The Complexity of Order-Finding for ROABPs

Vishwas Bhargava[*]   Pranjal Dutta[†]   Sumanta Ghosh[‡]   Anamay Tengse[§]

Abstract

We study the *order-finding problem* for Read-once Oblivious Algebraic Branching Programs (ROABPs). Given a polynomial f and a parameter w, the goal is to find an order σ in which f has an ROABP of *width w*. We show that this problem is NP-hard in the worst case, even when the input is a constant degree polynomial that is given in its dense representation. We provide a reduction from `CutWidth` to prove these results. Owing to the exactness of our reduction, all the known results for the hardness of approximation of `CutWidth` also transfer directly to the order-finding problem. Additionally, we also show that any constant-approximation algorithm for the order-finding problem would imply a polynomial time approximation scheme (PTAS) for it.

On the algorithmic front, we design algorithms that solve the order-finding problem for generic ROABPs in polynomial time, when the width w is polynomial in the individual degree d of the polynomial f. That is, our algorithm is efficient for most/random ROABPs, and requires more time only on a lower-dimensional subspace (or subvariety) of ROABPs. Even when the individual degree is constant, our algorithm runs in time $n^{O(\log w)}$ for most/random ROABPs. This stands in strong contrast to the case of (Boolean) ROBPs, where only heuristic order-finding algorithms are known.

---

[*]vishwas1384@gmail.com. Department of Computing and Mathematical Sciences, Caltech. Part of this work was done as a postdoc at University of Waterloo, Canada.

[†]duttpranjal@gmail.com. School of Computing, NUS.

[‡]besusumanta@gmail.com. Chennai Mathematical Institute.

[§]anamay.tengse@gmail.com. School of Computer Sciences, NISER, Bhubaneswar. Initial parts of this work were done when the author was a postdoc at University of Haifa (supported by ISF grant no 716/20) and Reichman University, Herzliya (supported by ISF grant no 843/23).

1   Introduction

Read-Once Algebraic Branching Programs (ROABPs) are a well-studied and well-understood model in algebraic circuit complexity. There are numerous reasons for studying this model, starting with the fact that ROABPs serve as the algebraic analog of ordered ROBPs: Read-Once Branching Programs, a.k.a. OBDDs. Consequently, Polynomial Identity Testing (PIT) for ROABPs is the algebraic analog of the fundamental RL vs L problem in derandomization of boolean computation.

Formally, an **ROABP** $R(x_1, \ldots, x_n)$ for computing $f(\mathbf{x})$ in order σ is a layered, directed graph with $n + 1$ layers. The source s and sink t are single vertices in the 0th and n^{th} layers, respectively. Edges between layers $i - 1$ and i are labeled by univariate polynomials in $x_{\sigma(i)}$ of degree at most d. The polynomial computed by the ROABP is the sum of the products of edge weights along all paths from s to t.

The structural restriction of being read-once also makes ROABPs the commutative analog of non-commutative algebraic branching programs (ABPs). Despite this restriction, ROABPs can efficiently simulate many interesting algebraic models, such as sparse polynomials, set-multilinear depth-3 circuits, diagonal depth-3 circuits, and polynomials that have polynomially large dimension of partial derivatives (see e.g. [FS13, BT24]). They can also efficiently compute complex polynomials like the Iterated Matrix Multiplication (IMM) polynomial [KNS15], which is provably hard for constant-depth circuits [LST24].

A major motivation for studying ROABPs is the characterization due to Nisan [Nis91] that precisely describes the ROABP size required for a given variable partition. This characterization has led to exponential lower bounds, a polynomial-time white-box PIT, and quasipolynomial-time black-box PITs for ROABPs and their variants (see e.g. [GKST17]). Investigating the structure and properties of ROABPs has also led to significant advances in other areas of algebraic complexity. For instance, PITs for bounded top and bottom fan-in depth-4 circuits and ABP upper-bounds for the border of $\Sigma\Pi\Sigma(k)$ circuits critically rely on the PIT and derandomization results for ROABPs ([DDS21a, DDS21b]).

ROABPs have also proven to be highly useful for designing learning algorithms for various circuit classes, such as depth-3 powering ($\sum \wedge \sum$), set-multilinear depth-3 circuits, and polynomials with polynomially bounded partial derivatives [KS06]. Given an order and black-box access to a polynomial f, we can (proper-)learn it in randomized polynomial time and deterministic quasipolynomial time. However, we currently do not know how to learn ROABPs without additional information about the ordering of variables. Therefore, it is extremely natural to ask whether we can design a learning algorithm that operates without this information of an optimal ordering. This question is the primary focus of our work. We begin by demonstrating the significant role that variable ordering plays in ROABPs.

Fact 1.1 (See e.g. [Raz06, FS13]). *The polynomial $f(x_1, \ldots, x_n, y_1, \ldots, y_n) := \prod_{i \in [n]} (x_i + y_i)$, has an*

ROABP of size $3n + 1$ and width 2 in the order $(x_1, y_1, x_2, y_2, \ldots, x_n, y_n)$, but requires ROABPs of size and width $2^{\Omega(n)}$ in the order $(x_1, \ldots, x_n, y_1, \ldots, y_n)$.

Given how the ROABP-size of a polynomial critically depends on the underlying ordering of the variables, let us formally state the problem of interest.

Question 1.2 (Order finding for ROABP). *Given a polynomial $f(x_1, \ldots, x_n)$ of individual degree d, and a number $w \in \mathbb{N}$, decide if there exists an order $\sigma \in s_n$ in which f has an ROABP of width at most w.*

For concreteness, let us fix that the polynomial is given as either as a black-box, or as a circuit ("white-box" representation). Note that a naïve randomized algorithm for this problem would take $n! \cdot \text{poly}(d, w)$ time by iterating over all the permutations, and checking whether for a particular partition σ, the ROABP width: RO-width$_\sigma(f)$ is at most w. But given the algebraic nature of the problem, and also the aforementioned characterization due to Nisan, it would be unfair to rule out a more tractable algorithm upfront. This brings us to our results.

1.1 Our Results

Somewhat surprisingly, we show that the order-finding problem is indeed NP-hard, by providing a polynomial-time Karp reduction from the `CutWidth` problem (see e.g. [GJ79]) for linear arrangement of graphs.

Theorem 1.3. *Given a polynomial $f(x_1, \ldots, x_n)$ with individual degree d, provided either as a black-box or a circuit, and a number $w \in \mathbb{N}$, deciding whether there exists an order $\sigma \in S_n$ such that f has an ROABP of width at most w is NP-hard.*

An important feature of this reduction is that it provides an *exact* relationship between the cut-width of the input graph and the ROABP-width of the output polynomial. Concretely, we show that for any graph G, we can construct a polynomial f_G such that RO-width$_\sigma(f_G) = 2 + \texttt{CutWidth}_\sigma(G)$.

Another important aspect of our construction is that for a graph of degree Δ, f_G has a total degree bounded by 2Δ. Combined with the known NP-hardness results for cut-width for graphs with maximum degree 3, this shows that the hard instances for ROABP order-finding can even be constant-degree polynomials.

An interesting consequence of our hardness result for order finding is that the corresponding learning problem and the algebraic circuit minimization problem for ROABPs also become NP-hard. Recall that in circuit minimization for any circuit class $\mathcal{C}$, we are given a polynomial (as an algebraic circuit) and a parameter s, and asked if there exists a circuit $C \in \mathcal{C}$ of size at most s.

Corollary 1.4 (Refer Theorem 4.5). *The algebraic circuit minimization problem for ROABPs is NP-hard.*

Given that the hardness of ROABP order finding is already encapsulated by instances of constant-degree polynomials, this directly implies the hardness of the algebraic minimum circuit size problem (MCSP) for ROABPs with constant-degree polynomials. Specifically, given a polynomial f of degree $d = O(1)$ in the dense representation (analogous to the truth table representation in the Boolean setting), and a parameter w, deciding whether there exists an ROABP computing f of width w, is NP-complete.

Corollary 1.5 (Refer Theorem 4.2). *For a fixed d, the algebraic MCSP for ROABPs is NP-hard.*

For precise definitions of these algebraic meta-complexity problems and corresponding results, see subsection 4.1.

Due to the existence of hard instances in constant-degree polynomials, the above hardness result is similar to the hardness of set-multilinear depth-3 MCSP and depth-3 powering MCSP for degree-3 polynomials, which arises from the hardness of tensor decomposition (respectively) [Hås90, MR13, Shi16]. This contrasts with the recent hardness result of [BDSS24] which crucially requires high-degree polynomials for their hardness.

Another outcome of the *exact* relationship between cut-width and ROABP-width is that known hardness of approximation results for cut-width directly imply the hardness of α-approximation for order finding in ROABPs. In this context, the objective is to test if there exists an ordering of variables such that the ROABP-width in that order is less than αw.

Assuming the Small Set Expansion (SSE) conjecture, it is known that cutwidth is hard to approximate within any constant factor greater than 1 [APW12]. This directly implies that, under SSE, it is hard to approximate the order-finding problem.

We also provide evidence of the hardness of approximation for RO-order finding without relying on the strong assumption of SSE. Specifically, we show how to boost the hardness of approximation for RO-order from a small fixed constant to an arbitrary constant. Formally, we demonstrate that an algorithm capable of approximating order finding within any (fixed) constant can be converted into a PTAS for the order-finding problem.

Theorem 1.6 (Refer Theorem 7.1). *For any constant α, an α-approximation algorithm for finding an order that minimizes ROABP-width, implies a PTAS for the same problem.*

Note that the above result, along with the non-existence of a PTAS for cut-width, would imply hardness of approximation (HOA) for RO-order unconditionally. Although we could not find such results for cut-width in the literature. There are known results for the closely related problem of optimal linear arrangement [AMS11], and we believe that similar results can also be adapted to cut-width.

On the algorithmic front, we design a randomized algorithm for solving the order-finding problem for ROABPs. The worst-case complexity of our algorithm is $2^n \operatorname{poly}(w, d)$, which is a significant improvement over the trivial $n! \operatorname{poly}(d, w)$ time required to check all possible orders. We

also show that for generic (random) ROABPs, our algorithm runs in polynomial time and outputs a correct order with high probability. This means our algorithm is efficient for most ROABPs, requiring additional time only on a lower-dimensional subspace (sub-variety) of ROABPs. This stands in strong contrast to the case of Boolean ROBPs, where only heuristic order-finding algorithms are known [Weg00, MT98].

Theorem 1.7 (Informal version of Theorem 6.6 and Theorem 6.12). *Over any large enough field* $\mathbb{F}$, *there is a randomized algorithm R that when given a random (or generic) polynomial* $f(\mathbf{x})$ *with an ROABP of width w (in some unknown order) as a black-box, outputs an order which achieves that ROABP-width for* f, *with high probability.*

Further, the running time of this algorithm is $\mathrm{poly}(n, d, w)$ *when* $w = d^{O(1)}$, *and quasi-polynomial* $(n^{O(\log w)})$ *when d is constant.*

1.2 Ideas behind our proofs

Owing to the simplicity and the linear algebraic nature of Nisan's characterization, the proofs of our results are fairly clean. We have therefore tried to describe almost all the key ideas behind them here.

NP-hardness in the worst case

As alluded to earlier, the motivation for looking at `CutWidth` as the candidate problem to show the NP-hardness for order finding (rather, ROABP-width) comes from the hardness result for OBDD-size [BW96], which provides a reduction from the optimal linear arrangement problem.

The particulars of our reduction are fairly natural, in that given a graph G we wish to design a polynomial f_G with the following property. An optimal linear arrangement of G should correspond to an optimal order for an ROABP of f_G, and a natural way to achieve that is to ensure that the size of each cut in the linear arrangement corresponds to the width in that layer for an optimal ROABP for f_G in that order.

These criteria mean that the variables should correspond to the vertices of the graphs, and for each edge (u, v) in the graph G, f_G should include a "gadget" with the variables x_u and x_v. Whenever a set $T \subseteq [n]$ includes exactly one of u and v, each such edge-gadget should ideally increase the rank of the Nisan matrix $M^T(F)$ by a definite amount, so that the rank $(M^T(f))$ then corresponds to the size of the cut induced by the vertices in T.

If we just use $x_u x_v$ as the gadget, it already gives us a unique monomial corresponding to that edge. But, in order to ensure that all edges incident on x_u (that cross the cut) add to the rank by the same amount, we need to ensure that even the exponent of x_u (and x_v) in the gadget retains some information about the edge it came from. This can be achieved by the gadget being $x_u^v x_v^u$. Finally, observing that x_u only needs to have degree-of-u many distinct exponents in the final polynomial

f_G, takes us to the gadget $x_u^{n_u(v)} x_v^{n_v(u)}$, where $n_u(v)$ is a unique index for v in the set of neighbours of u. This lets us achieve an individual degree that is essentially the maximum degree of G, and a total degree that is just twice that. This improvement is crucial in proving NP-hardness of MCSP for ROABPs, in Corollary 4.8.

Beyond this point, the final reduction (Lemma 4.3) follows after some minor tweaks that are necessary to handle some corner cases for the choice of the set T.

Order-finding in the average (or generic) case

When designing an algorithm for order-finding, we assume that the width parameter w is given as an input, and that the input polynomials are promised to have an ROABP in some order of width w. This is without loss of generality, since we can just search for the correct w, using the proper-learning (or reconstruction) algorithms in the known-order setting (see Section 6 for details).

On a high level, our algorithm (Algorithm 2) tries to perform a greedy exploration of the space of all the possible sets T that satisfy rank $(M^T(f)) \leq w$. Since for a "correct" order σ, any set P that is prefix of σ does satisfy rank $(M^T(f)) \leq w$, the goal of this exploration is to find a sequence of 'good' sets of sizes $1, 2, \ldots, n-1$, which looks like a sequence of prefixes of some permutation σ. Such a σ should then be a valid order for a width-w ROABP computing f.

Consider the space of all possible inputs to the order-finding algorithm as above. Every input polynomial has an ROABP of width w in some order, say that order is σ. What would be bad for the greedy exploration above, is that a set T that is not a prefix of a "correct" order for f (which is possibly different from σ), still has rank $(M^T(f)) \leq w$. Worse even, if several such sets exist, then the algorithm would have to explore the paths suggested by these sets, only to then abandon them when they fail.

Now the intuition behind why such an algorithm should run efficiently on a generic input is as follows. Having small Nisan rank for a set T that is not a prefix of σ should be a "special" property, and so a *generic* polynomial with a width-w ROABP in the order σ, should be unlikely to have any such "special" properties. Phrased in algebraic-geometric terms, this means that for any 'bad' set T and the *variety* V of polynomials that have a width-w ROABP in the order σ, the polynomials that additionally have rank $(M^T(f)) \leq w$ form a sub-variety of V of a strictly smaller dimension; this is precisely Lemma 6.4. This then means that such polynomials form a tiny fraction of V, and thus, for a large enough (and especially an infinite) field, even the union of all the "special" sub-varieties corresponding to all the bad sets should still leave out most of the original variety; this is shown in Lemma 6.5.

All these arguments translate to analogous results over large finite fields, with careful applications of the Schwartz-Zippel lemma, to give us the running time guarantees. A minor issue is that when the individual degree d is too small compared to w, any set T that is small enough trivially achieves a Nisan rank less than w. This means that our exploration has to inevitably explore some

'bad sets', which give us a super-polynomial running time when d is constant.

PTAS from a constant approximation

Our idea is to simply boost the approximation ratio. This is achieved using a "black-box" transformation on polynomials (Lemma 7.2), that maps any n-variate polynomial f that has a width-w ROABP in some order, to an n-variate polynomial g_k that has a width-w^k ROABP in the same order. In fact, this transformation works for any k, and further, it raises the width in each layer to the kth exponent. Here, by a "black-box" transformation, we mean that the transformation is oblivious to what f is, what the order σ is, or what the width w is.

With this transformation, and a constant-approximation algorithm at hand, we obtain a PTAS as follows. Say the approximation ratio of the algorithm is α. If we want an order that achieves a width that is at most $(1 + \varepsilon)$ times the optimal, we apply the transformation for $k \sim \log_{(1+\varepsilon)} \alpha$, and run the α-approximation algorithm on g_k. We then output the exact same order given by the approximation algorithm, as the $(1 + \varepsilon)$-approximate order for f. It turns out that this transformation is simple enough, so that the entire procedure runs in time $m^{O(1/\varepsilon)}$ on all inputs of length m, giving us a polynomial time approximation scheme.

2   Related works

Algorithms for ROABPs   Due to the strong structure of ROABPs, algorithms for various problems involving ROABPs have been extensively studied. For instance, we have a polynomial time PIT algorithm for ROABPs in the white-box setting [RS05] and a $(ndw)^{O(\log n)}$ (quasi-polynomial) time PIT algorithm in the black-box model [FSS14, AGKS15]. Here, n is the number of variables, d is the individual degree, and w is the width. Furthermore, there are faster $(ndw)^{O(\log \log w)}$ black-box PIT algorithms for subclasses of ROABPs that are order-oblivious (a.k.a. commutative) ROABPs [GKS17].

Additionally, when provided with black-box access to a polynomial and a variable ordering, learning ROABPs becomes a tractable problem. Formally, given a variable ordering, an ROABP can be learned in randomized polynomial time and deterministic quasi-polynomial time, specifically $(ndw)^{O(\log n)}$ [KS06, FS13].

Ordered ROBPs   The ROABP class is inspired by the Boolean circuit class of ordered read-once branching programs (ROBPs), also referred to as Ordered Binary Decision Diagrams (OBDDs). These classes have been extensively studied due to their connection to the RL vs L problem [Nis90] which is the central question in space-bounded derandomization.

An ordered ROBP, or OBDD, is a branching program in which the variables are read exactly once in a fixed, consistent order across all paths from the source to the sink. The program rep-

resents Boolean functions as directed acyclic graphs, where nodes correspond to variables, and edges represent the two possible outcomes for each variable.

Even for ordered ROBPs, the problem of *order finding*—that is, given an ordered ROBP, determining whether there exists an ordering with ROBP width at most w—is a natural computational problem. In fact, this problem has been extensively studied in the Boolean setting.

Numerous works address both the hardness and algorithms for exact as well as approximate *order finding* for ROBPs. For instance, it is known that the order finding for OBDDs is NP-hard [BW96] and even NP-hard to approximate for any constant factor [Sie02]. Despite these hardness results, and due to the importance of OBDDs in both theoretical and practical fields—such as VLSI design, formal verification, machine learning, and combinatorial problems—numerous *heuristic* algorithms have been developed to solve the order-finding problem for OBDDs, see [Weg00, MT98]. For a comprehensive survey about the study of ordering in OBDDs, we refer the reader to the beautiful and well detailed book by Wegner [Weg00].

Although ROABPs share an analogous criterion for width as ROBPs, none of these Boolean results directly imply hardness or algorithmic insights for order-finding in the ROABP setting. This is because the standard algebraization of a Boolean function can yield significantly different complexities in ROABPs compared to ROBPs. The key reason for this difference is that the result analogous to Nisan's characterization (Theorem 3.6) for OBDDs relies on the *number* of distinct sub-functions [Bry86], whereas Nisan's result is based on their rank (or the dimension of the space spanned by them). We illustrate this difference with a small example below.

Example 2.1. *Consider the function $f(a, b, c, d)$ that evaluates to 1 exactly on the following inputs.*

$$I = \{(0,0,0,0),(0,1,0,1),(1,0,1,0),(1,1,0,0),(1,1,1,0)\}$$

Any OBDD for f in the order (a, b, c, d) requires width 4 in the second layer, whereas (when viewed as a multilinear polynomial) f has an ROABP in the order (a, b, c, d) that has width 3 in the second layer. ◇

Nonetheless, these results serve as valuable guides for what we might expect in the algebraic setting as well. In fact, our use of graph linear arrangement and cutwidth was inspired by reductions for the hardness of order-finding in ROBPs from [BW96]. The idea of boosting a small, constant hardness to an arbitrary constant hardness also originates in work by [Sie02]. However, in both of these reductions, we can structure the algebraic instances such that the reductions are more efficient and conceptually simpler.

It is also worth noting that no rigorous algorithm is known for finding the order for generic or random OBDDs, indicating some key differences here.

Other NP-Hardness Results in Algebraic Complexity   Unlike the Boolean world, only limited NP-hardness results are known for problems involving algebraic circuits and algebraic computations. Firstly, there are hardness results regarding (proper) learning of set-multilinear depth-3

circuits and depth-3 powering circuits. These results follow directly from the hardness of computing tensor rank (and Waring rank) [Hås90, Shi16], along with standard connections between tensor rank and set-multilinear depth-3 circuits, and Waring rank with depth-3 powering circuits [BSV21].

Additionally, there are known hardness results for equivalence testing, which essentially asks if two polynomials are equivalent up to a change of basis. Kayal [Kay11], and Agrawal-Saxena [AS06] showed that this problem is as hard as the graph isomorphism problem even for cubic polynomials!

Recently, [CGS23] demonstrated the NP-hardness of testing whether there exists a shift of a polynomial that is sparser than the polynomial itself. They showed that this problem is NP-hard over finite fields and undecidable over the integers.

Even more recently, [BDSS24] showed that, given a sparse polynomial, testing whether it is equivalent to an s-sparse polynomial under an invertible change of basis is NP-hard. They also showed that this problem is NP-hard to approximate up to $s^{\frac{1}{3}}$-approximation factor.

Generic Learning Algorithms in Algebraic Complexity   It is widely believed that learning (or reconstruction) problems for most reasonably strong classes of algebraic circuits are hard. Consequently, there has been considerable effort in designing learning algorithms for the generic case. This focus is partly due to connections with important problems, such as tensor decomposition, subspace clustering, and learning mixtures of Gaussian [GKS20, CGK+24]. In the generic case, we assume that the input comes from the space excluding some strictly smaller-dimensional variety. Naturally, generic algorithms also give average-case algorithms, as a randomly sampled point will avoid the subvariety with high probability.

We know of generic learning algorithms for several constant depth circuit models such as homogeneous depth-3, set-multilinear depth-3, generalized depth-3 circuits [KS19, BGKS22] and also 'sum of power of low-degree polynomials' [GKS20]. The main idea, as established by the work of Kayal and Saha [KS19], is to utilize the space of partial derivatives (or shifted partial derivatives), which has been used to prove circuit lower bounds, and to apply vector decomposition algorithms on these spaces.

Notably, our learning algorithm for generic ROABPs diverges from this framework, since any generic algorithm for learning ROABPs would primarily require a general approach for order finding. Our work examines the relationship between various permissible orderings in generic ROABPs.

3 Preliminaries

Notation

Apart from the conventions of notation that are commonly used, we follow these additional conventions in our paper.

- We use lowercase letters to denote indeterminates, polynomials or scalars from the field, and use the boldfaced, lowercase versions of the same letters to denote sets or vectors of the corresponding things. E.g. $\mathbf{x} = \{x_1, \ldots, x_n\}$, $\mathbf{a} = \{a_1, \ldots, a_n\}$.

  We use subscripts on these symbols for variable or polynomial sets to denote a subset. E.g. $\mathbf{x}_T = \{x_i : i \in T\}$.

- We use uppercase letters to denote matrices (and numbers that are very large), and use $A[i, j]$ to denote the (i, j)th entry of the matrix A.

- We use calligraphic uppercase letters for random variables, and use the regular version of the same letters, lower or uppercase (but never both), to denote their values or realizations. E.g. the random variables $\mathcal{A}$ and $\mathcal{P}$ take values A and p.

- For a monomial m, the *support of m* refers to the number of distinct variables that appear in the monomial: e.g. xy has support 2, x^{100} has support 1.

- For a polynomial, its total degree is the maximum degree of any monomial appearing in it, and its individual degree is the highest power that any individual variable is raised to in any monomial appearing in it. For instance, an n-variate multilinear polynomial has individual degree 1, and total degree (possibly) n.

- For a polynomial f, the *coefficient vector of f* is a list of the coefficients of all its monomials (even those that are zero) in some predefined order. For instance, for an n-variate, total-degree d polynomial, the coefficient vector has length $\binom{n+d}{d}$, and an n-variate, individual-degree d polynomial has a coefficient vector of length $(d+1)^n$.

Linear arrangement problems on graphs. Given a graph G on n vertices, a linear arrangement of G is a mapping from the vertex set to the points $\{1, \ldots, n\}$, or, just a permutation of the vertices. For a linear arrangement σ of a graph G, one can then look at various graph parameters. One such parameter is the number of edges crossing each of the "induced cuts": $\{\sigma^{-1}(1), \ldots, \sigma^{-1}(i)\}$ and $\{\sigma^{-1}(i+1), \ldots, \sigma^{-1}(n)\}$ for each i. The *optimal* or *minimum linear arrangement* (OLA or MLA) asks for an arrangement that minimizes the sum of the sizes of all these cuts, and the `CutWidth` problem asks for an arrangement that minimizes the sizes of the largest such cut. We refer the reader to the survey by Díaz, Petit and Serna [DPS02] for an overview of some well-studied linear arrangement problems on graphs.

3.1 Basics about polynomials and ROABPs

Our proofs repeatedly use the following famous fact about multivariate polynomials, which has been proven in various guises in several works over the years [Ore22, DL78, Sch80, Zip79].

Lemma 3.1 (Schwartz-Zippel Lemma). *Let $\mathbb{F}$ be any field, let $S \subseteq \mathbb{F}$ be arbitrary, and suppose $f(\mathbf{x})$ is an n-variate polynomial with total degree d. Then, for independent $a_1, a_2, \ldots, a_n$ sampled uniformly at random from S, $\Pr[f(a_1, \ldots, a_n) = 0] \leq d/|S|$.* ☐

Definition 3.2 (Read-once Oblivious ABP (ROABP)). *Let $\mathbb{F}$ be any field. Let $n, d \in \mathbb{N}$ be arbitrary, let $\sigma \in s_n$ be a permutation, and consider an n-variate polynomial $f(\mathbf{x})$ of individual degree d.*

An ROABP $R(\mathbf{x})$ computing $f(\mathbf{x})$ in the order σ is a layered, directed graph with $n + 1$ layers of vertices, with the leftmost (0th) and the rightmost (n^{th}) layers having single vertices, called the source s and the sink t. All edges in the graph are only between layers $i - 1$ and i for $i \in [n]$ labelled from left to right. Further, the weights on all the edges between layers $i - 1$ and i are univariate polynomials of degree at most d in the variable $x_{\sigma(i)}$.

Similar to an algebraic branching program (ABP), the polynomial computed by the ROABP is the sum of the polynomials computed along the different s to t paths, where each path computes the products of the weights of the edges contained in it.

The size *of an ROABP is said to be the total number of vertices in it, and the* width *of an ROABP is the maximum number of vertices it has in any layer.* ◇

Using the adjacency matrices of the sets of edges between all pairs of consecutive layers of an ROABP, we get an equivalent matrix view defined below.

Definition 3.3 (ROABP (matrix view)). *Let $\mathbb{F}$ be any field. For any $n, d, w \in \mathbb{N}$, a permutation $\sigma \in s_n$, and an n-variate polynomial $f(\mathbf{x})$ of individual degree d, we say that it has a width w ROABP in the order σ, if there exist matrices $\{A_{i,j}\}$ with entries in $\mathbb{F}$ for all $i \in [n]$ and $0 \leq j \leq d$, such that the following holds.*

For $w_0 = w_n = 1$, there are numbers $w_1, w_2, \ldots, w_{n-1} \leq w$, so that for each $i \in [j]$, $A_{i,j} \in \mathbb{F}^{w_{i-1} \times w_i}$ for all $j \in \{0, 1, \ldots, d\}$, and

$$f(\mathbf{x}) = \left(L_{\sigma(1)}(x_{\sigma(1)}) \cdot L_{\sigma(2)}(x_{\sigma(2)}) \cdots L_{\sigma(n)}(x_{\sigma(n)}) \right) [1, 1],$$

where for all $i \in [n]$,

$$L_i(x_i) = A_{i,0} + A_{i,1}x_i + A_{i,2}x_i^2 + \cdots + A_{i,d}x_i^d.$$

We call the matrices $\{A_{i,j}\}$ the coefficient matrices *of the ROABP.* ◇

Definition 3.4 (Prefixes of a permutation). *For a permutation $\sigma \in s_n$ on n letters, we say that a set $T \subseteq [n]$ is a prefix of (or with respect to) σ if $T = \{\sigma(1), \sigma(2), \ldots, \sigma(i)\}$ for some $i \in [n]$.* ◇

Definition 3.5 (Nisan matrix). *For an n-variate polynomial $f(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ of individual degree d, and a subset $T \subseteq [n]$, the* Nisan matrix *of f with respect to T is a $(d+1)^{|T|} \times (d+1)^{n-|T|}$ matrix denoted*

by $M_T(f)$, with rows labelled by monomials in the variables $\mathbf{x}_T$, and columns labelled by the monomials in $\mathbf{x}_{[n]\setminus T}$, and the (m, m')th entry of this matrix is $M_T(f)[m, m'] = \operatorname{coeff}_f(m \cdot m')$. ◇

Theorem 3.6 (Nisan's characterization [Nis91])**.** *Let $\mathbb{F}$ be any field. For any $n \in \mathbb{N}$ and any permutation $\sigma \in s_n$, let $\varnothing, T_1, T_2, \ldots, T_{n-1}, [n]$ be the prefixes of σ of lengths $0, 1, 2, \ldots, n-1, n$, respectively.*

Then for any n-variate polynomial $f(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$, the optimal ROABP for f in the order σ has exactly rank $\left(M^{T_i}(f)\right)$ *vertices in the ith layer (after "reading" first i variables). Thus, the optimal ROABP has width exactly* $\max_{i \in [n]}$ rank $\left(M^{T_i}(f)\right)$*, and size exactly* $\sum_{i \in [n]}$ rank $\left(M^{T_i}(f)\right)$*.* □

3.2 Generic and random ROABPs

Generic Conditions: A generic point of an *algebraic variety* $V \subseteq \mathbb{F}^N$ is one that lies outside any proper closed subset of V defined by additional algebraic conditions. In other words, it avoids the "special" sub-varieties where additional polynomial relations hold, meaning it satisfies the "general" conditions that are typical of most points on V.

Thus, for $\mathbb{F}$ being either the field of rational, real or complex numbers, we consider the variety formed by the *coefficient vectors* of length $(d+1)^n$ of all n-variate, individual degree d polynomials that have an ROABP of width w in some order σ (see Section 6.1). Thus, by a generic ROABP (of a certain width in a certain order), we simply mean that the coefficient vector is generic and does not come from a strict subvariety of the above variety. It turns out that over any infinite field and for some well-behaved varieties, if we sample a random ROABP, it will be a generic ROABP with high probability. We formalize the notion of a random ROABP below (as it is more amenable for computer science applications).

Definition 3.7 (Random ROABP)**.** *Let $\mathbb{F}$ be any field, and let $n, d, w \in \mathbb{N}$, $\sigma \in s_n$, and $S \subseteq \mathbb{F}$ be arbitrary. Define random variables $\vec{\mathcal{A}} = \left\{\mathcal{A}_{i,j}\right\}_{i \in [n], j \in \{0,1,\ldots,d\}}$, where each $\mathcal{A}_{i,j} \in \mathbb{F}^{w \times w}$ each of whose entries is sampled independently and uniformly at random from S. This then gives us a random variable $\mathcal{P}_{S,n,d,w,\sigma}$, which is the polynomial computed by the ROABP given by*

$$\mathcal{P}_{S,n,d,w,\sigma}(x_1, \ldots, x_n) := \left(\prod_{i \in [n]} \left(\sum_{j=0}^{d} \mathcal{A}_{i,j} \cdot x_{\sigma(i)}^{j} \right) \right) [1, 1].$$

We call this random variable $\mathcal{P}_{S,n,d,w,\sigma}$ a random n-variate ROABP of individual degree d of width w in the order σ over the set S. That is, the distribution on the realizations of this random variable is induced by the uniform distribution over S in the entries of all the $\mathcal{A}_{i,j}$s.

We drop the subscripts S, n, d, w, σ and just write $\mathcal{P}$ when those details are clear from the context. ◇

Remark 3.8. *Note that Definition 3.7 can be defined analogously for other "standard" distributions the entries of the coefficient matrices. The only criterion is that these distributions should follow an analogue of the Schwartz-Zippel lemma Lemma 3.1.* ◇

Observation 3.9. *For any valid* $\mathbb{F}, \sigma, n, d, w$, *the coefficients of* $\mathcal{P}_{n,d,w,\sigma}$ *are homogeneous multilinear polynomials in the random variables* $\vec{A}$ *of total degree exactly* n.

Proof. The claim follows by observing that for any monomial $m = x_1^{e_1} x_2^{e_2} \ldots x_n^{e_n}$, its coefficient in $\mathcal{P}$ is exactly

$$\text{coeff}_{\mathcal{P}}(m) = \left(\prod_{i \in [n]} \mathcal{A}_{i,e_i} \right)[1,1]. \qquad \square$$

3.3 Useful facts about ROABPs

We start by stating the following facts about Nisan matrices, which are essentially facts about the ranks of Kronecker products and sums of matrices.

Observation 3.10 (Properties of Nisan matrices). *For polynomials* $f(x_1, \ldots, x_n), g(y_1, \ldots, y_n)$ *and subsets* $T, U \subseteq [n]$, *we have the following.*

1. *For* $h(z_1, \ldots, z_{2n}) := f(z_1, \ldots, z_n) \cdot g(z_{n+1}, \ldots, z_{2n})$ *and* $U' := \{n + u : u \in U\}$,
   $$\text{rank}\left(M^{T \cup U'}(h) \right) = \text{rank}\left(M^T(f) \right) \cdot \text{rank}\left(M^U(g) \right).$$

2. *For* $h(z_1, \ldots, z_n) := f(z_1, \ldots, z_n) + g(z_1, \ldots, z_n)$,
   $$\text{rank}\left(M^T(h) \right) \leq \text{rank}\left(M^T(f) \right) + \text{rank}\left(M^T(g) \right). \qquad \square$$

Lemma 3.11 (Evaluation dimension and Nisan matrix). *Suppose* $f(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ *be an n-variate polynomial with individual degree d. Let $T \subseteq [n]$ be any subset and $S \subset \mathbb{F}$ with at least $(d+1)$ elements. Define a matrix $E_T(f)$ with $(d+1)^{|T|}$ rows and $(d+1)^{(n-|T|)}$ columns, indexed by* evaluation points *from $S^{|T|}$ and $S^{(n-|T|)}$, respectively; the entries of $E_T(f)$ are: $E_T(f)[\alpha, \beta] = f(\mathbf{x}_T = \alpha, \mathbf{x}_{([n] \setminus T)} = \beta)$.*
   Then, $\text{rank}(E_T(f)) = \text{rank}\left(M^T(f) \right)$.

Proof sketch. Let $S = \{s_1, \ldots, s_{(d+1)}\}$, and suppose V_S is the $(d+1) \times (d+1)$ Van der Monde matrix, with $V[i,j] = s_i^{j-1}$. Then the proof follows from the fact that $E_T = V_S^{\otimes |T|} \cdot M^T \cdot \left(V_S^\mathsf{T} \right)^{\otimes |T|}$, where $A^{\otimes k}$ is the k-wise Kronecker/tensor product of the matrix A with itself. $\qquad \square$

Lemma 3.12 (Estimating Nisan rank using PIT). *Let $n, d, w \in \mathbb{N}$ be arbitrary, and let $S \subseteq \mathbb{F}$ be any set of field constants. Suppose $f(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ is an n-variate polynomial of individual degree d, and $T \subseteq [n]$ is of size k.*
   Then for $\alpha_1, \alpha_2, \ldots, \alpha_{w+1} \in S^k$ and $\beta_1, \beta_2, \ldots, \beta_{w+1} \in S^{n-k}$ drawn uniformly at random, the $(w+1) \times (w+1)$ matrix $\hat{E}$ with $\hat{E}[i,j] = f(\mathbf{x}_T = \alpha_i, \mathbf{x}_{[n] \setminus T} = \beta_j)$ for each $i, j \in [w+1]$, has a nonzero determinant if and only $\text{rank}\left(M^T(f) \right) > w$, *with probability at least* $nd(w+1)^2/|S|$.

Proof. Let us define a $(w+1) \times (w+1)$ matrix E_0, with polynomials in the "fresh" variables $\mathbf{y}_1, \ldots, \mathbf{y}_{w+1}$ and $\mathbf{z}_1, \ldots, \mathbf{z}_{w+1}$ as entries. In particular, $E_0[i,j] = f(\mathbf{x}_T = \mathbf{y}_i, \mathbf{x}_{[n] \setminus T} = \mathbf{z}_j)$ for each $i, j \in$

$[w + 1]$. The determinant of this matrix is a nonzero polynomial, *if and only if* rank $\left(M^T(f)\right) > w$, because we can substitute the $\mathbf{y}_i$s and the $\mathbf{z}_j$s appropriately, from Lemma 3.11. Further, the total degree of this determinant is $\deg(f) \cdot (w+1)^2 \leq nd \cdot (w+1)^2$.

Therefore, checking whether the rank $\left(M^T(f)\right) > w$ reduces to the PIT of this determinant. By the Schwartz-Zippel lemma (Lemma 3.1), this is the same as testing if the determinant of the random matrix E is nonzero, as claimed. □

4 Worst case NP-hardness

This section is devoted to proving the NP-hardness of Question 1.2 in the worst case as stated below. When the total degree is constant, we obtain an NP-hardness result for testing ROABP width, even when the polynomial is provided in its *dense representation*[1]. We also get an analogous result for when the degree grows with n, the only difference being that then the dense representation has size that is $n^{\omega(1)}$, and hence we have to switch to the *sparse representation*. We now formally define the corresponding languages.

Definition 4.1 (Problems around finding ROABP-width)**.** *For every positive integer d, we define the language* DenseROwidth-*d, to contain tuples $\left(\overline{\text{coeff}}(f), w\right)$, where $\overline{\text{coeff}}(f)$ is the vector of the $\binom{n+d}{d} = n^{O(d)}$ coefficients of an n-variate, total-degree-d polynomial f, and $w \in \mathbb{N}$, such that f has an ROABP of width w in some order.*

We define the language CktROwidth *to contain 4-tuples (C, n, d, w), where $n, d \in \mathbb{N}$ are given in unary, $w \in \mathbb{N}$ is given in binary, and C is the description of an n-variate, degree-d polynomial f represented as an algebraic circuit, such that f has an ROABP of width w in some order.*

The search version of these problems: Search-DenseROwidth-*d and* Search-CktROwidth *are defined analogously, where the parameter w is not provided as an input, and an algorithm solving the problem is expected to output an order σ that achieves the optimal ROABP-width.* ◇

Theorem 4.2. *For any integer $d \geq 6$, the language* DenseROwidth-*d is NP-hard under polynomial-time Karp reductions.*

In order to prove the above theorem, we provide a polynomial-time (Karp) reduction from the CutWidth problem, defined for graphs (see e.g. [DPS02]). An important feature of this reduction is that it provides an exact relationship between the cut-width of the input graph and the ROABP-width of the output polynomial.

Lemma 4.3 (Reduction from CutWidth)**.** *Given a graph $G = (V, E)$ with n vertices, m edges, and maximum degree Δ there is a deterministic polynomial-time algorithm that outputs a polynomial $f(x_1, \ldots, x_n)$ of individual degree $d = \Delta + 1$ with $n + m$ monomials, with the following property.*

---

[1]Dense representation of f lists only the coefficients of all the monomials of f in some pre-determined order; such a representation is suited for a polynomial that is *dense*: has many monomials. On the other hand, *sparse representation* of f lists pairs of monomials and their coefficients, which is ideal for a sparse polynomial.

There is an ROABP of width $w + 2$ in an order $\sigma \in s_n$ if and only if the linear arrangement of G in the order σ has CutWidth w.

Proof. Fix $V = [n]$. For every $u \in V$, let $n_u(v) \in [\Delta]$ denote a number that uniquely identifies v within the neighborhood of u (denoted $\text{Nbr}(u)$). One canonical way of fixing $n_u(v)$ is just by fixing an order on the vertices, and then for every $u \in V$ defining $n_u(v)$ in increasing order.

For $i \in [m]$ and $e_i = \{u, v\} \in E$, define the monomial $f_i(x_u, x_v) = x_u^{n_u(v)} \cdot x_v^{n_v(u)}$, and let

$$f_G(\mathbf{x}) := \sum_{v \in [n]} x_v^{\Delta+1} + \sum_{i \in [m]} f_i = \sum_{i \in [n]} x_i^{\Delta+1} + \sum_{i \in [m]} x_u^{n_u(v)} x_v^{n_v(u)}.$$

Note that f_G has a monomial for each edge along with an $\sum_{i \in [n]} x_i^{\Delta+1}$ 'gadget'. Its support size is constant (2), individual degree is $\Delta + 1$, and total degree is 2Δ.

The proof of the lemma follows directly from the following claim using Nisan's characterization (Theorem 3.6).

Claim 4.4. *For any non-trivial partition $A \sqcup B = [n]$, we have* $\text{rank}\left(M^A(f_G)\right) = 2 + c_G(A, B)$ *where $c_G(A, B)$ is the number of edges in G that have one endpoint in A and another in B.*

Proof. $\underline{\text{rank}(M_{f_G}^{A,B}) \leq 2 + c_G(A, B)}$: The bound follows from the following description of f_G as a sum of $2 + c_G(A, B)$ variable disjoint (w.r.t A, B) products.

$$
\begin{aligned}
f_G(\mathbf{x}) = &\sum_{e_i = \{u,v\} \subset A} x_u^{n_u(v)} x_v^{n_v(u)} + \sum_{v \in A} x_v^{\Delta+1} \\
&+ \sum_{e_j = \{u,v\} \subset B} x_u^{n_u(v)} x_v^{n_v(u)} + \sum_{v \in B} x_v^{\Delta+1} \\
&+ \sum_{\substack{e_k = \{u,v\} \\ u \in A, v \in B}} x_u^{n_u(v)} x_v^{n_v(u)}.
\end{aligned}
$$

That is, the first and second terms contribute rank 1 each, and the last term contributes at most $c_G(A, B)$.

$\underline{\text{rank}(M_f^{A,B}) \geq 2 + c_G(A, B)}$: To show this, we exhibit a permutation matrix of dimension $c_G(A, B)$ that is a submatrix of $M^{A,B}(f)$, along with 2 rows that lie in a disjoint space. First, let us partition the edges into three sets; L : edges with both endpoints in A, R : edges with both endpoints in B, and C : edges that have exactly one endpoint in each A and B.

Now consider the submatrix defined by the following: for each edge $e_i = \{u, v\} \in C$, select the *row* indexed by $x_u^{n_u(v)}$ and the *column* indexed by $x_v^{n_v(u)}$. Note that, by construction, for each row in this submatrix, there is *exactly* one column with a non-zero entry. The corresponding coefficient is also 1, thus forming a permutation matrix.

14

Further, the row corresponding to any other monomials in f_G: coming from edges in L, or $x_i^{\Delta+1}$ for $i \in A$, have the entry 1 (in $M^{A,B}(f_G)$) corresponding to the constant polynomial in the column index, and zeroes in all other columns. Similarly, the columns indexed by edges in R or $x_i^{\Delta+1}$ for $i \in B$, have a 1 (in $M^{A,B}(f_G)$) in row indexed by the constant polynomial, and zeroes everywhere else. Thus, those monomials cumulatively give an additional 2-dimensional row-space disjoint from the $c_G(A,B)$-dimensional row-space that we get from the permutation submatrix discussed above. □

Clearly, f_G has total degree $2\Delta(G)$, has exactly $m + n$ monomials, and since all the nonzero coefficients are 1, its dense representation has size $n^{O(\Delta(G))}$, and its sparse representation has size at most $\text{poly}(n, m)$. Further, both these representations can easily be computed in time $\text{poly}(n, m)$ from the graph G. □

Proof of Theorem 4.2. With the reduction outlined in Lemma 4.3, the theorem follows from the fact that CutWidth is NP-hard under polynomial-time Karp reductions for planar graphs with maximum-degree 3 [MS88], as the dense representation then has size $\text{poly}(n)$. □

Owing to the exactness of our reduction from the Cut-width problem, even when the total degree of the polynomial is growing (precisely when the graph G has degree $\omega(1)$), the polynomial f_G has the appropriate ROABP-width. However, it is no longer possible to output the dense representation in time $poly(n)$ just because its length would be $n^{\omega(1)}$. Thus, we only get the NP-hardness result for the sparse representation, as stated below.

Theorem 4.5. *The language* CktROwidth *is* NP-*hard under polynomial-time Karp reductions.*

Proof. As the polynomial f_G generated by the reduction in Lemma 4.3 has at most $n + m$, support-2 monomials of individual degree at most $d = \Delta(G) + 1 \leq n + 1$ with all nonzero coefficients being just 1, it can be represented as a sum of monomials with $O((n + m) \cdot 2 \log d) = O(n^2 \log n)$ bits. The theorem is then a direct result of CutWidth being NP-hard for general undirected, unweighted graphs. □

4.1 Algebraic MCSP

The Minimum Circuit Size Problem (MCSP) is a decision problem in which the input consists of the truth table of an n-variate Boolean function and a parameter $s \in \mathbb{N}$. The goal is to decide whether f is computable by a circuit of size at most s. Whether MCSP for general Boolean circuits is NP-hard remains a long-standing open question with intriguing connections to cryptography, learning theory, average-case complexity, and proof complexity. For more details, see, for instance, [Hir22, Ila21] and the references therein.

For restricted circuit classes $\mathcal{C}$, one can define $\mathcal{C}$-MCSP, which asks whether, given a Boolean function (as a truth table) and $s \in \mathbb{N}$, there exists a circuit $C \in \mathcal{C}$ of size at most s. It is known

that MCSP is NP-hard (under deterministic polynomial-time reductions) for DNF [Mas79] and DNF ∘ XOR formulas [HOS18]. However, no NP-hardness result is known (under deterministic polynomial-time reductions) for more general circuit models such as AC^0 circuits.

We now describe the minimum circuit size problem (MCSP) in the algebraic setting. Instead of a truth table for a Boolean function, you are given the complete coefficient vector of a polynomial. Specifically, for an n-variate polynomial of total degree d, you are provided with all $\binom{n+d}{d}$ coefficients. Alternatively, one could define the input as the evaluations of the polynomial at an interpolating set; note that it is possible to convert between these representations in polynomial time.

Furthermore, as in the Boolean setting, we can specialize this question to specific circuit classes $\mathcal{C}$. Here, the problem is to determine if there exists a circuit of size s in $\mathcal{C}$.

Below, we provide a formal definition for any algebraic circuit class $\mathcal{C}$.

Definition 4.6 (Algebraic MCSP for $\mathcal{C}$). *Given an n-variate polynomial f of degree d along with its coefficient vector of $\binom{n+d}{d}$ field coefficients and a parameter $s \in \mathbb{N}$, decide if there exists an algebraic circuit C in $\mathcal{C}$ of size s that computes f.* ◇

Another related problem in meta-complexity is the Circuit Minimization Problem. The key difference between circuit minimization and MCSP is that in circuit minimization, you are provided with an explicit representation of the function (a circuit computing f) that is usually polynomial in n size, and you are asked if there exists a circuit computing f with size less than s. See, for instance, [BU11] and the references therein.

We can similarly define the algebraic version of the circuit minimization problem.

Definition 4.7 (Algebraic Circuit Minimization for $\mathcal{C}$). *Given an n-variate polynomial f of degree d, a circuit of size s' computing f, and a parameter $s \in \mathbb{N}$, decide if there exists an algebraic circuit C in $\mathcal{C}$ of size s that computes f.* ◇

Note that the input size here is $\mathrm{poly}(n, d, s')$. It is intuitive to think that the circuit minimization problem may be harder than MCSP, indeed any algorithm for circuit minimization already implies a fast algorithm for MCSP. Thus making it easier to show NP-hardness results for circuit minimization.

An interesting distinction to note is that circuit minimization in the Boolean world is a Σ_2-complete problem. However, in the algebraic world, it lies within the class MA, due to the application of Polynomial Identity Testing (PIT). Specifically, given a candidate algebraic circuit, we can efficiently verify its correctness in randomized polynomial time. See [BDSS24] for a more elaborate discussion on this.

Researchers in meta-complexity and algebraic complexity have been exploring the algebraic MCSP for some time now. Despite these efforts, not much is known, although some preliminary observations have been made in recent works by Baraskar et al. [BDSS24] and Belova et al. [BGK$^+$23].

On the hardness of algebraic MCSP for special subclasses, one can obtain NP-hardness results for set-multilinear depth-3 circuits and depth-3 powering circuits essentially for free. This is a direct consequence of the hardness of computing tensor rank (and Waring rank) [Hås90, Shi16], along with standard connections between tensor and set-multilinear depth-3 circuits and Waring rank with depth-3 powering circuits [BSV21].

Recently, a beautiful work by Baraskar et al. [BDSS24] showed circuit minimization for the orbit of sparse polynomials, which is a special subclass of depth-3 circuits. They established the hardness of this problem via a natural reduction from SAT. However, their work only provides hardness results for circuit minimization within this class and fails to imply the hardness of Algebraic MCSP. A reason for this is that the reduction in the work by [BDSS24] also genuinely requires high-degree polynomials, making all dense representations very verbose and insufficient to imply the hardness of MCSP.

As a direct consequence of the NP-hardness of order finding for ROABPs, we can deduce hardness for circuit minimization and Algebraic MCSP for ROABPs. This is because, contrary to the hard instance of [BDSS24], our hardness can arise even from constant-degree polynomials. We state these observations formally below.

Corollary 4.8 (Circuit Minimization / Algebraic MCSP for ROABPs is NP-hard). *The following are true over any field* $\mathbb{F}$.

- *The algebraic circuit minimization problem for ROABPs is* NP-*hard. That is, given an n-variate polynomial f of degree d, a circuit of size s' computing f, and a parameter $w \in \mathbb{N}$, deciding whether there exists an ROABP C of width at most w that computes f is* NP-*hard.*

- *For a fixed $d \in \mathbb{N}$, algebraic MCSP for ROABPs is* NP-*hard. That is, given an n-variate polynomial f of degree d along with its coefficient vector of $\binom{n+d}{d}$ field coefficients and a parameter $w \in \mathbb{N}$, deciding whether there exists an ROABP C in $\mathcal{C}$ of width at most w that computes f is* NP-*hard.*

Proof. Note that the existence of an ROABP of width less than w directly implies that there exists an order under which the ROABP width is less than w. Thus, by Theorem 4.5 and Theorem 4.2, we obtain NP-hardness for ROABP circuit minimization and ROABP MCSP, respectively. □

5 Algorithm for order finding

We now describe a procedure (Algorithm 2) that solves the order finding problem when given an n-variate, individual degree d polynomial $f(\mathbf{x})$ as a "black-box" (algorithm can query f at any set of points). The algorithm solves a slightly different problem, in that it is given the width w as a parameter. It should be noted that this is essentially the same problem for the case of ROABPs, since given a purported order σ and a black-box for f, there is a randomized polynomial time algorithm for reconstructing an ROABP for f in the order σ (see Section 2). Therefore, if w is

not provided as an input, we could just guess a w and run our algorithm. If it fails or output an incorrect order, we infer that our guess for w was wrong. This gives a straight-forward way to perform a *binary search* for the correct w.

Our algorithm runs in two stages. In the first stage, it populates the list of all sets T such that rank $\left(M^T(f)\right) \leq w$, and in the second stage, it views this list as a graph (i.e. a subgraph of the *Hasse diagram* of the boolean hypercube) to then find a path in it from $\varnothing$ to $[n]$ which reads out an order τ in which the input polynomial has a width-w ROABP. As the second stage is just a standard DFS, we provide an explicit algorithm for the first stage in Algorithm 1, and only outline the other parts.

The algorithm runs in deterministic time $2^{O(n)} \cdot \text{poly}(d, w)$ in the worst case, which is consistent with the order finding problem being NP-hard for some settings of the parameters, as shown in Section 4. In Section 6, we show that Algorithm 2 runs in randomized time $n^{O(\log_{(d+1)} w)} \cdot \text{poly}(d, w)$ with high probability when the input is a random polynomial with an ROABP of width w in some order, as in Definition 3.7.

Algorithm 1: POPULATEGRAPH

Input : Parameters n, d, w and the polynomial $f(\mathbf{x}) \sim \mathcal{P}_{S,n,d,w,\sigma}$ given as a blackbox.
Output: Families $L_1, \ldots, L_{n-1}$ of subsets T with rank $\left(M^T(f)\right) \leq w$, categorized by size.

1 Set $k \leftarrow 1$ // Size of the subsets to be tested
2 Set $L_0 \leftarrow \{\varnothing\}$ // List of ''good'' subsets of size 0
3 Set $\mathcal{C} \leftarrow \varnothing$ // Initialize the collection to be finally output

4 **while** $k < n$ **do**

5     $L_k \leftarrow \varnothing$ // Initialize the list of subsets of size k

      /* Try and extend every subset from L_{k-1} */
6   **for** $T_0 \in L_{k-1}$ **do**
7   **for** $i \in [n]$ **do**
8   Set $T \leftarrow T_0$
9   **if** $i \notin T$ **then**
10   Set $T \leftarrow T \cup \{i\}$
              /* We check the Nisan rank using Lemma 3.12   */
              /* The PIT uses the same set S as in $\mathcal{P}_{S,n,d,w,\sigma}$ */
11   **if** $T \notin L_k$ *and* rank $\left(M^T(f)\right) \leq w$ **then**
12   Set $L_k \leftarrow L_k \cup \{T\}$

13   Set $\mathcal{C} \leftarrow \mathcal{C} \cup \{L_k\}$
14   Set $k \leftarrow k + 1$

15 **return** $\mathcal{C}$

Algorithm 2: FINDORDER

    Input : Parameters n, d, w and the polynomial $f(\mathbf{x}) \sim \mathcal{P}_{S,n,d,w,\sigma}$ given as a blackbox.

    Output: Permutation τ such that f has an ROABP of width w in the order τ (w.h.p.).

1 Let $\mathcal{C} = \texttt{PopulateGraph}(n, d, w, f)$

2 Generate a graph G using $\mathcal{C}$

    `/* Vertex set of G is the union of L₀, L₁,..., Lₙ.                          */`

    `/* There is an edge from T to T', if T' = T ∪ {t} for some t ∈ [n] \ T.      */`

3 Let $(\varnothing, T_1, T_2, \ldots, T_{n-1}, [n])$ be a path in G `// Found using, say a DFS of G`

4 Set τ: $\tau(i) = t$ if and only if $T_i = T_{i-1} \cup t$, for each $i \in [n]$

5 return τ

---

6   Complexity in the generic and average case

We now turn to showing that "on average", Algorithm 2 runs in time that is much better than 2^n with high probability. In particular, it runs in randomized time $n^{O(\log_{d+1} w)} \operatorname{poly}(d, w)$ for n-variate, individual degree d polynomials that have ROABPs of width w; this running time is polynomial when $w = \operatorname{poly}(d)$, and quasi-polynomial even when d is a constant and $w = \operatorname{poly}(n)$.

    We start by outlining the key ideas using the language of generic ROABPs, and then extend the same arguments to the random case (see Section 3.2 for the formal definitions).

6.1   The generic case

The simple intuition here is as follows. We expect the order-finding algorithm to work better in the generic case, because for any order σ, a generic polynomial that has a "small" ROABP in that order does not satisfy any additional conditions, essentially by definition. What then remains to check is that having a low-rank Nisan matrix $M^T(f)$ for a set T that is "not consistent with σ" is indeed an additional, or rather an independent, condition. We prove this in Lemma 6.4.

    We use some elementary concepts from algebraic geometry (e.g. algebraic varieties) in this subsection. While the intuitive meaning of these concepts should be clear from the discussion below, we point the reader to any text on algebraic geometry (e.g. [CLO07]) for the formal definitions.

Definition 6.1 (Varieties from Nisan matrices). *Let $\mathbb{F}$ be the field of rational, real or complex numbers. For any $n, d, w \in \mathbb{N}$ and $T \subseteq [n]$, we define for $N := (d+1)^n$, the set $V_{w,T} \subseteq \mathbb{F}^N$ to be the smallest algebraic variety that contains the coefficient vectors of all n-variate polynomials individual degree d that satisfy* $\operatorname{rank}\left(M^T(f)\right) \leq w$.

    Note that $V_{w,T}$ is generated by the $(w+1) \times (w+1)$ minors of the Nisan matrix $M^T()$. That is, every point in $f \in V_{w,T}$ is a common zero of all those minors (which are polynomials in coefficients of f). $\qquad \diamond$

    Analogously, we define the variety of polynomials that have ROABPs of some width in some

fixed order. This is precisely the variety of "generic ROABPs".

Definition 6.2 (Generic polynomials with ROABPs). *Let $\mathbb{F}$ be the field of rational, real or complex numbers. For any $n, d, w \in \mathbb{N}$ and $\sigma \in s_n$, we define for $N := (d+1)^n$, the set $V_{w,\sigma} \subseteq \mathbb{F}^N$ to be the smallest algebraic variety that contains the coefficient vectors of all n-variate polynomials individual degree d that have an ROABP of width w in the order σ.* ◊

The following relation between $V_{w,\sigma}$ and the appropriate $V_{w,T}$s is an immediate consequence of Nisan's characterization (Theorem 3.6). Note that $V_{w,\varnothing}$ and $V_{w,[n]}$ are trivial varieties.

Observation 6.3. *For any $w \in \mathbb{N}, \sigma \in s_n$, let $T_1, T_2, \ldots, T_{n-1}, T_n = [n]$ be the prefixes of σ as defined in Definition 3.4. Then, $V_{w,\sigma} = V_{w,T_1} \cap V_{w,T_2} \cap \cdots \cap V_{w,T_{n-1}}$.* □

In the language of the varieties defined above, the statement "having a small rank under a partition that is not consistent with σ is an additional condition" is formalized as follows.

Lemma 6.4. *Let $\sigma \in s_n$ be an arbitrary permutation, and let $n, d, w \in \mathbb{N}$ be any integers satisfying $n \geq 3 \cdot \log_{(d+1)} w$.*

Then for any size r such that $k \leq r \leq n-k$, where $k := \left\lfloor \log_{(d+1)} w \right\rfloor + 1$, and for any set T such that neither T nor $[n] \setminus T$ is a prefix of σ, we have that $V_{w,\sigma} \cap V_{w,T} \subsetneq V_{w,\sigma}$.

We will prove the lemma (at the end of Section 6.2) by showing the existence of a polynomial $f(\mathbf{x})$ that lies in $V_{w,\sigma}$, but not in $V_{w,T}$. In other words, we construct (for every valid choice of the parameters) a polynomial f that has a width-w ROABP in the order σ but has rank $\left(M^T(f) \right) > w$. We do this in two steps. First, we construct such a polynomial for $\sigma = \mathrm{id}$ and for some structured set T in Lemma 6.7. We then show how the same construction can be generalized to all sets of appropriate sizes, and all orders in Lemma 6.9.

Using Lemma 6.4, we can now say that the varieties corresponding to the "inconsistent partitions" do not cover the variety $V_{w,\sigma}$.

Lemma 6.5. *Let $\mathbb{F}$ be $\mathbb{Q}$, $\mathbb{R}$ or $\mathbb{C}$. Let $\sigma \in s_n$ be an arbitrary permutation, and let $n, d, w \in \mathbb{N}$ be any integers satisfying $n \geq 3 \cdot \log_{(d+1)} w$.*

Suppose $\mathcal{T}$ is the family of all sets T such that neither T nor $[n] \setminus T$ is a prefix of σ, and $k \leq |T| \leq n-k$, where $k := \left\lfloor \log_{(d+1)} w \right\rfloor + 1$. Then $V_{w,\sigma} \not\subset \bigcup_{T \in \mathcal{T}} V_{w,T}$.

Proof. Using the definition of a random ROABP (Definition 3.7), notice that the coefficient vectors in $V_{w,\sigma}$ are polynomials in the entries of the coefficient matrices. This means that there is a polynomial map $P : \mathbb{F}^M \to \mathbb{F}^N$, for $M = n(d+1)w^2$ and $N = (d+1)^n$, which maps the entries of the coefficient matrices to coefficient vectors inside $V_{w,\sigma}$. Since $\mathbb{F}^M$ is an irreducible[2] variety, so is $V_{w,\sigma}$.

Therefore, Lemma 6.4 tells us that the dimension of $V_{w,\sigma} \cap V_{w,T}$ is strictly smaller than that of $V_{w,\sigma}$. Finally, since $\mathcal{T}$ is a finite collection (and $\mathbb{F}$ is infinite), we cannot cover $V_{w,\sigma}$ using finitely many varieties of strictly smaller dimensions. □

---

[2]A variety is irreducible if it cannot be written as a union of two strictly smaller varieties.

Theorem 6.6 (Finding an optimal order in the generic case)**.** *Let $\mathbb{F}$ be the field of rational, real or complex numbers. Let $\sigma \in s_n$ be an arbitrary permutation, and let n, d and w be any positive integers satisfying $n \geq 3 \cdot \log_{(d+1)} w$.*

Then there is a randomized algorithm R that, when given a generic n-variate polynomial $f(\mathbf{x})$ of individual degree d that has an ROABP of width w in the order σ as a black-box*, outputs some $\tau \in s_n$, such that f has an ROABP of width at most w in the order τ.*

Further, with high probability, the running time of R is $n^{O(\log_{(d+1)} w)} \cdot \mathrm{poly}(d, w)$.

Proof. The algorithm R is exactly[3] the order finding algorithm outlined in Algorithm 2.

By Lemma 6.5, a generic polynomial exhibits Nisan matrices of rank strictly more than w for any T that is "not consistent" with the order σ. Thus, the algorithm PopulateGraph (Algorithm 1) generates lists $L_1, \ldots, L_{n-1}$, such that for $\log_{(d+1)} w < k < n - \log_{(d+1)} w$, $|L_k| \leq 2$.

Next, note that Algorithm 1 performs an identity test as described in Lemma 3.12 at most $n \cdot |L_k|$ times, for each $0 \leq k \leq n - 1$. This is because Algorithm 1 only attempts to extend the sets that the previous iteration has generated. Therefore, with probability at least $(1 - 2^{-n})$, the algorithm performs at most $\left[2(n - 2\log_{(d+1)} w) + 2 \cdot \log_{(d+1)} w \cdot \binom{n}{\log_{(d+1)} w} \right] < 2^n$ identity tests, each of which fails with probability at most $nd(w+1)^2 / |S| \leq 2^{-2n}$, when S is chosen to be appropriately large. Thus, the probability that all these PITs succeed, is at least $(1 - 2^n 2^{-2n})$, by the union bound. Therefore, R runs in time $n^{O(\log_{(d+1)} w)}$ and outputs a correct order with probability at least $(1 - 2^{-n})^2 \geq 1 - 2^{n-1}$. $\qquad\square$

We remark that since $M^T(f)$ has $(d+1)^{|T|}$ rows, $\mathrm{rank}\left(M^T(f) \right) \leq (d+1)^{|T|}$ for any f. Thus, any set T of size at most $k = \log_{d+1} w$ results in a Nisan matrix of rank at most w, and therefore, the factor of $n^{O(\log_{(d+1)} w)}$ is unavoidable in the running time for Algorithm 1.

6.2 Constructing the polynomials

Lemma 6.7. *Let $\mathbb{F}$ be any field, and n, d and w be any positive integers satisfying $n \geq 3 \cdot \log_{(d+1)} w$. Suppose $T \in [n]$ satisfies the following three properties.*

1. *T has a reasonable size: $\left\lfloor \log_{(d+1)} w \right\rfloor + 1 \leq |T| \leq n/2$.*

2. *T is "left-heavy": $|T \cap \{1, \ldots, \lfloor n/2 \rfloor\}| \geq |T \cap \{\lfloor n/2 \rfloor + 1, \ldots, n\}|$.*

3. *T is not a prefix with respect to id: $T \neq \{1, \ldots, |T|\}$.*

Then there exists an n-variate, individual-degree d polynomial $f(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ that has a width w ROABP in the order id, but satisfies $\mathrm{rank}\left(M^T(f) \right) > w$.

---

[3]The only tiny change is that since we do not have a predefined set $S \subseteq \mathbb{F}$, the rank-estimation has to choose such a set, which is taken to be $\{1, \ldots, 2^{2n+1} n d w^2\}$.
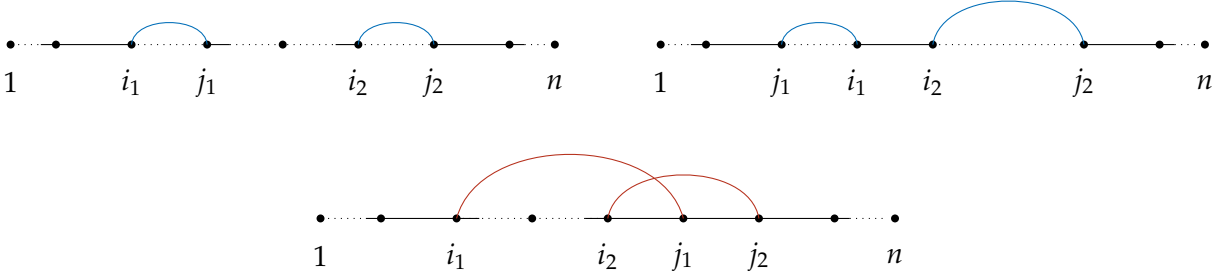
Figure 1: Examples of good (top) and bad (bottom) choices for (i_1, j_1) and (i_2, j_2).

Proof. Let $k = \left\lfloor \log_{(d+1)} w \right\rfloor + 1$, so that $k \leq |T| \leq n/2$. We will define a polynomial f that depends only on $2k$ of the n variables, split into two sets: L and R of size k each. For $L = \{i_1, i_2, \ldots, i_k\}$ and $R = \{j_1, j_2, \ldots, j_k\}$, we will then define

$$f(\mathbf{x}) = \prod_{\ell \in [k]} \left(1 + x_{i_\ell} x_{j_\ell} + x_{i_\ell}^2 x_{j_\ell}^2 + \cdots + x_{i_\ell}^d x_{j_\ell}^d \right)$$

The idea here is to choose L and R in a way that ensures that: first, $L \subseteq T$ and $R \subseteq [n] \setminus T$, and second, any prefix of id (or its complement) fully contains some pair (i_ℓ, j_ℓ). These conditions would then ensure that rank $\left(M^T(f) \right) = (d+1)^k = (d+1)^{\left\lfloor \log_{(d+1)} w \right\rfloor + 1} > w$, and that for all $i \in [n]$, rank $\left(M^{[i]}(f) \right) \leq (d+1)^{k-1} \leq w$, as required (see Observation 3.10).

We now describe how we can choose such sets L and R.

Observe that if (i_1, j_1) and (i_2, j_2) are such that the first pair is completely "on the left" of the second (that is $\max(i_1, j_1) < \min(i_2, j_2)$, see Figure 1), then we would ensure the second condition. Therefore, it is enough to show that such indices i_1, j_1 and i_2, j_2 exist with respect to any set T that satisfies the three properties in the hypothesis.

We choose i_1 to be the smallest index in T and j_1 to be the smallest index that is outside T. Similarly, we choose i_2 to be the largest index in T and j_2 to be the largest index outside T.

Since $i_1 < i_2$ and $j_1 < j_2$ by definition, we only need to show that $i_1 < j_2$ and $j_1 < i_2$. As T is left heavy, $i_1 < n/2$ and further since $|T| \leq n/2$, $j_2 > n/2$, thus $i_1 < j_2$. Also, $j_1 > i_2$ is possible only if $T = \{1, \ldots, |T|\}$, which contradicts T not being a prefix with respect to id.

Finally, we choose $i_3, \ldots, i_k$ arbitrarily from T, and $j_3, \ldots, j_k$ arbitrarily from $[n] \setminus T$. This is fine, since our choice of the first two pairs already ensures that rank $\left(M^{[i]}(f) \right) \leq w$ for any $i \in [n]$, and since T contains exactly half of all the k pairs being chosen, rank $\left(M^T(f) \right) = (d+1)^k > w$. This finishes the proof using Nisan's characterization (Theorem 3.6). $\square$

Remark 6.8. *Actually, the left-heaviness of the set T can be relaxed. That is, if T is right-heavy instead, then we would "flip" the choice of i_1, j_1, i_2 and j_2 "about the middle", by choosing i_1, j_1 to be the largest indices inside and outside T and i_2, j_2 to be the smallest indices inside and outside T. One just needs to*

"reflect the argument" about $n/2$: the first pair is to the right of the second pair and so on, for the proof to work. We skip the details to keep the proof simpler. ◇

Next, we extend the above result to all sets T that do not trivially lead to rank $\left(M^T(f)\right) \leq w$ when the order is id. This essentially boils down to working with either T or $\overline{T} = [n] \setminus T$, and with id or rev(id), to ensure that the hypothesis of Lemma 6.7. This extension then lets us bound the probability that a random polynomial with an ROABP in the order id has small Nisan-rank for an "incorrect" set T, as follows.

Lemma 6.9. *Let n, d and w be any positive integers satisfying $n \geq 3 \cdot \log_{(d+1)} w$, and let $\sigma \in s_n$ be arbitrary. Suppose $T \subseteq [n]$ is such that $k \leq |T| \leq n - k$ for $k := \left\lfloor \log_{(d+1)} w \right\rfloor + 1$, and neither T nor $([n] \setminus T)$ is a prefix with respect to σ.*

Then there exists an n-variate, individual-degree d polynomial $f(\mathbf{x})$ that has a width w ROABP in the order σ, but satisfies rank $\left(M^T(f)\right) > w$.

Proof. We can assume $\sigma = $ id by just renaming the variables in f and the indices in T to adjust for σ. That is, for $\tau = \sigma^{-1}$, if T satisfies the hypothesis of the lemma for σ, then $\tau(T) = \{\tau(t) : t \in T\}$ satisfies the hypothesis for id, and if $f(x_1, \ldots, x_n)$ satisfies the conclusion for T, then $f(x_{\tau(1)}, \ldots, x_{\tau(n)})$ satisfies the lemma for $\tau(T)$.

We now argue that we can make some simplifying assumptions on T so that we can then invoke Lemma 6.7.

- $|T| \leq n/2$: If this is not the case, then we work with the complement $\overline{T} = [n] \setminus T$, since rank $\left(M^T(f)\right) = $ rank $\left(M^{\overline{T}}(f)\right)$ for any polynomial f and any set T.

- $|T \cap \{1, \ldots, \lfloor n/2 \rfloor\}| \geq |T \cap \{\lfloor n/2 \rfloor + 1, \ldots, n\}|$: If this is not the case, then we "reverse all coordinates" by applying the permutation rev(id) and work with the reversed set rev$(T) = \{(n+1) - i : i \in T\}$. Again, this is without loss of generality, since for any permutation $\sigma \in s_n$, the set of polynomials that have an ROABP of width w in the order σ is exactly the same as the same set corresponding to rev(σ), as a direct consequence of Theorem 3.6. Refer to Remark 6.8 for further details on this assumption.

Now, since T (or $[n] \setminus T$) was not a prefix with respect to id, we get that the simplified T is not a prefix with respect to id. Thus, we can apply Lemma 6.7 to obtain an $f(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ such that rank $\left(M^T(f)\right) > w$. □

Proof of Lemma 6.4. We use Lemma 6.9 to find a polynomial $f(\mathbf{x})$ that has an ROABP of width w in the order σ, but has rank $\left(M^T(f)\right) > w$. Thus, the coefficient vector of f is a point in $V_{w,\sigma}$ that is not inside $V_{w,T}$, and therefore $V_{w,\sigma} \cap V_{w,T} \subsetneq V_{w,\sigma}$. □

6.3 Finding order of a random ROABP

We now give the "average case" analogues of our results in Section 6.1. At a high level, these results are obtained by carefully applying the Schwartz-Zippel lemma (Lemma 3.1) to bound the probabilities of a random ROABP (see Definition 3.7) satisfying any additional conditions that are not satisfied by a generic ROABP. An upshot of this exercise is that this allows us to talk about a random ROABP that is sampled to have coefficient matrices coming from large enough finite fields.

Lemma 6.10. *Let $S \subseteq \mathbb{F}$ be any set of size $2^{2n+1} \cdot n \cdot w$, let $\sigma \in s_n$ be an arbitrary permutation, and let n, d and w be any positive integers satisfying $n \geq 3 \cdot \log_{(d+1)} w$.*

Then for any size r such that $k \leq r \leq n - k$ for $k := \left\lfloor \log_{(d+1)} w \right\rfloor + 1$, and for a set T such that neither T nor $[n] \setminus T$ is a prefix of σ, the probability that the random ROABP $\mathcal{P}_{S,n,d,w,\sigma}$ has $\mathrm{rank}\left(M^T(\mathcal{P})\right) \leq w$, is at most 2^{-2n}.

Proof. For any polynomial f, we use $\overline{\mathrm{coeff}}(f)$ to denote the vector of coefficients of f.

First, we use Lemma 6.9 to find a polynomial $f(\mathbf{x})$ that has an ROABP of width w in the order σ, but has $\mathrm{rank}\left(M^T(f)\right) > w$. Now let h_0 be a $(d+1)^n$-variate polynomial that is a $(w+1) \times (w+1)$ minor of the Nisan matrix for T (viewing the coefficients of the polynomial as variables) M^T, such that $h_0(\overline{\mathrm{coeff}}(f)) \neq 0$. Such a minor, and thus the corresponding h_0, exists precisely because $\mathrm{rank}\left(M^T(f)\right) > w$.

Now consider substituting the coefficients of $\mathcal{P}_{S,n,d,w,\sigma}$: $\overline{\mathrm{coeff}}(\mathcal{P})$, in h_0. Since h_0 is a polynomial of degree $(w+1)$ in these coefficients, and the coefficients are polynomials of degree n in the entries of the random matrices $\{\mathcal{A}_{i,j}\}$ (see Observation 3.9), their composition is a polynomial of degree $n(w+1)$ in the entries of $\{\mathcal{A}_{i,j}\}$. Name this composed polynomial $g(\vec{\mathcal{A}})$.

Note that $g(\vec{\mathcal{A}})$ is a nonzero polynomial, since there is a realization of the variables that precisely yields the width-w ROABP for $f(\mathbf{x})$ in the order id, and the coefficient vector of f is not a root of h_0. This gives us the following bound on the probability of $\mathrm{rank}\left(M^T(\mathcal{P})\right) \leq w$, as claimed.

$$\Pr_{\vec{\mathcal{A}}}\left[\mathrm{rank}\left(M^T(\mathcal{P})\right) \leq w\right] = \Pr_{\vec{\mathcal{A}}}\left[\bigwedge_{h \text{ is a } (w+1) \times (w+1) \text{ minor of } M_T} h(\overline{\mathrm{coeff}}(\mathcal{P})) = 0\right]$$

$$(h_0 \text{ is one such minor}) \leq \Pr_{\vec{\mathcal{A}}}\left[h_0(\overline{\mathrm{coeff}}(\mathcal{P})) = 0\right]$$

$$(\text{Substituting for } \overline{\mathrm{coeff}}(\mathcal{P})) = \Pr_{\vec{\mathcal{A}}}\left[g(\vec{\mathcal{A}}) = 0\right]$$

$$(\text{Schwartz-Zippel lemma}) \leq \frac{\deg(g)}{|S|} \leq \frac{2nw}{|S|} \leq 2^{-2n} \qquad \square$$

Lemma 6.11. *Let $S \subseteq \mathbb{F}$ be any set of size $\geq 2^{2n+1} \cdot n \cdot w$, let $\sigma \in s_n$ be an arbitrary permutation, and let n, d and w be any positive integers satisfying $n \geq 3 \cdot \log_{(d+1)} w$.*

24

Then for $k = \left\lfloor \log_{(d+1)} w \right\rfloor + 1$, and any r such that $k \leq r \leq n - k$ the probability that the list L_r in Algorithm 1 has size greater than 2 for a random $f(\mathbf{x}) \sim \mathcal{P}_{S,n,d,w,\sigma}(\mathbf{x})$, is at most 2^{-n}.

Proof. Observe that for any fixed set T of size r that satisfies the hypothesis, we can just invoke Lemma 6.10 to derive that the probability that rank $\left(M^T(\mathcal{P}) \right) \leq w$ is at most $2nw/|\mathbb{F}| = 2^{-2n}$. Then by the union bound, the probability that any such set T exists is at most $2^n \cdot 2^{-2n} = 2^{-n}$.

As $\mathcal{P}(\mathbf{x})$ has a width-w ROABP, each list L_r in Algorithm 1 has size at least 2, corresponding to the prefixes with respect to the order σ and the complement of that prefix. If $|L_r| > 2$ for the given parameters, then there exists a subset T such that neither T nor $[n] \setminus T$ is a prefix of σ, with rank $\left(M^T(f) \right) \leq w$. The probability that such a set T is at most 2^{-n}, from the argument above. □

Theorem 6.12 (Finding an optimal order in the average case). *Over any large enough field $\mathbb{F}$, let $S \subseteq \mathbb{F}$ be of size $\geq 2^{2n+1} \cdot (ndw^2)$, let $\sigma \in s_n$ be an arbitrary permutation, and let n, d and w be any positive integers satisfying $n \geq 3 \cdot \log_{(d+1)} w$.*

Then there is a randomized algorithm R that, when given a random polynomial $f \sim \mathcal{P}_{S,n,d,w,\sigma}$ as a black-box, outputs some $\tau \in s_n$, such that f has an ROABP of width at most w in the order τ, with probability at least $(1 - 2^{n-1})$.

Further, with probability at least $(1 - 2^{n-1})$, R runs in time $n^{O(\log_{(d+1)w})} \cdot \text{poly}(d, w)$.

Proof. The algorithm R is the order-finding algorithm outlined in Algorithm 2. By Lemma 6.11, with probability at least $(1 - 2^{-n})$, the algorithm PopulateGraph (Algorithm 1) generates lists $L_1, \ldots, L_{n-1}$, such that for $\log_{(d+1)} w < k < n - \log_{(d+1)} w$, $|L_k| \leq 2$.

Next, note that Algorithm 1 performs an identity test as described in Lemma 3.12 at most $n \cdot |L_k|$ for each $0 \leq k \leq n - 1$. This is because Algorithm 1 only attempts to extend the sets that the previous iteration has generated. Therefore, with probability at least $(1 - 2^{-n})$, the algorithm performs at most $2(n - 2\log_{(d+1)} w) + 2 \cdot \log_{(d+1)} w \cdot \binom{n}{\log_{(d+1)} w}$ identity tests, each of which fails with probability at most $nd(w+1)^2/|S| \leq 2^{-2n}$. Thus, the probability that all these PITs succeed, is at least $(1 - 2^n 2^{-2n})$, by the union bound, and hence R runs in the promised time and outputs a correct order, with probability at least $(1 - 2^{-n})^2 \geq 1 - 2^{n-1}$, as required. □

7   Approximate order finding

As mentioned earlier, since CutWidth is known to be hard to approximate up to any constant factor assuming the Small Set Expansion (SSE) conjecture [APW12]. However, one can still ask if the hardness of approximation for ROABP-width follows from a more widely believed assumption.

There are two relevant results in this regard. First is a hardness of approximation (under P ≠ NP) result for approximating the OBDD-size of a boolean function, due to Sieling [Sie02]; however this result is quite technical, and it is unclear to us if it implies such a result for ROABP-width (or size). The other result is due to Ambühl, Mastrolilli and Svensson [AMS11], which

rules out a PTAS for the *minimum/optimal linear arrangement* problem, under the assumption that NP $\not\subseteq$ SUBEXP. Again, due to the exactness of our reduction from width of any cut in a linear arrangement to the width of the ROABP in the corresponding layer (see Lemma 4.3), this result directly rules out a PTAS for ROABP-size.

Turning to the problem of finding an order that approximates the optimal ROABP-width, we (the authors) do not know of a hardness result that rules out a constant approximation algorithm, or even a PTAS, for the `CutWidth` problem, under any stronger assumption that the SSE conjecture.

7.1 Constant-factor Approximation implies a PTAS

We show that for any constant α, an α-approximation algorithm for `Search-CktROwidth` implies a PTAS for it, which is formally stated below. Thus, combined with our reduction from `CutWidth` to ROABP-width, any result that rules out a PTAS for `CutWidth` under any strong assumption(s), would also rule out any constant-factor approximation algorithm for ROABP-width.

Theorem 7.1. *Suppose for a constant $\alpha > 1$ that* `ApproxWidth` *is a deterministic polynomial time algorithm, which when given the tuple (C, n, d) — where C is an algebraic circuit computing an n-variate polynomial $f(\mathbf{x})$ of individual degree d — outputs an order σ such that f has an ROABP of width w^* in the order σ, and w^* is at most α times the optimal ROABP-width of $f(\mathbf{x})$.*

Then, there is a deterministic algorithm `WidthPTAS`*, that takes as input (C, n, d, ε) (where C, n, d are as above), and outputs an order σ such that the ROABP-width of the polynomial computed by f in the order σ, is at most $(1 + \varepsilon)$ times its optimal ROABP-width. Further, this algorithm runs in time $m^{O_\alpha(1/\varepsilon)}$ on all inputs of length m.*

We prove the following lemma, which gives a way of multiplicatively increasing the optimal ROABP-width of a polynomial with an easy transformation.

Lemma 7.2 (Tensoring ROABP width). *Over any field $\mathbb{F}$, let $f(x_1, \ldots, x_n) \in \mathbb{F}[\mathbf{x}]$ be a polynomial with individual degree d, and let $\sigma \in s_n$ be arbitrary. Suppose the optimal ROABP-width for f in the order σ is w, then the following polynomial has width w^k in the order σ, for any integer $k \geq 1$.*

$$ g(\mathbf{x}) := f(x_1, \ldots, x_n) \cdot f(x_1^{d+1}, \ldots, x_n^{d+1}) \cdot \cdots \cdot f\left(x_1^{(d+1)^{k-1}}, \ldots, x_n^{(d+1)^{k-1}}\right) $$

We introduce below the concept of a lifted partition to prove the theorem.

Definition 7.3 (Lifted partitions). *Let $\mathbf{Y} = Y_1 \cup Y_2 \cup \cdots \cup Y_n$ be a set of variables containing n blocks: $Y_i = \{y_{i,1}, \ldots, y_{i,k}\}$ for each $i \in [n]$. We call a partition $\mathbf{Y} = S \sqcup T$ lifted, if for each Y_i, either $Y_i \subseteq S$ or $Y_i \subseteq T$.*

In other words, a lifted partition $S \sqcup T$ has an underlying partition $A \sqcup B = [n]$ such that S and T are exactly the unions of the blocks in A and B respectively. ◇

Along the lines on Observation 3.10, lifted partitions let us increase the optimal ROABP-width multiplicatively.

Lemma 7.4. *For a polynomial $f(x_1, \ldots, x_n)$ and any integer $k \geq 1$, let $h_k(\mathbf{Y})$ be*

$$h_k(\mathbf{Y}) := f(y_{1,1}, \ldots, y_{n,1}) \cdot f(y_{1,2}, \ldots, y_{n,2}) \cdots f(y_{1,k}, \ldots, y_{n,k}),$$

with $\mathbf{Y} = Y_1 \cup Y_2 \cup \cdots \cup Y_n$, and each $Y_i = \{y_{i,1}, \ldots, y_{i,k}\}$.

Suppose $\mathbf{X} = S \sqcup T$ is a lifted partition, where the underlying partition is $[n] = A \sqcup B$. Then, if the rank $\left(M^A(f)\right) = w$, then $\operatorname{rank}\left(M^S(h_k)\right) = w^k$.

Proof. Let $M = M^{(A,B)}(f)$, then we show that $M^{(S,T)}(h_k)$ is precisely the Kronecker (or "tensor") product $M^{\otimes k} = M \otimes M \otimes \cdots \otimes M$.

To see this, let $m = \prod_{i \in A}(\prod_{\ell \in [k]} y_{i,\ell}^{a_{i,\ell}})$ be a monomial over the variables in the set S: $\bigcup_{i \in A} Y_i$, and similarly, let $m' = \prod_{j \in B}(\prod_{\ell \in [k]} y_{j,\ell}^{b_{j,\ell}})$ be a monomial over the variables in the set T. Then, by definition of h_k, $\operatorname{coeff}_{h_k}(m \cdot m') = \prod_{\ell \in [k]} \operatorname{coeff}_f \left(\prod_{i \in A} y_{i,\ell}^{a_{i,\ell}} \cdot \prod_{j \in B} y_{j,\ell}^{b_{j,\ell}}\right)$. This essentially means that $M^{(S,T)}(h_k)[m, m'] = \prod_{\ell \in [k]} M^{(A,B)}(f)[m_\ell, m'_\ell]$, where $m_\ell = \prod_{i \in A} y_{i,\ell}^{a_{i,\ell}}$ and m'_ℓ is defined analogously. This is exactly the same as saying $M^{(S,T)}(h_k) = (M^{(A,B)}(f))^{\otimes k}$. This finishes the proof. $\qquad \square$

Proof of Lemma 7.2. First, define the polynomial h_k from f, in a similar manner as in Lemma 7.4. Therefore, for the unique $\mathbb{F}$-linear map $\mu : \mathbb{F}[\mathbf{Y}] \to \mathbb{F}[\mathbf{x}]$ that satisfies $\mu : y_{i,j} \mapsto x_i^{(d+1)^{j-1}}$ for all $y_{i,j}$, we have that $g(\mathbf{x}) = \mu(h_k)$.

Thus, since the individual degree of h_k is d, for any monomial $m_0 = \prod_{i \in [n]} \prod_{j \in [k]} y_{i,j}^{e_{i,j}}$, there is a *unique* corresponding monomial $m = \prod_{i \in [n]} \prod_{j \in [k]} x_i^{e_{i,j} \cdot (d+1)^{j-1}}$, such that $\operatorname{coeff}_g(m) = \operatorname{coeff}_{h_k}(m_0)$. Therefore, the map μ is a bijection between the monomials of h_k and g. This means that for any partition $[n] = A \sqcup B$, the matrices $M^{(A,B)}(g)$ and $M^{(S,T)}(h_k)$ are exactly the same, up to renaming their rows and columns, under the corresponding *lifted partition* (S, T). So the $\operatorname{rank}(M^{(A,B)}(g))$ is exactly the k-th power of the rank of the Nisan matrix of f under the partition (A, B).

Since the rank of every Nisan matrix has been raised to the power k in the transformation from f to g, the width of an optimal ROABP for g under any order σ is exactly the k-th power of that for f, using Nisan's characterization (Theorem 3.6). $\qquad \square$

Proof of Theorem 7.1. We first describe the algorithm `WidthPTAS` and then argue its correctness and running time. Suppose the input is (C, n, d, ε), where C computes the polynomial $f(x_1, \ldots, x_n)$.

1. Let $k = \lceil \log \alpha / \log(1 + \varepsilon) \rceil$.

2. Define $\tilde{C}(\mathbf{x}) := C(x_1, \ldots, x_n) \cdot C\left(x_1^{(d+1)}, \ldots, x_n^{(d+1)}\right) \cdots C\left(x_1^{(d+1)^{k-1}}, \ldots, x_n^{(d+1)^{k-1}}\right)$.

3. Let $w^* = \texttt{ApproxWidth}(\tilde{C}, n, d_k)$ for $d_k = (d+1)^k - 1$.

4. Compute $\hat{w} = (w^*)^{1/k}$, and return $\hat{w}$.

Correctness. Suppose the optimal ROABP width of the input $f(\mathbf{x})$ is w. From Lemma 7.2, it is clear that the polynomial computed by $\tilde{C}$ has ROABP-width exactly w^k, whenever $f(\mathbf{x})$ has width w. Since we are assured that $w^* \leq \alpha \cdot w^k$, $\hat{w} = (w^*)^{1/k} \leq \alpha^{1/k} \cdot w$. Therefore, $\hat{w} \leq \alpha^{\log(1+\varepsilon)/\log\alpha} \cdot w$, which is at most $(1+\varepsilon)w$, as required.

Running time. Suppose `ApproxWidth` runs in time at most m^a on any input of length m. The only non-trivial step of the above algorithm is that of computing the circuit $\tilde{C}$. Such an algebraic circuit has size at most $k\left(\text{size}(C) + 2k\log(d+1)\right)$, using 'repeated-squaring' gadgets for computing the appropriate powers of the x_is. So $\text{size}(\tilde{C}) \leq O_\alpha\!\left((1/\varepsilon)^2 \cdot \text{size}(C)\right)$, where the hidden constant depends on the approximation ratio α of `ApproxWidth`. Next, the length of the unary representation for $(d+1)^k - 1$ has length $d^{O(k)} = d^{O_\alpha(1/\varepsilon)}$. Thus, step 3 runs `ApproxWidth` on an input of length at most $m_0^{\beta\cdot(1/\varepsilon)}$, where m_0 is the length of the input (C, n, d) and β is a constant dependent on α. The algorithm `WidthPTAS` thus runs in time at most $m_0^{a\cdot\beta\cdot(1/\varepsilon)}$. This finishes the proof. $\qquad\square$

Open questions

We conclude with the following questions that we feel are natural to explore from here about the order-finding problem.

- Does Algorithm 1 run in randomized polynomial time in the average case for fields of size $\text{poly}(n,d,w)$? That is, is there a different argument that shows such a result?

- Are there efficient algorithms that approximate ROABP-width up to a $\text{poly}(w,n,d)$ approximation factor?

Acknowledgements

References

[AGKS15] Manindra Agrawal, Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Hitting-Sets for ROABP and Sum of Set-Multilinear Circuits. *SIAM Journal of Computing*, 44(3):669–697, 2015. Pre-print available at arXiv:1406.7535. [Cited on page 6.]

[AMS11]   Christoph Ambühl, Monaldo Mastrolilli, and Ola Svensson. Inapproximability Results for Maximum Edge Biclique, Minimum Linear Arrangement, and Sparsest Cut. *SIAM Journal on Computing*, 40(2):567–596, 2011. [Cited on pages 3 and 25.]

[APW12]   Per Austrin, Toniann Pitassi, and Yu Wu. Inapproximability of Treewidth, One-Shot Pebbling, and Related Layout Problems. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings*, volume 7408 of *Lecture Notes in Computer Science*, pages 13–24. Springer, 2012. [Cited on pages 3 and 25.]

[AS06]   Manindra Agrawal and Nitin Saxena. Equivalence of-algebras and cubic forms. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 115–126. Springer, 2006. [Cited on page 8.]

[BDSS24]   Omkar Baraskar, Agrim Dewan, Chandan Saha, and Pulkit Sinha. NP-Hardness of Testing Equivalence to Sparse Polynomials and to Constant-Support Polynomials. In *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia*, volume 297 of *LIPIcs*, pages 16:1–16:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. [Cited on pages 3, 8, 16, and 17.]

[BGK+23]   Tatiana Belova, Alexander Golovnev, Alexander S. Kulikov, Ivan Mihajlin, and Denil Sharipov. Polynomial formulations as a barrier for reduction-based hardness proofs. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 3245–3281. SIAM, 2023. [Cited on page 16.]

[BGKS22]   Vishwas Bhargava, Ankit Garg, Neeraj Kayal, and Chandan Saha. Learning Generalized Depth Three Arithmetic Circuits in the Non-Degenerate Case. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2022, September 19-21, 2022, University of Illinois, Urbana-Champaign, USA (Virtual Conference)*, volume 245 of *LIPIcs*, pages 21:1–21:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. [Cited on page 8.]

[Bry86]   Randal E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986. [Cited on page 7.]

[BSV21]   Vishwas Bhargava, Shubhangi Saraf, and Ilya Volkovich. Reconstruction algorithms for low-rank tensors and depth-3 multilinear circuits. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 809–822. ACM, 2021. [Cited on pages 8 and 17.]

[BT24]   Vishwas Bhargava and Anamay Tengse. Explicit Commutative ROABPs from Partial Derivatives. *CoRR*, abs/2407.10143, 2024. Pre-print available at `arXiv:2407.10143`. [Cited on page 1.]

[BU11]   David Buchfuhrer and Christopher Umans. The complexity of boolean formula minimization. *Journal of Computer and System Sciences*, 77(1):142–153, 2011. [Cited on page 16.]

[BW96]   Beate Bollig and Ingo Wegener. Improving the variable ordering of OBDDs is NP-complete. *IEEE Transactions on computers*, 45(9):993–1002, 1996. [Cited on pages 4 and 7.]

[CGK+24] Pritam Chandra, Ankit Garg, Neeraj Kayal, Kunal Mittal, and Tanmay Sinha. Learning Arithmetic Formulas in the Presence of Noise: A General Framework and Applications to Unsupervised Learning. In *15th Innovations in Theoretical Computer Science Conference, ITCS 2024, January 30 to February 2, 2024, Berkeley, CA, USA*, volume 287 of *LIPIcs*, pages 25:1–25:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. [Cited on page 8.]

[CGS23]   Suryajith Chillara, Coral Grichener, and Amir Shpilka. On Hardness of Testing Equivalence to Sparse Polynomials Under Shifts. In *40th International Symposium on Theoretical Aspects of Computer Science, STACS 2023, March 7-9, 2023, Hamburg, Germany*, volume 254 of *LIPIcs*, pages 22:1–22:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. [Cited on page 8.]

[CLO07]   David A. Cox, John B. Little, and Donal O'Shea. *Ideals, Varieties and Algorithms*. Undergraduate texts in mathematics. Springer, 2007. [Cited on page 19.]

[DDS21a]   Pranjal Dutta, Prateek Dwivedi, and Nitin Saxena. Demystifying the border of depth-3 algebraic circuits. In *Proceedings of the 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2021)*, 2021. [Cited on page 1.]

[DDS21b]   Pranjal Dutta, Prateek Dwivedi, and Nitin Saxena. Deterministic Identity Testing Paradigms for Bounded Top-Fanin Depth-4 Circuits. In *36th Computational Complexity Conference, CCC 2021, July 20-23, 2021, Toronto, Ontario, Canada (Virtual Conference)*, volume 200 of *LIPIcs*, pages 11:1–11:27. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. [Cited on page 1.]

[DL78]   Richard A. DeMillo and Richard J. Lipton. A Probabilistic Remark on Algebraic Program Testing. *Information Processing Letters*, 7(4):193–195, 1978. [Cited on page 10.]

[DPS02]   Josep Díaz, Jordi Petit, and Maria Serna. A survey of graph layout problems. *ACM Comput. Surv.*, 34(3):313–356, September 2002. [Cited on pages 9 and 13.]

[FS13]   Michael A. Forbes and Amir Shpilka. Quasipolynomial-Time Identity Testing of Non-commutative and Read-Once Oblivious Algebraic Branching Programs. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2013)*, pages 243–252, 2013. Full version at arXiv:1209.2408. [Cited on pages 1 and 6.]

[FSS14]   Michael A. Forbes, Ramprasad Saptharishi, and Amir Shpilka. Hitting sets for multi-linear read-once algebraic branching programs, in any order. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC 2014)*, pages 867–875, 2014. [Cited on page 6.]

[GJ79]   Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA, 1979. [Cited on page 2.]

[GKS17]   Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Identity Testing for Constant-Width, and Commutative, Read-Once Oblivious ABPs. *Theory of Computing*, 13(1):1–21, 2017. Preliminary version in the *31st Annual Computational Complexity Conference (CCC 2016)*. arXiv:1601.08031. [Cited on page 6.]

[GKS20]   Ankit Garg, Neeraj Kayal, and Chandan Saha. Learning sums of powers of low-degree polynomials in the non-degenerate case. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 889–899. IEEE, 2020. [Cited on page 8.]

[GKST17]   Rohit Gurjar, Arpita Korwar, Nitin Saxena, and Thomas Thierauf. Deterministic Identity Testing for Sum of Read-once Oblivious Arithmetic Branching Programs. *Computational Complexity*, 26(4):835–880, 2017. Preliminary version in the *30th Annual Computational Complexity Conference (CCC 2015)*. arXiv:1411.7341. [Cited on page 1.]

[Hås90]   Johan Håstad. Tensor Rank is NP-Complete. *J. Algorithms*, 11(4):644–654, 1990. [Cited on pages 3, 8, and 17.]

[Hir22]   Shuichi Hirahara. NP-Hardness of Learning Programs and Partial MCSP. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 968–979. IEEE, 2022. [Cited on page 15.]

[HOS18]   Shuichi Hirahara, Igor C. Oliveira, and Rahul Santhanam. NP-hardness of Minimum Circuit Size Problem for OR-AND-MOD Circuits. In *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, volume 102 of *LIPIcs*,

pages 5:1–5:31. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. [Cited on page 16.]

[Ila21]   Rahul Ilango. The Minimum Formula Size Problem is (ETH) Hard. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 427–432. IEEE, 2021. [Cited on page 15.]

[Kay11]   Neeraj Kayal. Efficient algorithms for some special cases of the polynomial equivalence problem. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1409–1421. SIAM, 2011. [Cited on page 8.]

[KNS15]   Neeraj Kayal, Vineet Nair, and Chandan Saha. Separation between Read-once Oblivious Algebraic Branching Programs (ROABPs) and Multilinear Depth Three Circuits. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:154, 2015. [Cited on page 1.]

[KS06]   Adam R. Klivans and Amir Shpilka. Learning Restricted Models of Arithmetic Circuits. *Theory of Computing*, 2(10):185–206, 2006. Preliminary version in the *16th Annual Conference on Computational Learning Theory (COLT 2003)*. [Cited on pages 1 and 6.]

[KS19]   Neeraj Kayal and Chandan Saha. Reconstruction of non-degenerate homogeneous depth three circuits. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 413–424. ACM, 2019. [Cited on page 8.]

[LST24]   Nutan Limaye, Srikanth Srinivasan, and Sébastien Tavenas. Superpolynomial Lower Bounds Against Low-Depth Algebraic Circuits. *Communications of the ACM*, 67(2):101–108, 2024. Preliminary version in the *62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2021)*. [Cited on page 1.]

[Mas79]   W. J. Masek. Some NP-complete set covering problems. *Unpublished Manuscript*, 1979. [Cited on page 16.]

[MR13]   Ernst W Mayr and Stephan Ritscher. Dimension-dependent bounds for Gröbner bases of polynomial ideals. *Journal of Symbolic Computation*, 49:78–94, 2013. [Cited on page 3.]

[MS88]   B. Monien and I.H. Sudborough. Min cut is NP-complete for edge weighted trees. *Theoretical Computer Science*, 58(1):209–229, 1988. [Cited on page 15.]

[MT98]   Christoph Meinel and Thorsten Theobald. *Algorithms and Data Structures in VLSI Design: OBDD-foundations and applications*. Springer Science & Business Media, 1998. [Cited on pages 4 and 7.]

[Nis90]   Noam Nisan. Psuedorandom Generators for Space-Bounded Computation. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 204–212. ACM, 1990. [Cited on page 6.]

[Nis91]   Noam Nisan. Lower bounds for non-commutative computation. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing (STOC 1991)*, pages 410–418, 1991. Available on citeseer:10.1.1.17.5067. [Cited on pages 1 and 11.]

[Ore22]   Øystein Ore. Über höhere Kongruenzen. *Norsk Mat. Forenings Skrifter*, 1(7):15, 1922. [Cited on page 10.]

[Raz06]   Ran Raz. Separation of Multilinear Circuit and Formula Size. *Theory of Computing*, 2(1):121–135, 2006. Preliminary version in the *45th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2004)*. Pre-print available at eccc:TR04-042. [Cited on page 1.]

[RS05]   Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non-commutative models. *Computational Complexity*, 14(1):1–19, 2005. Preliminary version in the *19th Annual IEEE Conference on Computational Complexity (CCC 2004)*. [Cited on page 6.]

[Sch80]   Jacob T. Schwartz. Fast Probabilistic Algorithms for Verification of Polynomial Identities. *Journal of the ACM*, 27(4):701–717, 1980. [Cited on page 10.]

[Shi16]   Yaroslav Shitov. How hard is the tensor rank?, 2016. Pre-print available at arXiv:1611.01559. [Cited on pages 3, 8, and 17.]

[Sie02]   Detlef Sieling. The nonapproximability of OBDD minimization. *Information and Computation*, 172(2):103–138, 2002. [Cited on pages 7 and 25.]

[Weg00]   Ingo Wegener. *Branching programs and binary decision diagrams: theory and applications*. SIAM, 2000. [Cited on pages 4 and 7.]

[Zip79]   Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 1979. [Cited on page 10.]

A Definitions and Technical Statements

Definition A.1 (Algebraic Circuit)**.** *An* algebraic circuit *is specified by a directed acyclic graph, with leaves (nodes with in-degree zero, called* inputs*) labelled by field constants or variables, and internal nodes labelled by $+$ or $\times$. The nodes with out-degree zero are called the* outputs *of the circuit. Computation proceeds in the natural way, where inductively each $+$ gate computes the sum of its children and each $\times$ gate computes the product of its children.*

The size *of the circuit is defined as the number of edges (or* wires*) in the underlying graph.*   ◇