Subquadratic Algorithms for Algebraic Generalizations of 3SUM

Luis Barba[1], Jean Cardinal[*2], John Iacono[†3], Stefan Langerman[‡4], Aurélien Ooms[§5], and Noam Solomon[¶6]

1 **Department of Computer Science, ETH Zürich, Zürich, Switzerland**
luis.barba@inf.ethz.ch
2 **Département d'Informatique, Université libre de Bruxelles (ULB), Brussels, Belgium**
jcardin@ulb.ac.be
3 **Department of Computer Science and Engineering, New York University (NYU), New York, NY, USA**
socg2017@johniacono.com
4 **Département d'Informatique, Université libre de Bruxelles (ULB), Brussels, Belgium**
slanger@ulb.ac.be
5 **Département d'Informatique, Université libre de Bruxelles (ULB), Brussels, Belgium**
aureooms@ulb.ac.be
6 **School of Computer Science, Tel Aviv University (TAU), Tel Aviv, Israel**
noam.solom@gmail.com

Abstract

The 3SUM problem asks if an input n-set of real numbers contains a triple whose sum is zero. We consider the 3POL problem, a natural generalization of 3SUM where we replace the sum function by a constant-degree polynomial in three variables. The motivations are threefold. Raz, Sharir, and de Zeeuw gave an $O(n^{11/6})$ upper bound on the number of solutions of trivariate polynomial equations when the solutions are taken from the cartesian product of three n-sets of real numbers. We give algorithms for the corresponding problem of counting such solutions. Grønlund and Pettie recently designed subquadratic algorithms for 3SUM. We generalize their results to 3POL. Finally, we shed light on the General Position Testing (GPT) problem: "Given n points in the plane, do three of them lie on a line?", a key problem in computational geometry.

We prove that there exist bounded-degree algebraic decision trees of depth $O(n^{\frac{12}{7}+\varepsilon})$ that solve 3POL, and that 3POL can be solved in $O(n^2(\log\log n)^{\frac{3}{2}}/(\log n)^{\frac{1}{2}})$ time in the real-RAM model. Among the possible applications of those results, we show how to solve GPT in subquadratic time when the input points lie on $o((\log n)^{\frac{1}{6}}/(\log\log n)^{\frac{1}{2}})$ constant-degree polynomial curves. This constitutes the first step towards closing the major open question of whether GPT can be solved in subquadratic time. To obtain these results, we generalize important tools – such as batch range searching and dominance reporting – to a polynomial setting. We expect these new tools to be useful in other applications.

* Supported by the "Action de Recherche Concertée" (ARC) COPHYMA, convention number 4.110.H.000023.
† Research partially completed while on sabbatical at the Algorithms Research Group of the Département d'Informatique at the Université libre de Bruxelles with support from a Fulbright Research Fellowship, the Fonds de la Recherche Scientifique – FNRS, and NSF grants CNS-1229185, CCF-1319648, and CCF-1533564.
‡ Directeur de recherches du F.R.S.-FNRS.
§ Supported by the Fund for Research Training in Industry and Agriculture (FRIA).
¶ Supported by Grant 892/13 from the Israel Science Foundation.

33rd International Symposium on Computational Geometry (SoCG 2017).
Editors: Boris Aronov and Matthew J. Katz; Article No. 13; pp. 13:1–13:15

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases 3SUM, subquadratic algorithms, general position testing, range searching, dominance reporting, polynomial curves

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.13

1 Introduction

The 3SUM problem is defined as follows: given n distinct real numbers, decide whether any three of them sum to zero. A popular conjecture is that no $O(n^{2-\delta})$-time algorithm for 3SUM exists. This conjecture has been used to show conditional lower bounds for problems in P, notably in computational geometry with problems such as GeomBase, general position [21] and Polygonal Containment [7], and more recently for string problems such as Local Alignment [2] and Jumbled Indexing [5], as well as dynamic versions of graph problems [32, 1], triangle enumeration and Set Disjointness [27]. For this reason, 3SUM is considered one of the key subjects of an emerging theory of complexity-within-P, along with other problems such as all-pairs shortest paths, orthogonal vectors, boolean matrix multiplication, and conjectures such as the Strong Exponential Time Hypothesis [3, 26, 11].

Because fixing two of the numbers a and b in a triple only allows for one solution to the equation $a+b+x=0$, an instance of 3SUM has at most n^2 solution triples. An instance with a matching lower bound is for example the set $\{\frac{1-n}{2},\ldots,\frac{n-1}{2}\}$ (for odd n) with $\frac{3}{4}n^2+\frac{1}{4}$ solution triples. One might be tempted to think that the number of solutions to the problem would lower bound the complexity of algorithms for the decision version of the problem, as it is the case for restricted models of computation [18]. This is a common misconception. Indeed, Grønlund and Pettie [23] recently proved that there exist $\tilde{O}(n^{3/2})$-depth linear decision trees and $o(n^2)$-time real-RAM algorithms for 3SUM.

A natural generalization of the 3SUM problem is to replace the sum function by a constant-degree polynomial in three variables $F\in\mathbb{R}[x,y,z]$ and ask to determine whether there exists any triple (a,b,c) of input numbers such that $F(a,b,c)=0$. We call this new problem the *3POL problem*.

For the particular case $F(x,y,z)=f(x,y)-z$ where $f\in\mathbb{R}[x,y]$ is a constant-degree bivariate polynomial, Elekes and Rónyai [16] show that the number of solutions to the 3POL problem is $o(n^2)$ unless f is *special*. Special for f means that f has one of the two special forms $f(u,v)=h(\varphi(u)+\psi(v))$ or $f(u,v)=h(\varphi(u)\cdot\psi(v))$, where h,φ,ψ are univariate polynomials of constant degree. Elekes and Szabó [17] later generalized this result to a broader range of functions F using a wider definition of specialness. Raz, Sharir and Solymosi [37] and Raz, Sharir and de Zeeuw [35] recently improved both bounds on the number of solutions to $O(n^{11/6})$. They translated the problem into an incidence problem between points and constant-degree algebraic curves. Then, they showed that unless f (or F) is special, these curves have low multiplicities. Finally, they applied a theorem due to Pach and Sharir [31] bounding the number of incidences between the points and the curves. Some of these ideas appear in our approach.

In computational geometry, it is customary to assume the real-RAM model can be extended to allow the computation of roots of constant degree polynomials. We distance ourselves from this practice and take particular care of using the real-RAM model and the bounded-degree algebraic decision tree model with only the four arithmetic operators.

1.1 Our results

We focus on the computational complexity of 3POL. Since 3POL contains 3SUM, an interesting question is whether a generalization of Grønlund and Pettie's 3SUM algorithm exists for 3POL. If this is true, then we might wonder whether we can beat the $O(n^{11/6}) = O(n^{1.833\ldots})$ combinatorial bound of Raz, Sharir and de Zeeuw [35] with nonuniform algorithms. We give a positive answer to both questions: we show there exist $O(n^2(\log\log n)^{\frac{3}{2}}/\log n)^{\frac{1}{2}})$-time real-RAM algorithms and $O(n^{12/7+\varepsilon}) = O(n^{1.7143})$-depth bounded-degree algebraic decision trees for 3POL.[1] To prove our main result, we present a fast algorithm for the Polynomial Dominance Reporting (PDR) problem, a far reaching generalization of the Dominance Reporting problem. As the algorithm for Dominance Reporting and its analysis by Chan [13] is used in fast algorithms for all-pairs shortest paths, (min,+)-convolutions, and 3SUM, we expect this new algorithm will have more applications.

Our results can be applied to many degeneracy testing problems, such as the General Position Testing (GPT) problem: "Given n points in the plane, do three of them lie on a line?" It is well known that GPT is 3SUM-hard, and it is open whether GPT admits a subquadratic algorithm. Raz, Sharir and de Zeeuw [35] give a combinatorial bound of $O(n^{11/6})$ on the number of collinear triples when the input points are known to be lying on a constant number of polynomial curves, provided those curves are neither lines nor cubic curves. A corollary of our first result is that GPT where the input points are constrained to lie on $o((\log n)^{\frac{1}{6}}/(\log\log n)^{\frac{1}{2}})$ constant-degree polynomial curves (including lines and cubic curves) admits a subquadratic real-RAM algorithm and a strongly subquadratic bounded-degree algebraic decision tree. Interestingly, both reductions from 3SUM to GPT on 3 lines (map a to $(a,0)$, b to $(b,2)$, and c to $(\frac{c}{2},1)$) and from 3SUM to GPT on a cubic curve (map a to (a^3,a), b to (b^3,b), and c to (c^3,c)) construct such special instances of GPT. This constitutes the first step towards closing the major open question of whether GPT can be solved in subquadratic time. This result is described in the arXiv e-print where we also explain how to apply our algorithms to the problems of counting triples of points spanning unit circles or triangles.

1.2 Definitions

3POL. We look at two different generalizations of 3SUM. In the first one, which we call 3POL, we replace the sum function by a trivariate polynomial of constant degree.

▶ **Problem** (3POL). *Let $F \in \mathbb{R}[x,y,z]$ be a trivariate polynomial of constant degree, given three sets A, B, and C, each containing n real numbers, decide whether there exist $a \in A$, $b \in B$, and $c \in C$ such that $F(a,b,c) = 0$.*

The second one is a special case of 3POL where we restrict the trivariate polynomial F to have the form $F(a,b,c) = f(a,b) - c$.

▶ **Problem** (explicit 3POL). *Let $f \in \mathbb{R}[x,y]$ be a bivariate polynomial of constant degree, given three sets A, B, and C, each containing n real numbers, decide whether there exist $a \in A$, $b \in B$, and $c \in C$ such that $c = f(a,b)$.*

We look at both uniform and nonuniform algorithms for explicit 3POL and 3POL. We begin with an $O(n^{\frac{12}{7}+\varepsilon})$-depth bounded-degree algebraic decision tree for explicit 3POL in §2. In

[1] Throughout this document, ε denotes a positive real number that can be made as small as desired.

§3, we continue by giving a similar real-RAM algorithm for explicit 3POL that achieves subquadratic running time. We show how to generalize those results to the implicit version of 3POL in the arXiv e-print.

Models of Computation Similarly to Grønlund and Pettie [23], we consider both nonuniform and uniform models of computation. For the nonuniform model, Grønlund and Pettie consider linear decision trees, where one is only allowed to manipulate the input numbers through linear queries to an oracle. Each linear query has constant cost and all other operations are free but cannot inspect the input. In this paper, we consider *bounded-degree algebraic decision trees (ADT)* [34, 41, 39], a natural generalization of linear decision trees, as the nonuniform model. In a bounded-degree algebraic decision tree, one performs constant cost branching operations that amount to test the sign of a constant-degree polynomial for a constant number of input numbers. Again, operations not involving the input are free. For the uniform model we consider the real-RAM model with only the four arithmetic operators.

The problems we consider require our algorithms to manipulate polynomial expressions and, potentially, their real roots. For that purpose, we will rely on Collins cylindrical algebraic decomposition (CAD) [14]. To understand the power of this method, and why it is useful for us, we give some background on the related concept of first-order theory of the reals.

▶ **Definition 1.** A Tarski formula $\phi \in \mathbb{T}$ is a grammatically correct formula consisting of real variables ($x \in \mathbb{X}$), universal and existential quantifiers on those real variables ($\forall, \exists\colon \mathbb{X} \times \mathbb{T} \to \mathbb{T}$), the boolean operators of conjunction and disjunction ($\wedge, \vee\colon \mathbb{T}^2 \to \mathbb{T}$), the six comparison operators ($<, \leq, =, \geq, >, \neq\colon \mathbb{R}^2 \to \mathbb{T}$), the four arithmetic operators ($+, -, *, /\colon \mathbb{R}^2 \to \mathbb{R}$), the usual parentheses that modify the priority of operators, and constant real numbers. A Tarski sentence is a fully quantified Tarski formula. The first-order theory of the reals ($\forall\exists\mathbb{R}$) is the set of true Tarski sentences.

Tarski [40] and Seidenberg [38] proved that $\forall\exists\mathbb{R}$ is decidable. However, the algorithm resulting from their proof has nonelementary complexity. This proof, as well as other known algorithms, are based on quantifier elimination, that is, the translation of the input formula to a much longer quantifier-free formula, whose validity can be checked. There exists a family of formulas for which any method of quantifier elimination produces a doubly exponential size quantifier-free formula [15]. Collins CAD matches this doubly exponential complexity.

▶ **Theorem 2** (Collins [14]). *$\forall\exists\mathbb{R}$ can be solved in $2^{2^{O(n)}}$ time.*

See Basu, Pollack, and Roy [9] for additional details, Basu, Pollack, and Roy [8] for a singly exponential algorithm when all quantifiers are existential (existential theory of the reals, $\exists\mathbb{R}$), Caviness and Johnson [12] for an anthology of key papers on the subject, and Mishra [29] for a review of techniques to compute with roots of polynomials.

Collins CAD solves any *geometric* decision problem that does not involve quantification over the integers in time doubly exponential in the problem size. This does not harm our results as we exclusively use this algorithm to solve constant size subproblems. Geometric is to be understood in the sense of Descartes and Fermat, that is, the geometry of objects that can be expressed with polynomial equations. In particular, it allows us to make the following computations in the real-RAM and bounded-degree ADT models:

1. Given a constant-degree univariate polynomial, count its real roots in $O(1)$ operations,
2. Given a constant number of univariate polynomials of constant degree, compute the interleaving of their real roots in $O(1)$ operations,
3. Given a point in the plane and an arrangement of a constant number of constant-degree polynomial planar curves, locate the point in the arrangement in $O(1)$ operations.

Instead of bounded-degree algebraic decision trees as the nonuniform model we could consider decision trees in which each decision involves a constant-size instance of the decision problem in the first-order theory of the reals. The depth of a bounded-degree algebraic decision tree simulating such a tree would only be blown up by a constant factor.

1.3 Previous Results

3SUM. For the sake of simplicity, we consider the following definition of 3SUM

▶ **Problem** (3SUM). *Given 3 sets A, B, and C, each containing n real numbers, decide whether there exist $a \in A$, $b \in B$, and $c \in C$ such that $c = a + b$.*

A quadratic lower bound for solving 3SUM holds in a restricted model of computation: the 3-linear decision tree model. Erickson [18] and Ailon and Chazelle [4] showed that in this model, where one is only allowed to test the sign of a linear expression of up to three elements of the input, there are a quadratic number of critical tuples to test.

▶ **Theorem 3** (Erickson [18]). *The depth of a 3-linear decision tree for 3SUM is $\Omega(n^2)$.*

While no evidence suggested that this lower bound could be extended to other models of computation, it was eventually conjectured that 3SUM requires $\Omega(n^2)$ time.

Baran et al. [6] were the first to give concrete evidence for doubting the conjecture. They gave subquadratic Las Vegas algorithms for 3SUM, where input numbers are restricted to be integer or rational, in the circuit RAM, word RAM, external memory, and cache-oblivious models of computation. Their idea is to exploit the parallelism of the models, using linear and universal hashing.

Grønlund and Pettie [23], using a trick due to Fredman [19], recently showed that there exist subquadratic decision trees for 3SUM when the queries are allowed to be 4-linear.

▶ **Theorem 4** (Grønlund and Pettie [23]). *There is a 4-linear decision tree of depth $O(n^{\frac{3}{2}}\sqrt{\log n})$ for 3SUM.*

They also gave deterministic and randomized subquadratic real-RAM algorithms for 3SUM, refuting the conjecture. Similarly to the subquadratic 4-linear decision trees, these new results use the power of 4-linear queries. These algorithms were later improved by Freund [20] and Gold and Sharir [22].

▶ **Theorem 5** (Grønlund and Pettie [23]). *There is a deterministic $O(n^2(\log\log n)^{2/3}/(\log n)^{2/3})$-time and a randomized $O(n^2(\log\log n)^2/\log n)$-time real-RAM algorithm for 3SUM.*

Since then, the conjecture was eventually updated. This new conjecture is considered an essential part of the theory of complexity-within-P.

▶ **Conjecture 1** (3SUM Conjecture). *There is no $O(n^{2-\delta})$-time algorithm for 3SUM.*

Elekes-Rónyai, Elekes-Szabó. In a series of results spanning fifteen years, Elekes and Rónyai [16], Elekes and Szabó [17], Raz, Sharir and Solymosi [37], and Raz, Sharir and de Zeeuw [35] give upper bounds on the number of solution triples to the 3POL problem. The last and strongest result is the following

▶ **Theorem 6** (Raz, Sharir and de Zeeuw [35]). *Let A, B, C be n-sets of real numbers and $F \in \mathbb{R}[x,y,z]$ be a polynomial of constant degree, then the number of triples $(a,b,c) \in A \times B \times C$ such that $F(a,b,c) = 0$ is $O(n^{11/6})$ unless F has some group related form.*[2]

Raz, Sharir and de Zeeuw [35] also look at the number of solution triples for the General Position Testing problem when the input is restricted to points lying on a constant number of constant-degree algebraic curves.

▶ **Theorem 7** (Raz, Sharir and de Zeeuw [35]). *Let C_1, C_2, C_3 be three (not necessarily distinct) irreducible algebraic curves of degree at most d in $\mathbb{C}^2$, and let $S_1 \subset C_1, S_2 \subset C_2, S_3 \subset C_3$ be finite subsets. Then the number of proper collinear triples in $S_1 \times S_2 \times S_3$ is*

$$O_d(|S_1|^{1/2}|S_2|^{2/3}|S_3|^{2/3} + |S_1|^{1/2}(|S_1|^{1/2} + |S_2| + |S_3|)),$$

unless $C_1 \cup C_2 \cup C_3$ is a line or a cubic curve.

Recently, Nassajian Mojarrad, Pham, Valculescu and de Zeeuw [30] and Raz, Sharir and de Zeeuw [36] proved bounds for versions of the problem where F is a 4-variate polynomial.

2 Nonuniform algorithm for explicit 3POL

We begin with the description of a nonuniform algorithm for explicit 3POL which we use later as a basis for other algorithms. We prove the following:

▶ **Theorem 8.** *There is a bounded-degree ADT of depth $O(n^{\frac{12}{7}+\varepsilon})$ for explicit 3POL.*

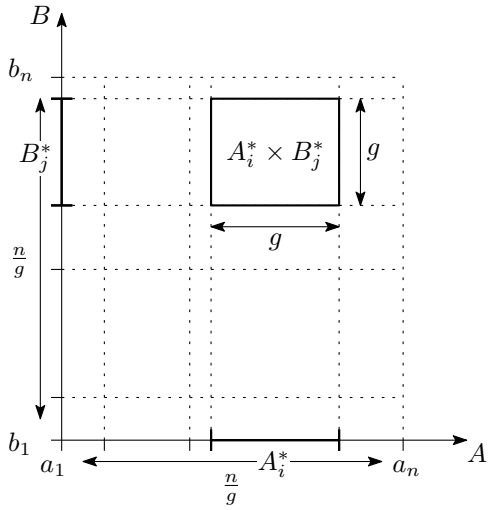
Idea. The idea is to partition the sets A and B into small groups of consecutive elements. That way, we can divide the $A \times B$ grid into cells with the guarantee that each curve $c = f(x,y)$ in this grid intersects a small number of cells. For each such curve and each cell it intersects, we search c among the values $f(a,b)$ for all (a,b) in a given intersected cell. We generalize Fredman's trick [19] – and how it is used in Grønlund and Pettie's paper [23] – to quickly obtain a sorted order on those values, which provides us a logarithmic search time for each cell. Below is a sketch of the algorithm.

Algorithm 1 (Nonuniform algorithm for explicit 3POL).

input $A = \{a_1 < \cdots < a_n\}, B = \{b_1 < \cdots < b_n\}, C = \{c_1 < \cdots < c_n\} \subset \mathbb{R}$.
output *accept* if $\exists\,(a,b,c) \in A \times B \times C$ such that $c = f(a,b)$, *reject* otherwise.
1. Partition the intervals $[a_1, a_n]$ and $[b_1, b_n]$ into blocks A_i^* and B_j^* such that $A_i = A \cap A_i^*$ and $B_j = B \cap B_j^*$ have size g.
2. Sort the sets $f(A_i \times B_j) = \{f(a,b) \colon (a,b) \in A_i \times B_j\}$ for all A_i, B_j. This is the only step that is nonuniform.
3. For each $c \in C$,
3.1. For each cell $A_i^* \times B_j^*$ intersected by the curve $c = f(x,y)$,
3.1.1. Binary search for c in the sorted set $f(A_i \times B_j)$. If c is found, *accept* and halt.
4. *reject* and halt.

Note that it is easy to modify the algorithm to count or report the solutions. In the latter case, the algorithm becomes output sensitive. Like in Grønlund and Pettie's $\tilde{O}(n^{\frac{3}{2}})$ decision tree for 3SUM [23], the tricky part is to give an efficient implementation of step **2**.

[2] Because our results do not depend on the meaning of *group related form*, we do not bother defining it here. We refer the reader to Raz, Sharir and de Zeeuw [35] for the exact definition.



(a) Partitioning A and B.

(b) An xy-monotone arc of $c = f(x, y)$ intersects a staircase of at most $2\frac{n}{g} - 1$ cells in the grid.

Figure 1 Properties of the $A \times B$ grid.

***A* × *B* grid partitioning.** Let $A = \{ a_1 < a_2 < \cdots < a_n \}$ and $B = \{ b_1 < b_2 < \cdots < b_n \}$. For some positive integer g to be determined later, partition the interval $[a_1, a_n]$ into n/g blocks $A_1^*, A_2^*, \ldots, A_{n/g}^*$ such that each block contains g numbers in A. Do the same for the interval $[b_1, b_n]$ with the numbers in B and name the blocks of this partition $B_1^*, B_2^*, \ldots, B_{n/g}^*$. For the sake of simplicity, and without loss of generality, we assume here that g divides n. We continue to make this assumption in the following sections. To each of the $(n/g)^2$ pairs of blocks A_i^* and B_j^* corresponds a cell $A_i^* \times B_j^*$. By definition, each cell contains g^2 pairs in $A \times B$. For the sake of notation, we define $A_i = A \cap A_i^* = \{ a_{i,1} < a_{i,2} < \cdots < a_{i,g} \}$ and $B_j = B \cap B_j^* = \{ b_{j,1} < b_{j,2} < \cdots < b_{j,g} \}$. Figure 1a depicts this construction.

The following two lemmas result from this construction:

▶ **Lemma 9.** *For a fixed value $c \in C$, the curve $c = f(x, y)$ intersects $O(\frac{n}{g})$ cells. Moreover, those cells can be found in $O(\frac{n}{g})$ time.*

Proof. Since f has constant degree, the curve $c = f(x, y)$ can be decomposed into a constant number of xy-monotone arcs. Split the curve into x-monotone pieces, then each x-monotone piece into y-monotone arcs. The endpoints of the xy-monotone arcs are the intersections of $f(x, y) = c$ with its derivatives $f'_x(x, y) = 0$ and $f'_y(x, y) = 0$. By Bézout's theorem, there are $O(\deg(f)^2)$ such intersections and so $O(\deg(f)^2)$ xy-monotone arcs. Figure 1b shows that each such arc intersects at most $2\frac{n}{g} - 1$ cells since the cells intersected by a xy-monotone arc form a staircase in the grid. This proves the first part of the lemma. To prove the second part, notice that for each connected component of $c = f(x, y)$ intersecting at least one cell of the grid either: (1) it intersects a boundary cell of the grid, or (2) it is a (singular) point or contains vertical and horizontal tangency points. The cells intersected by $c = f(x, y)$ are computed by exploring the grid from $O(\frac{n}{g})$ starting cells. Start with an empty set. Find and add all boundary cells containing a point of the curve. Finding those cells is achieved by solving the Tarski sentence $\exists (x, y) c = f(x, y) \wedge x \in A_i^* \wedge y \in B_j^*$, for each cell $A_i^* \times B_j^*$ on the boundary. This takes $O(\frac{n}{g})$ time. Find and add the cells containing endpoints of xy-monotone arcs of $c = f(x, y)$. Finding those cells is achieved by first finding the constant

number of vertical and horizontal slabs $A_i^* \times \mathbb{R}$ and $\mathbb{R} \times B_j^*$ containing such points:

$$\exists(x,y) c = f(x,y) \wedge (f'_x(x,y) = 0 \vee f'_y(x,y) = 0) \wedge x \in A_i^*,$$
$$\exists(x,y) c = f(x,y) \wedge (f'_x(x,y) = 0 \vee f'_y(x,y) = 0) \wedge y \in B_j^*.$$

This takes $O(\frac{n}{g})$ time. Then for each pair of vertical and horizontal slab containing such a point, check that the cell at the intersection of the slab also contains such a point:

$$\exists(x,y) c = f(x,y) \wedge (f'_x(x,y) = 0 \vee f'_y(x,y) = 0) \wedge x \in A_i^* \wedge y \in B_j^*.$$

This takes $O(1)$ time. Note that we can always assume the constant-degree polynomials we manipulate are square-free, as making them square-free is trivial [42]: since $\mathbb{R}[x]$ and $\mathbb{R}[y]$ are unique factorization domains, let $Q = P/\gcd(P, P'_x; x)$ and $\mathrm{sf}(P) = Q/\gcd(P, P'_y; y)$, where $\gcd(P, Q; z)$ is the greatest common divisor of P and Q when viewed as polynomials in $R[z]$ where R is a unique factorization domain and $\mathrm{sf}(P)$ is the square-free part of P. The set now contains, for each component of each type, at least one cell intersected by it. Initialize a list with the elements of the set. While the list is not empty, remove any cell from the list, add each of the eight neighbouring cells to the set and the list, if it contains a point of $c = f(x,y)$ – this can be checked with the same sentences as in the boundary case – and if it is not already in the set. This costs $O(1)$ per cell intersected. The set now contains all cells of the grid intersected by $c = f(x,y)$. ◀

▶ **Lemma 10.** *If the sets A, B, C can be preprocessed in $S_g(n)$ time so that, for any given cell $A_i^* \times B_j^*$ and any given $c \in C$, testing whether $c \in f(A_i \times B_j) = \{ f(a,b) \colon (a,b) \in A_i \times B_j \}$ can be done in $O(\log g)$ time, then, explicit 3POL can be solved in $S_g(n) + O(\frac{n^2}{g} \log g)$ time.*

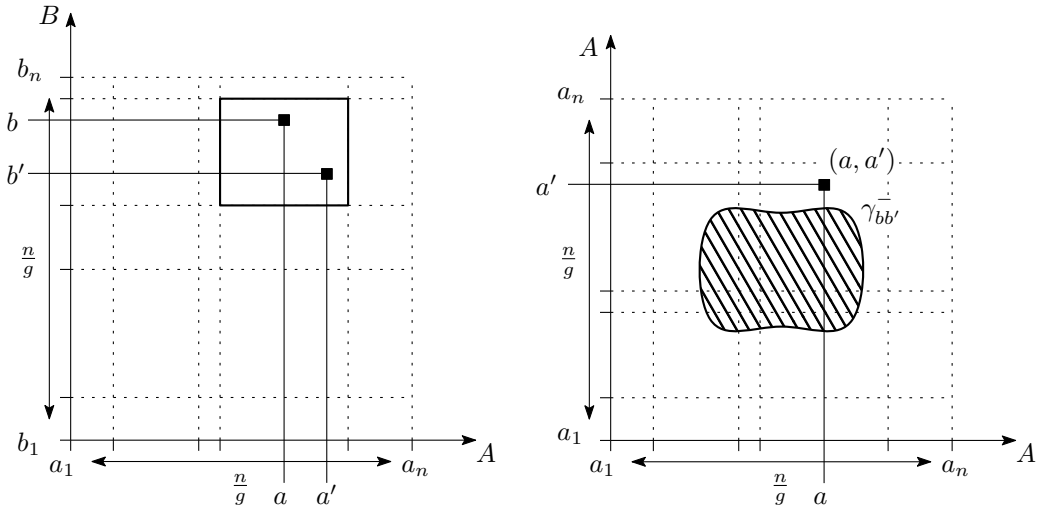
Proof. We need $S_g(n)$ preprocessing time plus the time required to search each of the n numbers $c \in C$ in each of the $O(\frac{n}{g})$ cells intersected by $c = f(x,y)$. Each search costs $O(\log g)$ time. We can compute the cells intersected by $c = f(x,y)$ in $O(\frac{n}{g})$ time by Lemma 9. ◀

▶ Remark. We do not give a $S_g(n)$-time real-RAM algorithm for preprocessing the input, but only a $S_g(n)$-depth bounded-degree ADT. In fact, this preprocessing step is the only nonuniform part of Algorithm 1. A real-RAM implementation of this step is given in §3.

Preprocessing. All that is left to prove is that $S_g(n)$ is subquadratic for some choice of g. To achieve this we sort the points inside each cell using Fredman's trick [19]. Grønlund and Pettie [23] use this trick to sort the sets $A_i + B_j = \{ a + b \colon (a,b) \in A_i \times B_j \}$ with few comparisons: sort the set $D = (\cup_i [A_i - A_i]) \cup (\cup_j [B_j - B_j])$, where $A_i - A_i = \{ a - a' \colon (a,a') \in A_i \times A_i \}$ and $B_j - B_j = \{ b - b' \colon (b,b') \in B_j \times B_j \}$, using $O(n \log n + |D|)$ comparisons, then testing whether $a + b \le a' + b'$ can be done using the free (already computed) comparison $a - a' \le b' - b$. We use a generalization of this trick to sort the sets $f(A_i \times B_j)$. For each B_j, for each pair $(b,b') \in B_j \times B_j$, define the curve $\gamma_{b,b'} = \{ (x,y) \colon f(x,b) = f(y,b') \}$. Define the sets $\gamma^0_{b,b'} = \gamma_{b,b'}, \gamma^-_{b,b'} = \{ (x,y) \colon f(x,b) < f(y,b') \}, \gamma^+_{b,b'} = \{ (x,y) \colon f(x,b) > f(y,b') \}$. The following lemma – illustrated by Figure 2 – follows by definition:

▶ **Lemma 11.** *Given a cell $A_i^* \times B_j^*$ and two pairs $(a,b), (a',b') \in A_i \times B_j$, deciding whether $f(a,b) < f(a',b')$ (respectively $f(a,b) = f(a',b')$ and $f(a,b) > f(a',b')$) amounts to deciding whether the point (a,a') is contained in $\gamma^-_{b,b'}$ (respectively $\gamma^0_{b,b'}$ and $\gamma^+_{b,b'}$).*

There are $N \coloneqq \frac{n}{g} \cdot g^2 = ng$ pairs $(a,a') \in \cup_i [A_i \times A_i]$ and there are N pairs $(b,b') \in \cup_j [B_j \times B_j]$. Sorting the $f(A_i \times B_j)$ for all (A_i, B_j) amounts to solving the following problem:



(a) The pairs $(a, b), (a', b')$.

(b) $(a, a') \in \gamma^+_{b,b'}$ implies $f(a, b) > f(a', b')$.

Figure 2 Generalization of Fredman's trick (Lemma 11).

▶ **Problem** (Polynomial Batch Range Searching). *Given N points and N polynomial curves in $\mathbb{R}^2$, locate each point with respect to each curve.*

We can now refine the description of step **2** in Algorithm 1

Algorithm 2 (Sorting the $f(A_i \times B_j)$ with a nonuniform algorithm).

input $A = \{ a_1 < a_2 < \cdots < a_n \}, B = \{ b_1 < b_2 < \cdots < b_n \} \subset \mathbb{R}$
output The sets $f(A_i \times B_j)$, sorted.
2.1. Locate each point $(a, a') \in \cup_i [A_i \times A_i]$ w.r.t. each curve $\gamma_{b,b'}, (b, b') \in \cup_j [B_j \times B_j]$.
2.2. Sort the sets $f(A_i \times B_j)$ using the information retrieved in step **2.1**.

Note that this algorithm is nonuniform: step **2.2** costs at least quadratic time in the real-RAM model, however, this step does not need to query the input at all, as all the information needed to sort is retrieved during step **2.1**. Step **2.2** incurs no cost in our nonuniform model.

To implement step **2.1**, we use a modified version of the $N^{\frac{4}{3}} 2^{O(\log^* N)}$ algorithm of Matoušek [28] for Hopcroft's problem. We prove the following upper bound in the arXiv e-print:

▶ **Lemma 12.** *Polynomial Batch Range Searching can be solved in $O(N^{\frac{4}{3}+\varepsilon})$ time in the real-RAM model when the input curves are the $\gamma_{b,b'}$.*

Analysis. Combining Lemma 10 and Lemma 12 yields a $O((ng)^{4/3+\varepsilon} + n^2 g^{-1} \log g)$-depth bounded-degree ADT for 3POL. By optimizing over g, we get $g = \Theta(n^{2/7-\varepsilon})$, and the previous expression simplifies to $O(n^{12/7+\varepsilon})$, proving Theorem 8.

3 Uniform algorithm for explicit 3POL

We now build on the first algorithm and prove the following:

▶ **Theorem 13.** *Explicit 3POL can be solved in $O(n^2 (\log \log n)^{\frac{3}{2}} / (\log n)^{\frac{1}{2}})$ time.*

We generalize again Grønlund and Pettie [23]. The algorithm we present is derived from the first subquadratic algorithm in their paper.

Idea. We want the implementation of step **2** in Algorithm 1 to be uniform, because then, the whole algorithm is. We use the same partitioning scheme as before except we choose g to be much smaller. This allows to store all permutations on g^2 items in a lookup table, where g is chosen small enough to make the size of the lookup table $\Theta(n^\varepsilon)$. The preprocessing part of the previous algorithm is replaced by $g^2!$ calls to an algorithm that determines for which cells a given permutation gives the correct sorted order. This preprocessing step stores a constant-size[3] pointer from each cell to the corresponding permutation in the lookup table. Search can now be done efficiently: when searching a value c in $f(A_i \times B_j)$, retrieve the corresponding permutation on g^2 items from the lookup table, then perform binary search on the sorted order defined by that permutation. The sketch of the algorithm is exactly Algorithm 1. The only differences with respect to §2 are the choice of g and the implementation of step **2**.

***A* × *B* grid partitioning.** We use the same partitioning scheme as before, hence Lemma 9 and Lemma 10 hold. We just need to find a replacement for Lemma 12.

Preprocessing. For their simple subquadratic 3SUM algorithm, Grønlund and Pettie [23] explain that for a permutation to give the correct sorted order for a cell, that permutation must define a *certificate* – a set of inequalities – that the cell must verify. They cleverly note – using Fredman's Trick [19] as in Chan [13] and Bremner et al. [10] – that the verification of a single certificate by all cells amounts to solving a red/blue point dominance reporting problem. We generalize their method. For each permutation $\pi\colon [g^2] \to [g]^2$, where $\pi = (\pi_r, \pi_c)$ is decomposed into row and column functions $\pi_r, \pi_c\colon [g^2] \to [g]$, we enumerate all cells $A_i^* \times B_j^*$ for which the following *certificate* holds:

$$f(a_{i,\pi_r(1)}, b_{j,\pi_c(1)}) \leq f(a_{i,\pi_r(2)}, b_{j,\pi_c(2)}) \leq \cdots \leq f(a_{i,\pi_r(g^2)}, b_{j,\pi_c(g^2)}).$$

▶ Remark. Since some entries may be equal, to make sure each cell corresponds to exactly one certificate, we replace $\leq$ symbols by choices of $g^2 - 1$ symbols in $\{=, <\}$. Each permutation π gets a certificate for each of those choices. This adds a 2^{g^2-1} factor to the number of certificates to test, which will eventually be negligible. Note that some of those 2^{g^2-1} certificates are equivalent. We need to skip some of them, as otherwise we might output some cells more than once, and then there will be no guarantee with respect to the output size. For example, the certificate $f(a_{i,9}, b_{j,5}) = f(a_{i,6}, b_{j,7}) < \cdots < f(a_{i,4}, b_{j,4})$ is equivalent to the certificate $f(a_{i,6}, b_{j,7}) = f(a_{i,9}, b_{j,5}) < \cdots < f(a_{i,4}, b_{j,4})$. Among equivalent certificates, we only consider the certificate whose permutation π precedes the others lexicographically. In the previous example, $((6,7),(9,5),\ldots,(4,4)) \prec ((9,5),(6,7),\ldots,(4,4))$ hence we would only process the second certificate. For the sake of simplicity, we will write inequality when we mean strict inequality or equation, and "$\leq$" when we mean "$<$" or "$=$".

Fredman's Trick. This is where Fredman's Trick comes into play. By Lemma 11, each inequality $f(a_{i,\pi_r(t)}, b_{j,\pi_c(t)}) \leq f(a_{i,\pi_r(t+1)}, b_{j,\pi_c(t+1)})$ of a certificate can be checked by computing the relative position of $(a_{i,\pi_r(t)}, a_{i,\pi_r(t+1)})$ with respect to $\gamma_{b_{j,\pi_c(t)}, b_{j,\pi_c(t+1)}}$. For a given certificate, for each A_i and each B_j, define

$$p_i = ((a_{i,\pi_r(1)}, a_{i,\pi_r(2)}), (a_{i,\pi_r(2)}, a_{i,\pi_r(3)}), \ldots, (a_{i,\pi_r(g^2-1)}, a_{i,\pi_r(g^2)})),$$
$$q_j = \big(f(x, b_{j,\pi_c(1)}) \leq f(y, b_{j,\pi_c(2)}), \ldots, f(x, b_{j,\pi_c(g^2-1)}) \leq f(y, b_{j,\pi_c(g^2)})\big).$$

[3] In the real-RAM and word-RAM models.

A certificate is verified by a cell $A_i \times B_j$ if and only if, for all $t \in [g^2-1]$, the point $p_{i,t}$ verifies the inequality $q_{j,t}$. Enumerating all cells $A_i \times B_j$ for which the certificate holds therefore amounts to solving the following problem:

▶ **Problem** (Polynomial Dominance Reporting (PDR)). *Given N k-tuples p_i of points in $\mathbb{R}^2$ and N k-tuples q_j of bivariate polynomial inequalities of degree at most $\deg(f)$, enumerate all pairs (p_i, q_j) where, for all $t \in [k]$, the point $p_{i,t}$ verifies the inequality $q_{j,t}$.*

In the next section, we explain how to solve PDR efficiently and prove the following lemma:

▶ **Lemma 14.** *We can enumerate all ℓ such pairs in time $2^{O(k)} N^{2-\frac{4}{\deg(f)^2+3\deg(f)+2}+\varepsilon} + O(\ell)$.*

We can now give a uniform implementation of step **2** in Algorithm 1:

Algorithm 3 (Sorting the $f(A_i \times B_j)$ with a uniform algorithm).

input $A = \{ a_1 < a_2 < \cdots < a_n \}, B = \{ b_1 < b_2 < \cdots < b_n \} \subset \mathbb{R}$
output The sets $f(A_i \times B_j)$, sorted.
2.1. Initialize a lookup table that will contain all $O(2^{g^2-1}(g^2!))$ certificates on g^2 elements.
2.2. For each permutation $\pi\colon [g^2] \to [g]^2$,
2.2.1. For each choice of g^2-1 symbols in $\{=, <\}$,
2.2.1.1. If there is any "=" symbol that corresponds to a lexicographically decreasing pair of tuples of indices in π, skip this choice of symbols.
2.2.1.2. Append the certificate associated to Π and the choice of symbols to the table.
2.2.1.3. Solve the PDR instance associated to A, B, Π and the choice of symbols.
2.2.1.4. For each output pair (i, j), store a pointer from (i, j) to the last entry in the table.

Analysis. Plugging in $k = g^2 - 1$, $N = \frac{n}{g}$, iterating over all permutations ($\sum_\pi \ell = (n/g)^2$), and adding the binary search step we get that explicit 3POL can be solved in time

$$(g^2!)2^{g^2}2^{O(g^2)}(n/g)^{2-\frac{4}{\deg(f)^2+3\deg(f)+2}+\varepsilon} + O((n/g)^2) + O(n^2 \log g/g).$$

The first two terms correspond to the complexity of step **2** in Algorithm 1, and the last term corresponds to the complexity of step **3** in Algorithm 1. To get subquadratic time we can set $g = c_{\deg(f)}\sqrt{\log n / \log\log n}$, because then for some appropriate choice of the constant factor $c_{\deg(f)}$, $(g^2)!2^{g^2}2^{O(g^2)} = n^\delta$ where $\delta < 4/(\deg(f)^2 + 3\deg(f) + 2) - \varepsilon$, making the first term negligible. The complexity of the algorithm is dominated by $O(n^2 \log g/g) = O(n^2(\log\log n)^{\frac{3}{2}}/(\log n)^{\frac{1}{2}})$. This proves Theorem 13.

4 Polynomial Dominance Reporting

In this section, we combine a standard dominance reporting algorithm [33] with Matoušek's algorithm [28] to prove Lemma 14. We say a pair of blue and red points in $\mathbb{R}^k$ is dominating if for all indices $i \in [k]$ the ith coordinate of the blue point is greater or equal to the ith coordinate of the red point. The standard algorithm [33] solves the following problem:

▶ **Problem.** *Given N blue and M red points in $\mathbb{R}^k$, report all bichromatic dominating pairs.*

Our problem is significantly more complicated and general. Instead of blue points we have blue k-tuples p_i of 2-dimensional points, instead of red points we have red k-tuples q_j of bivariate polynomial inequalities, and we want to report all bichromatic pairs (p_i, q_j) such that, for all $t \in [k]$, the point $p_{i,t}$ verifies the inequality $q_{j,t}$. The standard algorithm essentially works by

a combination of divide and conquer and prune and search, using a one-dimensional cutting (median selection) to split a problem into subproblems. We generalize the standard algorithm by using higher dimensional cuttings, in a way similar to Matoušek's algorithm [28]. For the analysis, we generalize Chan's analysis of the standard algorithm when k is not constant [13].

Proof of Lemma 14. We use the Veronese embedding [25, 24]. Since the polynomials have constant degree, we can trade polynomial inequalities for linear inequalities by lifting everything to a space of higher – but constant – dimension. The degree of each polynomial is at most $\deg(f)$. There are exactly $d = \binom{\deg(f)+2}{2} - 1$ different bivariate monomials of degree at most $\deg(f)$[4]. To each monomial we associate a variable in $\mathbb{R}^d$. By this association, points in the plane are mapped to points in $\mathbb{R}^d$ and bivariate polynomial inequalities are mapped to d-variate linear inequalities.

By abuse of notation, let p_i denote the tuple p_i where each 2-dimensional point has been replaced by its d-dimensional counterpart, and let q_i denote the tuple q_i where each bivariate polynomial inequality has been replaced by its d-variate linear counterpart. We have N k-tuples p_i and M k-tuples q_j. The algorithm checks each of the k components of the tuples in turn and can be described recursively as follows for some positive integer $r > 1$:

Algorithm 4 (Polynomial Dominance Reporting)**.**

input N k-tuples p_i of d-dimensional points, M k-tuples q_j of d-variate linear inequalities.
output All (p_i, q_j) pairs such that, for all $t \in [k]$, the point $p_{i,t}$ verifies the inequality $q_{j,t}$.

1. If $k = 0$, then output all pairs (p_i, q_j) and halt.
2. If $N < r^d$ or $M < r$, solve the problem by brute force in $O((N + M)k)$ time.
3. We now only consider the kth component of each input k-tuple and call these *active* components. To each active d-variate linear inequality corresponds a defining hyperplane in $\mathbb{R}^d$. Construct, as in [28], a hierarchical cutting of $\mathbb{R}^d$ using $O(r^d)$ simplicial cells such that each simplicial cell is intersected by at most $\frac{M}{r}$ of the defining hyperplanes. This construction also gives us for each simplical cell of the cutting the list of defining hyperplanes intersecting it. This takes $O(Mr^{d-1})$ time. Locate each active point inside the hierarchical cutting in time $O(N \log r)$. Let S be a simplicial cell of the hierarchical cutting. Denote by Π_S the set of active points in S. Partition each Π_S into $\left\lceil \frac{|\Pi_S|}{Nr^{-2}} \right\rceil$ disjoint subsets of size at most $\frac{N}{r^d}$. For each simplicial cell, find the active inequalities whose corresponding geometric object (hyperplane, closed or open half-space) contains the cell. This takes $O(Mr^d)$ time. The whole step takes $O(N \log r + Mr^d)$ time.
4. For each of the $O(r^d)$ simplicial cells, recurse on the at most $\frac{N}{r^d}$ k-tuples p_i whose active point is inside the simplicial cell and the at most $\frac{M}{r}$ k-tuples q_j whose active inequality's defining hyperplane intersects the simplicial cell.
5. For each of the $O(r^d)$ simplicial cells, recurse on the at most $\frac{N}{r^d}$ $(k-1)$-prefixes of k-tuples p_i whose active point is inside the simplicial cell and the $(k-1)$-prefixes of k-tuples q_j whose active inequality's corresponding geometric object contains the simplicial cell.

Correctness. In each recursive call, either k is decremented or M and N are divided by some constant, hence, one of the conditions in steps **1** and **2** is met in each of the paths of the recursion tree and the algorithm always terminates. Step **5** is correct because it only recurses on (p_i, q_j) pairs whose suffix pairs are dominating. The base case in step **1** is correct

[4] Not including the independent monomial, namely, 1.

because the only way for a pair (p_i, q_j) to reach this point is to have had all k components checked in step **5**. The base case in step **2** is correct by definition. Each dominating pair is output exactly once because the recursive calls of step **4** and **5** partition the set of pairs (p_i, q_j) that can still claim to be candidate dominating pairs.

Analysis. For $k, N, M \geq 0$, the total complexity $T_k(N, M)$ of computing the inclusions for the first k components, excluding the output cost (steps **1** and **2**), is bounded by

$$T_k(N,M) \leq \underbrace{O(r^d)\, T_{k-1}(N,M)}_{\text{Step } \mathbf{5}} + \underbrace{O(r^d)\, T_k\left(\frac{N}{r^d}, \frac{M}{r}\right)}_{\text{Step } \mathbf{4}} + \underbrace{O(N+M)}_{\text{Step } \mathbf{3}},$$

$$T_0(N,M) = 0, \quad T_k(N,M) = O(Nk) \text{ if } M < r, \quad T_k(N,M) = O(Mk) \text{ if } N < r^d.$$

By point-hyperplane duality, $T_k(N, M) = T_k(M, N)$, hence, we can execute step **4** on dual linear inequalities and dual points to balance the recurrence. For some constant $c_1 \geq 1$,

$$T_k(N,M) \leq c_1 r^{2d} T_{k-1}(N,M) + c_1 r^{2d} T_k\left(\frac{N}{r^{d+1}}, \frac{M}{r^{d+1}}\right) + c_1(N+M).$$

For simplicity, we ignore some problem-size reductions occuring in this balancing step.

Let $T_k(N) = T_k(N, N)$ denote the complexity of solving the problem when $M = N$, excluding the output cost. Hence,

$$T_k(N) \leq c_1 r^{2d} T_{k-1}(N) + c_1 r^{2d} T_k\left(\frac{N}{r^{d+1}}\right) + c_1 N, \tag{1}$$

$$T_0(N) = 0, \quad T_k(N) = O(k) \text{ if } N < r^{d+1}.$$

Solving the recurrence gives $T_k(N) = 2^{O(k)} N^{\frac{2d}{d+1}+\varepsilon_r}$, and since $d = \binom{\deg(f)+2}{2} - 1$, we have

$$T_k(N) = 2^{O(k)} N^{2-\frac{4}{\deg(f)^2+3\deg(f)+2}+\varepsilon_r}.$$

To that complexity we add a constant time unit for each output pair in steps **1** and **2**. ◀

5 3POL

Extending the previous techniques to work for the (implicit) 3POL problem is nontrivial:

1. Instead of sorting the sets $f(A_i \times B_j)$ we need to sort the real roots of the $F(A_i \times B_j, z)$,
2. The $\gamma_{b,b'}$ curves must be redefined. The redefined curve $\gamma_{b,b'}$ is still the zero-set of some constant-degree bivariate polynomial $P(x, y)$. However, retrieving the information we need for sorting becomes more challenging than just computing the sign of the $P(A_i \times A_i)$,
3. The implementation of the certificates for the uniform algorithm gets much more convoluted: each certificate checks the validity of a conjunction of Tarski sentences.

Those extensions are explained in detail in the arXiv e-print. There we show

▶ **Theorem 15.** *There is a bounded-degree ADT of depth $O(n^{\frac{12}{7}+\varepsilon})$ for 3POL.*

▶ **Theorem 16.** *3POL can be solved in $O(n^2(\log\log n)^{\frac{3}{2}}/(\log n)^{\frac{1}{2}})$ time.*

6 Applications

To illustrate the expressive power of 3POL, we give some geometric applications in the arXiv e-print. We show the following:

1. GPT can be solved in subquadratic time provided the input points lie on few parameterized constant-degree polynomial curves.
2. In the plane, given three sets C_i of n unit circles and three points p_i such that a circle $c \in C_i$ contains p_i, deciding whether there exists $(a, b, c) \in C_1 \times C_2 \times C_3$ such that $a \cap b \cap c \neq \emptyset$ can be done in subquadratic time.
3. Given n input points in the plane, deciding whether any triple spawns a unit triangle can be done in subquadratic time, provided the input points lie on few parameterized constant-degree polynomial curves.

References

1 Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *FOCS*, pages 434–443. IEEE Computer Society, 2014.

2 Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *ICALP (1)*, volume 8572 of *LNCS*, pages 39–51, 2014.

3 Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. Matching triangles and basing hardness on an extremely popular conjecture. In *STOC*, pages 41–50. ACM, 2015.

4 Nir Ailon and Bernard Chazelle. Lower bounds for linear degeneracy testing. *J. ACM*, 52(2):157–171, 2005.

5 Amihood Amir, Timothy M. Chan, Moshe Lewenstein, and Noa Lewenstein. On hardness of jumbled indexing. In *ICALP (1)*, volume 8572 of *LNCS*, pages 114–125, 2014.

6 Ilya Baran, Erik D. Demaine, and Mihai Pătraşcu. Subquadratic algorithms for 3SUM. *Algorithmica*, 50(4):584–596, 2008.

7 Gill Barequet and Sariel Har-Peled. Polygon containment and translational min Hausdorff distance between segment sets are 3SUM-hard. *Int. J. Comput. Geometry Appl.*, 11(4):465–474, 2001.

8 Saugata Basu, Richard Pollack, and Marie-Françoise Roy. Computing roadmaps of semi-algebraic sets (extended abstract). In *STOC*, pages 168–173. ACM, 1996.

9 Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in real algebraic geometry*, volume 10 of *Algorithms and Computation in Mathematics*. Springer, 2006.

10 David Bremner, Timothy M. Chan, Erik D. Demaine, Jeff Erickson, Ferran Hurtado, John Iacono, Stefan Langerman, Mihai Patrascu, and Perouz Taslakian. Necklaces, Convolutions, and X+Y. *Algorithmica*, 69(2):294–314, 2014.

11 Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *ITCS*, pages 261–270. ACM, 2016.

12 Bob F. Caviness and Jeremy R. Johnson. *Quantifier elimination and cylindrical algebraic decomposition*. Springer, 2012.

13 Timothy M. Chan. All-pairs shortest paths with real weights in $O(n^3/\log n)$ time. *Algorithmica*, 50(2):236–243, 2008.

14 George E. Collins. Hauptvortrag: Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Automata Theory and Formal Languages*, volume 33 of *LNCS*, pages 134–183. Springer, 1975.

15 James H. Davenport and Joos Heintz. Real quantifier elimination is doubly exponential. *J. Symb. Comput.*, 5(1/2):29–35, 1988.

16 György Elekes and Lajos Rónyai. A combinatorial problem on polynomials and rational functions. *J. Comb. Theory, Ser. A*, 89(1):1–20, 2000.

17 György Elekes and Endre Szabó. How to find groups? (and how to use them in Erdős geometry?). *Combinatorica*, 32(5):537–571, 2012.
18 Jeff Erickson. Lower bounds for linear satisfiability problems. *Chicago J. Theor. Comput. Sci.*, 1999.
19 Michael L. Fredman. How good is the information theory bound in sorting? *Theor. Comput. Sci.*, 1(4):355–361, 1976.
20 Ari Freund. Improved subquadratic 3SUM. *Algorithmica*, pages 1–19, 2015.
21 Anka Gajentaan and Mark H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom.*, 5:165–185, 1995.
22 Omer Gold and Micha Sharir. Improved bounds for 3SUM, k-SUM, and linear degeneracy. *ArXiv e-prints*, 2015. arXiv:1512.05279 [cs.DS].
23 Allan Grønlund and Seth Pettie. Threesomes, degenerates, and love triangles. In *Foundations of Computer Science (FOCS 2014)*, pages 621–630. IEEE, 2014.
24 Joe Harris. *Algebraic geometry: a first course*, volume 133. Springer, 2013.
25 Robin Hartshorne. *Algebraic geometry*, volume 52. Springer, 1977.
26 Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *STOC*, pages 21–30. ACM, 2015.
27 Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3SUM conjecture. In *SODA*, pages 1272–1287. SIAM, 2016.
28 Jirí Matoušek. Range searching with efficient hierarchical cutting. *Discrete & Computational Geometry*, 10:157–182, 1993.
29 Bhubaneswar Mishra. Computational real algebraic geometry. In *Handbook of Discrete and Computational Geometry, 2nd Ed.*, pages 743–764. Chapman and Hall/CRC, 2004.
30 H. Nassajian Mojarrad, T. Pham, C. Valculescu, and F. de Zeeuw. Schwartz-Zippel bounds for two-dimensional products. *ArXiv e-prints*, 2016. arXiv:1507.08181 [math.CO].
31 János Pach and Micha Sharir. On the number of incidences between points and curves. *Combinatorics, Probability & Computing*, 7(1):121–127, 1998.
32 Mihai Pătraşcu. Towards polynomial lower bounds for dynamic problems. In *STOC*, pages 603–610. ACM, 2010.
33 Franco P. Preparata and Michael Ian Shamos. *Computational Geometry – An Introduction*. Texts and Monographs in Computer Science. Springer, 1985.
34 Michael O. Rabin. Proving simultaneous positivity of linear forms. *J. Comput. Syst. Sci.*, 6(6):639–650, 1972.
35 Orit E. Raz, Micha Sharir, and Frank de Zeeuw. Polynomials vanishing on cartesian products: The Elekes-Szabó theorem revisited. In *SoCG*, volume 34 of *LIPIcs*, pages 522–536, 2015.
36 Orit E. Raz, Micha Sharir, and Frank de Zeeuw. The elekes-szabó theorem in four dimensions. *ArXiv e-prints*, 2016. arXiv:1607.03600 [math.CO].
37 Orit E. Raz, Micha Sharir, and József Solymosi. Polynomials vanishing on grids: The Elekes-Rónyai problem revisited. In *SoCG*, page 251. ACM, 2014.
38 Abraham Seidenberg. Constructions in algebra. *Transactions of the AMS*, 197:273–313, 1974.
39 J. Michael Steele and Andrew Yao. Lower bounds for algebraic decision trees. *J. Algorithms*, 3(1):1–8, 1982.
40 Alfred Tarski. A decision method for elementary algebra and geometry, 1951. Rand Corporation.
41 Andrew Yao. A lower bound to finding convex hulls. *J. ACM*, 28(4):780–787, 1981.
42 David Yun. On square-free decomposition algorithms. In *SYMSACC*, pages 26–35, 1976.