of $\mathcal{A}(\{f\}\cup\mathcal{G})$, i.e., each cell lies in the realization of a single sign condition of $\mathcal{P}$. It thus suffices to find the cell of $\mathcal{A}(\{f\}\cup\mathcal{G})$ that contains the first intersection point of a vertical ray with a set of $\mathcal{S}$ in order to find the sign condition of $\mathcal{P}$ whose realization contains such a point. We construct $\mathsf{DS}_1, \mathsf{DS}_2$ on the cells of $\mathcal{A}(\{f\}\cup\mathcal{G})$ to quickly determine the desired cell.

***The structure* DS_1.** Fix a cell τ of $\mathcal{A}(\{f\}\cup\mathcal{G})$. DS_1 is used to determine whether a query ray ρ_q whose source point lies in τ intersects any set of $\mathcal{S}$ inside τ.

For each input set $S \in \mathcal{S}$ that intersects τ, let $\uparrow S$ be the set of points x in $\mathbb{R}^d$ such that the vertical ray ρ_x intersects $S \cap \tau$, i.e., $\uparrow(S)$ is the union of the rays in the $(-x_d)$-direction emanating from the points of $S \cap \tau$. $\uparrow S$ is a semi-algebraic set whose complexity depends only on b, d, and D. Let $\uparrow\mathcal{S}_\tau = \{\uparrow S \mid S \in \mathcal{S}, S \cap \tau \neq \emptyset\}$. $\uparrow\mathcal{S}_\tau$ can be computed in $O(n)$ time. Using Theorem 5, we process $\uparrow\mathcal{S}$ into a data structure $\mathsf{DS}_1(\tau)$ of size $O(n^{d+\varepsilon})$ so that for a query point $q \in \mathbb{R}^d$, we can determine in $O(\log n)$ time whether $q \in \bigcup \uparrow\mathcal{S}$, i.e., whether ρ_q intersects any set of $\mathcal{S}$ within τ. We construct $\mathsf{DS}_1(\tau)$ for all cells τ of $\mathcal{A}(\{f\}\cup\mathcal{G})$. The total size of the data structure, summed over all cells of $\mathcal{A}(\{f\}\cup\mathcal{G})$, is $O(n^{d+\varepsilon})$, and it can be constructed in $O(n^{d+\varepsilon})$ randomized expected time.

***The structure* DS_2.** Fix a cell τ of $\mathcal{A}(\{f\}\cup\mathcal{G})$. DS_2 is used to determine whether a line parallel to the x_d-axis intersects any set of $\mathcal{S}$ inside τ.

For each input set $S \in \mathcal{S}$ that intersects τ, let $\downarrow S_\tau$ be the projection of $S \cap \tau$ onto the hyperplane $x_d = 0$. For a point $q \in \mathbb{R}^d$, the vertical line (parallel to the x_d-axis) through q intersects S inside τ if and only if $\downarrow q \in \downarrow S_\tau$ (where $\downarrow q$ is the projection of q onto the hyperplane $x_d = 0$). $\downarrow S_\tau$ is a $(d-1)$-dimensional semi-algebraic set whose complexity depends only on b and D. Let $\downarrow\mathcal{S}_\tau = \{\downarrow S_\tau \mid S \in \mathcal{S}, S \cap \tau \neq \emptyset\}$. $\downarrow\mathcal{S}_\tau$ can be constructed in $O(n)$ time. Using Theorem 5, we process $\downarrow\mathcal{S}_\tau$ into a data structure $\mathsf{DS}_2(\tau)$ of size $O(n^{d-1+\varepsilon})$ so that, for a query point $q \in \mathbb{R}^d$, we can

[5]This notion is somewhat different than the notion of realizable sign conditions, where one can have several connected components representing the same sign condition.

determine in $O(\log n)$ time whether $\downarrow q \in \bigcup \downarrow \mathcal{S}_\tau$, i.e., whether the vertical line through q intersects any set of $\mathcal{S}$ inside τ. We construct $\mathsf{DS}_2(\tau)$ for all cells of $\mathcal{A}(\{f\} \cup \mathcal{G})$. The total size of the data structure, summed over all cells of $\mathcal{A}(\{f\} \cup \mathcal{G})$, is $O(n^{d-1+\varepsilon})$, and it can be constructed in $O(n^{d-1+\varepsilon})$ randomized expected time.

Answering a query. Given a query point $q \in \mathbb{R}^d$, we determine the cell of $\mathcal{A}(\{f\} \cup \mathcal{G})$ that contains the first intersection point of ρ_q with a set of $\mathcal{S}$ as follows. First, we determine the cell τ of $\mathcal{A}(\{f\} \cup \mathcal{G})$ that contains the query point q. Using $\mathsf{DS}_1(\tau)$, we determine in $O(\log n)$ time whether ρ_q intersects $\mathcal{S}$ inside τ. If the answer is yes, then τ is the desired cell. So assume that the answer is no. Let φ be the cell of the $(d-1)$-dimensional arrangement $\mathcal{A}(\mathcal{G})$ such that $\downarrow q \in \varphi$. Let $V(\varphi) = \langle \tau_1, \ldots, \tau_r \rangle$ be the stack of cells over φ, and let $\tau = \tau_i$ for some $i \le k$. We visit the cells of $V(\tau)$ one by one in order, starting from τ_{i+1} until we find a cell τ_j such that $\downarrow q \in \bigcup \downarrow \mathcal{S}_{\tau_j}$. Since q lies below τ_j, ρ_q intersects $\mathcal{S}$ inside τ_j if and only if $\downarrow q \in \bigcup \downarrow(\mathcal{S}_{\tau_j})$. If there is no such cell, we conclude that ρ_q does not intersect $\mathcal{S}$. Otherwise τ_j is the cell of $\mathcal{A}(\{f\} \cup \mathcal{G})$ that contains the first intersection point of ρ_q with a set of $\mathcal{S}$.

Putting everything together we obtain the following.

Theorem 7. Let $\mathcal{S}$ be a collection of n semi-algebraic sets in $\mathbb{R}^d$, each of complexity at most b for some constant $b > 1$. $\mathcal{S}$ can be preprocessed, in $O(n^{d+\varepsilon})$ randomized expected time, into a data structure of size $O(n^{d+\varepsilon})$, for any constant $\varepsilon > 0$, so that a vertical ray-shooting query can be answered in $O(\log^2 n)$ time.

4.4 Cutting algebraic curves into pseudo-segments

Sharir and Zahl [18] presented a technique for cutting algebraic plane curves into pseudo-segments, by lifting curves into three dimensions. More precisely, they prove that n non-overlapping algebraic curves of bounded degree d can be cut into $O(n^{3/2} \log^{O(1)} n)$ Jordan arcs so that each pair of arcs intersect in at most one point. Their procedure exploits Proposition 1 for algebraic curves in three dimensions; see [18, Theorem 1.1]. Theorem 4 can be used to prove a slightly weaker constructive and efficient variant of the above result. Specifically, we have:

Theorem 8. Let $\mathcal{C}$ be a set of n algebraic plane curves, each of degree at most d, with no two sharing a common component. Then $\mathcal{C}$ can be cut into $O(n^{3/2+C_d/\log\log n})$ Jordan arcs, where C_d is a constant that depends on d, so that each pair of arcs intersect in at most one point. This cutting can be computed in randomized expected time $O(n^{3/2+C_d/\log\log n})$.

Proof Sketch. The proof of Theorem 1.1 in [18] is completely algorithmic, except for the application of Guth's partitioning for lifted curves in $\mathbb{R}^3$. In [18], Guth's partitioning for curves is applied using a parameter D that is polynomial in n (the specific value used is $D = n^{1/4}$, though any value $D = n^c$ with $0 < c \le 1/4$ would suffice).

Instead of using a partitioning of degree $D = n^{1/4}$, we will use a partitioning of degree $D = c_d \log n$, where c_d is a sufficiently small constant that depends only on d. Throughout this argument, we will use C_d and c_d to denote large and small constants that depend only on d.

The partitioning step is applied as follows. We begin at stage 0 with $(C_d D^3)^0 = 1$ sets, each of which contains $n/(D^2)^0 = n$ curves from $\mathcal{C}$. At stage k, we apply Theorem 4 to each of the $(C_d D^3)^k$ sets of curves to obtain at most $C_d D^3 (C_d D^3)^k = (C_d D^3)^{k+1}$ sets, each of which contains at most $n/D^{2(k+1)}$ curves from $\mathcal{C}$, in

$$(C_d D^3)^k \cdot O_d((n/D^{2k}) \operatorname{Poly}(D) + e^{\operatorname{Poly}(D)}) = O_d\big(n(C_d D)^k \operatorname{Poly}(D) + (C_d D^3)^k e^{\operatorname{Poly}(D)}\big)$$

expected time. Note that the implicit constant in the $\operatorname{Poly}(D)$ terms are independent of k.

We repeat this process until each set contains $O(1)$ curves from the original collection; this requires $k_0 = O_d(\log n/\log D) \le \log n/(C_d c_d \log\log n)$ repetitions, and $nD^{k_0} \le n^{3/2}$.

The total expected time to perform all of these steps is

$$O_d\big(C_d^{\log n/(C_d c_d \log\log n)} n^{3/2}\,\mathrm{Poly}(c_d \log n) + (C_d(c_d \log n))^{(C_d c_d \log\log n)} e^{\mathrm{Poly}(c_d \log n)}\big)$$
$$= O_d(n^{3/2 + C_d/\log\log n}).$$

A similar analysis shows that the number of Jordan arcs obtained through this cutting procedure is $O_d(n^{3/2+C_d/\log\log n})$. $\square$

By replacing Theorem 1.1 in [18] with our Theorem 8, we obtain effective and efficient version of all of the subsequent results in [18].

4.5 Eliminating depth cycles for triangles in $\mathbb{R}^3$.

Aronov, Miller, and Sharir [5] used Proposition 1 for lines in three dimensions in order to prove that n pairwise disjoint non-vertical triangles in $\mathbb{R}^3$ can be cut into $O(n^{3/2+\varepsilon})$ pieces that form a *depth order*, for any $\varepsilon > 0$; see the previous work [4] by the last three authors for a brief overview about the problem of eliminating depth cycles among triangles (or just lines) in 3-space, and the notion of depth order. In an earlier result of Aronov and Sharir [6], they showed a bound of $O(n^{3/2}\,\mathrm{polylog}\, n)$ on the number of cuts needed to apply in a collection of n pairwise disjoint non-vertical lines in $\mathbb{R}^3$, in order to eliminate all depth cycles they form. Aside from the ε loss in the exponent of the first bound (and $\mathrm{polylog}\, n$ factor in the latter bound), these bounds are optimal.

The main difficulty in turning the combinatorial bounds in [6, 5] to efficiently constructive is the lack of an effective and efficient version of Proposition 1. The previous work of Aronov *et al.* [4] addressed the case of lines in 3-space, by integrating the three-dimensional space decomposition they developed with the analysis in [6]. They also presented a brief description for the setting of triangles in 3-space and the integration with the analysis in [5]. However, the latter was not given in full detail. Moreover, we note that the space decomposition in [4] is not given merely by a partitioning polynomial, but it also consists of a semi-algebraic set, due to which the integration with the analysis in [5] is cumbersome and even tedious.[6]

Equipped with Theorem 4 we revisit the setting of triangles in 3-space, and obtain an efficient algorithm to cut n given triangles, as above, into $O(n^{3/2+\varepsilon})$ pieces that form a depth order. Since our space decomposition (summarized in Theorem 4) produces an actual partitioning polynomial (with similar properties as in Proposition 1), the integration with the analysis in [5] is rather simple.

Let $\mathcal{T}$ be a set of n pairwise disjoint non-vertical triangles in $\mathbb{R}^3$. The analysis in [4] recursively subdivides space, by applying a polynomial partitioning of constant degree $D > 1$ for the $3n$ lines supporting the edges of the triangles in $\mathcal{T}$, where at every step in the recursion one draws curves on each triangle. These curves mark the cuts to be applied on the triangles. A close inspection of the analysis in [5] shows that the total number of pieces is in fact proportional to the number of these "cutting" curves up to a polylogarithmic factor. Moreover, the time to produce the triangle pieces once these curves are given is also proportional to the number of curves (up to a polylogarithmic factor). Therefore we focus on the construction of the cutting curves.

Let f be a partitioning polynomial of degree $D > 1$ constructed at the current recursive step. A cutting curve drawn on a triangle $\Delta \in \mathcal{T}$ is one of three types:[7]

[6]The analysis in [6], addressing the case of lines in 3-space, is somewhat simpler, and the work in [4] describes in detail how to obtain a constructive bound in this case.

[7]In fact, the triangles Δ under consideration belong to a subset of $\mathcal{T}$ determined by their interaction with the cells in the partition; we omit these details in this discussion.

(i) *Trace*: We draw $Z(f) \cap \Delta$ on Δ, unless $\Delta \subset Z(f)$. The underlying curve has degree at most D. Regarding the complexity of this operations, we first need to determine whether $\Delta \subset Z(f)$, which can be done by computing *resultants* [9, Chapter 4]. If $\Delta \not\subset Z(f)$ we represent the curve $\Delta \cap Z(f)$ by restricting f to the plane of Δ and then computing the intersection of this curve with $\partial\Delta$. This latter operation exploits *root representation* [9, Chapter 10]. Overall, all these operations can be performed in $O_D(1)$ time [9], where $O_D(\cdot)$ implies that the constant of proportionality depends on D.
(ii) *Critical shadows*: Let S be the common zero set of f and its z-derivative, that is, this is the set of singular points or points of z-vertical tangency of $Z(f)$. Let H be the "vertical curtain" spanned by S, that is, this is the union of all vertical lines that pass through points of S. As pointed out in [5], H is a two-dimensional algebraic variety of degree $O(D^2)$. We next draw on Δ the curve $H \cap \Delta$. Computing this curve is done by first constructing the solution set S of the system $\{f = 0, \partial f/\partial z = 0\}$ in $O_D(1)$ time, using properties of resultants and root representation [9], as above. Then the restriction to Δ (recall that Δ is non-vertical) is done by projecting S onto the plane containing Δ, and then intersecting this projection with $\partial\Delta$. Applying similar considerations as above, this can be done in $O_D(1)$ time as well.
(iii) *Wall shadows*: This step involves the construction of the *vertical decomposition* of the zero set of $k := O(\log_D n)$ polynomials f, each of degree at most D.[8] We denote by F the product of these polynomials, and by $\mathcal{V} := \mathcal{V}(Z(F))$ their vertical decomposititon. Given a triangle Δ under consideration, and an open three-dimensional cell ν of $\mathcal{V}$ meeting Δ, we draw the one-dimensional boundary of $\nu \cap \Delta$ on Δ. We apply this curve drawing also in the case where $\Delta \subset Z(F)$ is part of the floor or ceiling of ν.

As pointed out in [5], the construction of $\mathcal{V}$ is based on producing a collection of $O(k^2) = O(\log^2 n)$ curves, each of degree $O(D^2)$, where these curves are obtained by the intersection of pairs of surfaces $Z(f)$, and the loci of singular points and points with z-vertical tangencies on the individual surfaces. Once these curves are produced, they are projected onto the plane, in order to produce a corresponding planar vertical decomposition (which is then lifted in the z-direction). Omitting any further details, the construction of $\mathcal{V}$ can be completed in $O_D(\text{polylog}\, n)$ time. The analysis in [5] shows that on each triangle Δ we draw $O(D^4 \log^2 n)$ curves overall. Note that by construction each cell of $\mathcal{V}$ is a vertical prism, whose floor and ceiling are portions of some $Z(f)$, and its edges are portions of either vertical segments of curves of degree at most $O(D^2)$. Given this property, and applying similar considerations as in cases (i) and (ii), we conclude that in time $O_D(\text{polylog}\, n)$ we can produce all cutting curves (of type (iii)) on Δ.

Recall that the analysis in [4] recursively subdivides space using a partitioning polynomial f of degree D, where, in each cell $\tau \in \mathbb{R}^3 \setminus Z(f)$ the triangles Δ meeting τ are processed in turn. A crucial step of the analysis is that we keep processing in recursion (within each cell τ) only the triangles that "pierce" the cell τ, that is, triangles whose boundary meets τ. The remaining triangles (referred to as "slicing") are not passed down the recursion, and we dispose them as soon as we draw their corresponding cutting curves, as described above.

It thus follows from the above discussion, the analysis in [5], and Theorem 4 that the expected running time to draw all cutting curves over the triangles $\Delta \in \mathcal{T}$ satisfies the recurrence

$$T(n) = O(D^3) \cdot T(cn/D^2) + O_D(n\, \text{polylog}\, n),$$

where $c > 0$ is an absolute constant. This recursive relation is similar to the one shown in [5] for bounding the overall number of triangle pieces obtained in this process, and has a similar asymptotic

[8]We refer the reader to [5] for further details concerning vertical decomposition of arrangements of algebraic varieties.

solution. ESTHER SAYS: I did not address the base case. Should I? Using induction on n, it is easy to verify that $T(n) = O_D(n^{3/2+\varepsilon})$, once we pick a sufficiently large constant $D > 0$; $\varepsilon > 0$ depends on D and can be made arbitrarily small by increasing D, so we can rewrite the bound as $O(n^{3/2+\varepsilon})$. In summary, we have shown:

Theorem 9. Let $\mathcal{T}$ be a collection of n pairwise disjoint triangles in general position in $\mathbb{R}^3$. Then for any prescribed $\varepsilon > 0$, we can cut the triangles in $\mathcal{T}$ into $O(n^{3/2+\varepsilon})$ pieces, bounded by algebraic arcs of constant maximum degree $\delta = \delta(\varepsilon)$, which form a depth order. These pieces can be produced in expected $O(n^{3/2+\varepsilon})$ time.

References

[1] P. Agarwal. Simplex range searching and its variants: A review. *A Journey Through Discrete Mathematics*, pages 1–30, 2017.

[2] P. K. Agarwal, B. Aronov, E. Ezra, and J. Zahl. An efficient algorithm for generalized polynomial partitioning and its applications. *CoRR*, abs/1812.10269, 2018.

[3] P. K. Agarwal, J. Matouvsek, and M. Sharir. On range searching with semialgebraic sets. II. *SIAM J. Comput.*, 42(6):2039–2062, 2013.

[4] B. Aronov, E. Ezra, and J. Zahl. Constructive polynomial partitioning for algebraic curves in r^3 with applications. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2636–2648, 2019.

[5] B. Aronov, E. Y. Miller, and M. Sharir, Eliminating depth cycles among triangles in three dimensions, In *Proc. 28th Annu. ACM-SIAM Sympos. Discr. Alg.*, 2017, 2476–2494. *CoRR*, abs/1607.06136, 2019.

[6] B. Aronov and M. Sharir, Almost tight bounds for eliminating depth cycles in three dimensions, *Discrete Comput. Geom.*, 59 (2018), 725–741.

[7] S. Barone and S. Basu. Refined bounds on the number of connected components of sign conditions on a variety. *Discrete & Computational Geometry*, 47(3):577–597, 2012.

[8] S. Basu. Algorithms in real algebraic geometry: A survey. *CoRR*, abs/1409.1534, 2014.

[9] S. Basu, R. Pollack, and M. F. Roy, *Algorithms in Real Algebraic Geometry*, 2nd edition, *Springer-Verlag*, Berlin Heidelberg, 2006.

[10] B. Chazelle, H. Edelsbrunner, L. J. Guibas, and M. Sharir. A singly exponential stratification scheme for real semi-algebraic varieties and its applications. *Theor. Comput. Sci.*, 84(1):77–105, 1991.

[11] L. Guth. Polynomial partitioning for a set of varieties. *Math. Proc. Camb. Philos. Soc.*, 159(3):459–469, 2015.

[12] L. Guth and N. Katz. On the erdos distinct distance problem in the plane. *Ann. of Math.*, 181:155–190, 2015.

[13] S. Har-Peled. *Geometric Approximation Algorithms*, volume 173. AMS Press, Boston, MA, 2011.

[14] M. C. J. Bochnak and M. Roy. *Géométrie algébrique réelle (Second edition in english: Real Algebraic Geometry)*, volume 12(36). Springer Verlag, Berlin-Heidelberg, 1998.

[15] A. S. A. S. J. Fox, J. Pach and J. Zahl. A semi-algebraic version of zarankiewicz's problem. *J. Eur. Math. Soc.*, 19(6):1785–1810, 2017.

[16] V. Koltun. Almost tight upper bounds for vertical decompositions in four dimensions. *J. ACM*, 51(5):699–730, 2004.

[17] R. P. S. Basu and M. Roy. *Algorithms in Real Algebraic Geometry*, volume 10. Springer Verlag, Berlin-Heidelberg, 2006.

[18] M. Sharir and J. Zahl. Cutting algebraic curves into pseudo-segments and applications. *J. Comb. Theory, Ser. A*, 150:1–35, 2017.

[19] V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl.*, 16:264–280, 1971.

[20] H. Warren. Lower bound for approximation by nonlinear manifolds. *Trans. Amer. Math. Soc.*, 133:167–178, 1968.