HITTING-SETS FOR ROABP AND SUM OF SET-MULTILINEAR CIRCUITS

MANINDRA AGRAWAL , ROHIT GURJAR , ARPITA KORWAR , AND NITIN SAXENA

Abstract. We give a $n^{O(\log n)}$-time (n is the input size) blackbox polynomial identity testing algorithm for unknown-order read-once oblivious algebraic branching programs (ROABP). The best time-complexity known for this class was $n^{O(\log^2 n)}$ due to Forbes-Saptharishi-Shpilka (STOC 2014), and that too only for multilinear ROABP. We get rid of their exponential dependence on the individual degree. With this, we match the time-complexity for the unknown order ROABP with the known order ROABP (due to Forbes-Shpilka (FOCS 2013)) and also with the depth-3 set-multilinear circuits (due to Agrawal-Saha-Saxena (STOC 2013)). Our proof is simpler and involves a new technique called basis isolation.

The depth-3 model has recently gained much importance, as it has become a stepping-stone to understanding general arithmetic circuits. Its restriction to *multilinearity* has known exponential lower bounds but no nontrivial blackbox identity tests. In this paper, we take a step towards designing such hitting-sets. We give the first subexponential whitebox PIT for the sum of constantly many set-multilinear depth-3 circuits. To achieve this, we define notions of *distance* and *base sets*. Distance, for a multilinear depth-3 circuit (say, in n variables and k product gates), measures how far are the partitions from a mere *refinement*. The 1-distance strictly subsumes the set-multilinear model, while n-distance captures general multilinear depth-3. We design a hitting-set in time $(nk)^{O(\Delta \log n)}$ for Δ-distance. Further, we give an extension of our result to models where the distance is large (close to n) but it is small when restricted to certain base sets (of variables).

We also explore a new model of read-once algebraic branching programs (ROABP) where the factor-matrices are *invertible* (called invertible-factor ROABP). We design a hitting-set in time poly(n^{w^2}) for width-w invertible-factor ROABP. Further, we could do *without* the invertibility restriction when $w = 2$. Previously, the best result for width-2 ROABP was quasi-polynomial time (Forbes-Saptharishi-Shpilka, STOC 2014).

1. Introduction. The problem of *Polynomial Identity Testing* is that of deciding if a given polynomial is nonzero. The complexity of the question depends crucially on the way the polynomial is input to the PIT test. For example, if the polynomial is given as a set of coefficients of the monomials, then we can easily check whether the polynomial is nonzero in polynomial time. The problem has been studied for different input models. Most prominent among them is the model of arithmetic circuits. Arithmetic circuits are the arithmetic analog of boolean circuits and are defined over a field $\mathbb{F}$. They are directed acyclic graphs, where every node is a '+' or '×' gate and each input gate is a constant from the field $\mathbb{F}$ or a variable from $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$. Every edge has a weight from the underlying field $\mathbb{F}$. The computation is done in the natural way. Clearly, the output gate computes a polynomial in $\mathbb{F}[\overline{x}]$. We can restate the PIT problem as: Given an arithmetic circuit $\mathcal{C}$, decide if the polynomial computed by $\mathcal{C}$ is nonzero in time polynomial in the circuit size. Note that, given a circuit, computing the polynomial explicitly is not possible, as it can have exponentially many monomials. However, given the circuit, it is easy to compute an evaluation of the polynomial by substituting the variables with constants.

Though there is no known *deterministic* algorithm for PIT, there are easy randomized algorithms, e.g. [Sch80]. These randomized algorithms are based on the theorem: A nonzero polynomial, evaluated at a random point, gives a nonzero value with a good probability. Observe that such an algorithm does not need to access the structure of the circuit, it just uses the evaluations; it is a *blackbox* algorithm. The other kind of algorithms, where the structure of the input is used, are called *whitebox* algorithms. Whitebox algorithms for PIT have many known applications. E.g. graph matching reduces to PIT. On the other hand, blackbox algorithms (or *hitting-sets*)

have connections to circuit lower bound proofs. Arguably, this is currently the only concrete approach towards lower bounds, see [Mul12b, Mul12a]. See the surveys by Saxena [Sax09, Sax14] and Shpilka & Yehudayoff [SY10] for more applications.

An Arithmetic Branching Program (ABP) is another interesting model of computing polynomials. It consists of a directed acyclic graph with a source and a sink. The edges of the graph have polynomials as their weights. The weight of a path is the product of the weights of the edges present in the path. The polynomial computed by the ABP is the sum of the weights of all the paths from the source to the sink. It is well known that for an ABP, the underlying graph can seen as a layered graph such that all paths from the source to the sink have exactly one edge in each layer. And the polynomial computed by the ABP can be written as a *matrix product*, where each matrix corresponds to a layer. The entries in the matrices are weights of the corresponding edges. The maximum number of vertices in a layer, or equivalently, the dimension of the corresponding matrices is called the *width* of the ABP. It is known that symbolic determinant and ABP are equivalent models of computation [Tod91, MV97]. Ben-Or & Cleve [BOC92] have shown that a polynomial computed by a formula of logarithmic depth and constant fan-in, can also be computed by a width-3 ABP. Thus, ABP is a strong model for computing polynomials. The following chain of reductions shows the power of ABP and its constant-width version relative to other arithmetic computation models (see [BOC92] and [Nis91, Lemma 1]).

$$\text{Constant-depth Arithmetic Circuits} \leq_p \text{Constant-width ABP}$$
$$\leq_p \text{Formulas} \leq_p \text{ABP} \leq_p \text{Arithmetic Circuits}$$

Our first result is for a special class of ABP called *Read Once Oblivious Arithmetic Branching Programs (ROABP)*. An ABP is a read once ABP (ROABP) if the weights in its n layers are univariate polynomials in n distinct variables, i.e. the i-th layer has weights coming from $\mathbb{F}[x_{\pi(i)}]$, where π is a permutation on the set $\{1, 2, \ldots, n\}$. When we know this permutation π, we call it an ROABP with *known* variable order (it is significant only in the blackbox setting).

Raz and Shpilka [RS05] gave a $\text{poly}(n, w, \delta)$-time whitebox algorithm for n-variate polynomials computed by a width-w ROABP with individual degree bound δ. Recently, Forbes and Shpilka [FS12, FS13] gave a $\text{poly}(n, w, \delta)^{\log n}$-time blackbox algorithm for the same, when the variable order is known. Subsequently, Forbes et al. [FSS14] gave a blackbox test for the case of unknown variable order, but with time complexity being $\text{poly}(n)^{\delta \log w \log n}$. Note the exponential dependence on the degree. Their time complexity becomes quasi-polynomial in case of multilinear polynomials, i.e. $\delta = 1$.

In another work Jansen et al. [JQS10b] gave quasi-polynomial time blackbox test for a sum of constantly many multilinear "ROABP". Their definition of "ROABP" is more stringent. They assume that every variable appears in at most once in the ABP. Later, this result was generalized to "read-r OABP" [JQS10a], where a variable can occur in at most one layer, and on at most r edges. Our definition of ROABP seems much more powerful than both of these.

We improve the result of [FSS14] and match the time complexity for the unknown order case with the known order case (given by [FS12, FS13]). Unlike [FSS14], we do not have exponential dependence on the individual degree. Formally,

THEOREM 1. *Let $C(\mathbf{x})$ be an n-variate polynomial computed by a width-w ROABP (unknown order) with the degree of each variable bounded by δ. Then there is a $\text{poly}(n, w, \delta)^{\log n}$-time hitting set for C.*

REMARK. *Our algorithm also works when the layers have their weights as general sparse polynomials (still over disjoint sets of variables) instead of univariate polynomials (see the detailed version in Section 3).*

A polynomial computed by a width-w ABP can be written as $S^\top D(\mathbf{x})T$, where $S, T \in \mathbb{F}^w$ and $D(\mathbf{x}) \in \mathbb{F}^{w \times w}[\mathbf{x}]$ is a polynomial over the matrix algebra. Like [ASS13, FSS14], we try to construct a basis (or extract the rank) for the coefficient vectors in $D(\mathbf{x})$. We actually construct a weight assignment on the variables, which *isolates* a basis in the coefficients in $D(\mathbf{x})$. This idea is inspired from the rank extractor techniques in [ASS13, FSS14]. Our approach is to directly work with $D(\mathbf{x})$, while [ASS13, FSS14] have applied a rank extractor to small subcircuits of $D(\mathbf{x})$, by shifting it carefully. In fact, the idea of *basis isolating weight assignment* evolved when we tried to find a direct proof, for the rank extractor in [ASS13], which does not involve subcircuits. But, our techniques go much further than both [ASS13, FSS14], as is evident from our strictly better time-complexity results.

The boolean analog of ROABP, read once ordered branching programs (ROBP) have been studied extensively, with regard to the RL vs. L question. For ROBP, a pseudorandom generator (PRG) with seed length $O(\log^2 n)$ ($n^{O(\log n)}$ size sample set) is known in the case of known variable order [Nis90]. This is analogous to the [FS13] result for known order ROABP. On the other hand, in the unknown order case, the best known seed length is of size $n^{1/2+o(1)}$) ($2^{n^{1/2+o(1)}}$ size sample set) [IMZ12]. One can ask: Can the result for the unknown order case be matched with the known order case in the boolean setting as well. Recently, there has been a partial progress in this direction by [SVW14].

The PIT problem has also been studied for various restricted classes of circuits. One such class is depth-3 circuits. Our second result is about a special case of this class. A depth-3 circuit is usually defined as a $\Sigma\Pi\Sigma$ circuit: The circuit gates are in three layers, the top layer has an output gate which is $+$, second layer has all $\times$ gates and the last layer has all $+$ gates. In other words, the polynomial computed by a $\Sigma\Pi\Sigma$ circuit is of the form $C(\overline{x}) = \sum_{i=1}^{k} a_i \prod_{j=1}^{n_i} \ell_{ij}$, where n_i is the number of input lines to the i-th product gate and ℓ_{ij} is a linear polynomial of the form $b_0 + \sum_{r=1}^{n} b_r x_r$. An efficient solution for depth-3 PIT is still not known. Recently, it was shown by Gupta et al. [GKKS13], that depth-3 circuits are almost as powerful as general circuits. A polynomial time hitting-set for a depth-3 circuit implies a quasi-poly-time hitting-set for general circuits. Till now, for depth-3 circuits, efficient PIT is known when the top fan-in is assumed to be constant [DS07, KS07, KS09, KS11, SS11, SS12, SS13] and for certain other restrictions [Sax08, SSS13, ASSS12].

On the other hand, there are exponential lower bounds for depth-3 *multilinear* circuits [RY09]. Since there is a connection between lower bounds and PIT [Agr05], we can hope that solving PIT for depth-3 multilinear circuits should also be feasible. This should also lead to new tools for general depth-3.

A polynomial is said to be multilinear if the degree of every variable in every term is at most 1. The circuit $C(\overline{x})$ is a multilinear circuit if the polynomial computed at every gate is multilinear. A polynomial time algorithm is known only for a sub-class of multilinear depth-3 circuits, called *depth-3 set-multilinear circuits*. This algorithm is due to Raz and Shpilka [RS05] and is whitebox. In a depth-3 multilinear circuit, since every product gate computes a multilinear polynomial, a variable occurs in at most one of the n_i linear polynomials input to it. Thus, each product gate naturally induces a *partition* of the variables, where each *color* (i.e. part) of the partition contains the variables present in a linear polynomial ℓ_{ij}. Further, if the partitions induced by all

the k product gates are the same then the circuit is called a depth-3 set-multilinear circuit.

Agrawal et al. [ASS13] gave a quasi-polynomial time blackbox algorithm for the class of depth-3 set-multilinear circuits. But till now, no subexponential time test (not even whitebox) was known even for sum of two set-multilinear circuits. We give a subexponential time whitebox PIT for sum of constantly many set-multilinear circuits.

THEOREM 2. *Let $C(\mathbf{x})$ be a n-variate polynomial, which is a sum of c set-multilinear depth-3 circuits, each having top fan-in k. Then there is a $n^{O(2^{c-1}n^{1-\epsilon}\log k)}$-time whitebox test for C, where $\epsilon := 1/2^{c-1}$.*

To achieve this, we define a new class of circuits, as a tool, called *multilinear depth-3 circuits with Δ-distance*. A multilinear depth-3 circuit has Δ-distance if there is an ordering on the partitions induced by the product gates, say $(\mathbb{P}_1, \mathbb{P}_2, \ldots, \mathbb{P}_k)$, such that for any color in the partition $\mathbb{P}_i$, there exists a set of $\leq (\Delta - 1)$ other colors in $\mathbb{P}_i$ such that the set of variables in the union of these $\leq \Delta$ colors are *exactly* partitioned in the upper partitions, i.e. $\{\mathbb{P}_1, \mathbb{P}_2, \ldots, \mathbb{P}_{i-1}\}$. As we will see, such sets of Δ colors form equivalence classes of the colors at partition $\mathbb{P}_i$. We call them friendly neighborhoods and they help us in identifying subcircuits. Intuitively, the distance measures how far away are the partitions from a mere *refinement* sequence of partitions, $\mathbb{P}_1 \leq \mathbb{P}_2 \leq \cdots \leq \mathbb{P}_k$. A refinement sequence of partitions will have distance 1. On the other hand, general multilinear depth-3 circuits can have at most n-distance.

As it turns out, a polynomial computed by a depth-3 Δ-distance circuit (top fan-in k) can also be computed by a width-$O(kn^\Delta)$ ROABP (see Lemma 14). Thus, we get a poly$(nk)^{\Delta \log n}$-time hitting set for this class, from Theorem 1. Next, we use a general result about finding a hitting set for a class m-*base-sets*-C, if a hitting set is known for class C. A polynomial is in m-base-sets-C, if there exists a partition of the variables into m base sets such that restricted to each base set (treat other variables as field constants), the polynomial is in class C. We combine these two tools to prove Theorem 2. We show that a sum of constantly many set-multilinear circuits falls into the class m-base-sets-Δ-distance, for $m\Delta = o(n)$.

Agrawal et al. [AGKS13] had achieved *rank concentration*, which implies a hitting set, for the class m-base-sets-Δ-distance, but through complicated proofs. On the other hand, this work gives only a hitting set for the same class, but with the advantage of simplied proofs.

Our third result deals again with arithmetic branching programs. The results of [BOC92] and [SSS09] show that the constant-width ABP is already a strong model. Here, we study constant-width ABP with some natural restrictions.

We consider a class of ROABPs where all the matrices in the matrix product, except the left-most and the right-most matrices, are invertible. We give a blackbox test for this class of ROABP. In contrast to [FSS14] and our Theorem 1, this test works in *polynomial time* if the dimension of the matrices is constant.

Note that the class of ABP, where the factor matrices are invertible, is quite powerful, as Ben-Or and Cleve [BOC92] actually reduce formulas to width-3 ABP with *invertible* factors. Saha, Saptharishi and Saxena [SSS09] reduce depth-3 circuits to width-2 ABP with invertible factors. But the constraints of invertibility and read-once together seem to restrict the computing power of ABP. Interestingly, an analogous class of read-once boolean branching programs called *permutation branching programs* has been studied recently [KNP11, De11, Ste12]. These works give PRG for this class

(for constant width) with seed-length $O(\log n)$, in the known variable order case. In other words, they give polynomial size sample set which can fool these programs. For the unknown variable order case, Reingold et al. [RSV13] gave a PRG with seed-length $O(\log^2 n)$. Our polynomial size hitting sets for the arithmetic setting work for any unknown variable order. Hence, it is better as compared to the currently known results for the boolean case.

THEOREM 3 (Informal version). *Let* $C(\overline{x}) = D_0^\top (\prod_{i=1}^d D_i) D_{d+1}$ *be a polynomial such that* $D_0 \in \mathbb{F}^w[x_{j_0}]$ *and* $D_{d+1} \in \mathbb{F}^w[x_{j_{d+1}}]$ *and for all* $i \in [d]$, $D_i \in \mathbb{F}^{w \times w}[x_{j_i}]$ *is an invertible matrix (order of the variables is unknown). Let the degree bound on* D_i *be* δ *for* $0 \le i \le d+1$. *Then there is a* $\mathrm{poly}((\delta n)^{w^2})$*-time hitting-set for* $C(\overline{x})$.

The proof technique here is very different from the first two theorems (here we show *rank concentration* over a *non-commutative* algebra, see the proof idea in Section 5). Our algorithm works even when the factor matrices have their entries as general sparse polynomials (still over disjoint sets of variables) instead of univariate polynomials (see the detailed version in Section 5). Running time in this case grows to quasi-polynomial (but is still better than Theorem 1 in several interesting cases).

If the matrices are 2×2, then we do not need the assumption of invertibility (see Theorem 34, Section 5.3). So, for width-2 ROABP our results are strictly stronger than [FSS14] and our Theorem 1. Here again, there is a comparable result in the boolean setting. PRG with seed-length $O(\log n)$ (polynomial size sample set) are known for width-2 ROBP [BDVY13].

2. Preliminaries.

Hitting Set. A set of points $\mathcal{H}$ is called a hitting set for a class C of polynomials if for any nonzero polynomial P in C, there exists a point in $\mathcal{H}$ where P evaluates to a nonzero value. An $f(n)$-time hitting set would mean that the hitting set can be generated in time $f(n)$ for input size n.

2.1. Notation. $\mathbb{Z}_+$ denotes the set $\mathbb{N} \cup \{0\}$. $[n]$ denotes the set $\{1, 2, \ldots, n\}$. $[[n]]$ denotes the set $\{0, 1, \ldots, n\}$. $\mathbf{x}$ will denote a set of variables. For a set of n variables $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ and for an exponent $\mathbf{e} = (e_1, e_2, \ldots, e_n) \in \mathbb{Z}_+^n$, $\mathbf{x}^{\mathbf{e}}$ will denote the monomial $\prod_{i=1}^n x_i^{e_i}$. The *support* of a monomial is the set of variables that have degree ≥ 1 in that monomial. The *support size* of the monomial is the cardinality of its support. A polynomial is called *s-sparse* if there are s monomials in it with nonzero coefficients. For a polynomial P, the coefficient of the monomial m in $P(\mathbf{x})$ is denoted by $\mathrm{coef}_P(m)$.

$\mathbb{F}^{m \times n}$ represents the set of all $m \times n$ matrices over the field $\mathbb{F}$. $\mathbb{M}_{m \times m}(\mathbb{F})$ will denote the algebra of $m \times m$ matrices over the field $\mathbb{F}$. Let $\mathbb{A}_k(\mathbb{F})$ be any k-dimensional algebra over the field $\mathbb{F}$. For any two elements $A = (a_1, a_2, \ldots a_k) \in \mathbb{A}_k(\mathbb{F})$ and $B = (b_1, b_2, \ldots b_k) \in \mathbb{A}_k(\mathbb{F})$ (having a natural basis representation in mind), their dot product is defined as $A \cdot B = \sum_{i=1}^n a_k b_k$; and the product AB will denote the product in the algebra $\mathbb{A}_k(\mathbb{F})$.

$\mathrm{Part}(S)$ denotes the set of all possible partitions of the set S. Elements in a partition are called *colors* (or parts).

2.2. Arithmetic Branching Programs. An ABP is a directed graph with $d + 1$ layers of vertices $\{V_0, V_1, \ldots, V_d\}$ and a start node u and an end node t such that the edges are only going from u to V_0, V_{i-1} to V_i for any $i \in [d]$, V_d to t. A width-w ABP has $|V_i| \le w$ for all $i \in [[d]]$. Let the set of nodes in V_i be $\{v_{i,j} \mid j \in [w]\}$. All the edges in the graph have weights from $\mathbb{F}[\mathbf{x}]$, for some field $\mathbb{F}$. As a convention, the edges going from s and coming to t are assumed to have weights from the field $\mathbb{F}$.

For an edge e, let us denote its weight by $W(e)$. For a path p from u to t, its weight $W(p)$ is defined to be the product of weights of all the edges in it, i.e. $\prod_{e \in p} W(e)$. Consider the polynomial $C(\mathbf{x}) = \sum_{p \in \text{paths}(u,t)} W(p)$ which is the sum of the weights of all the paths from u to t. This polynomial $C(\mathbf{x})$ is said to be computed by the ABP.

It is easy to see that this polynomial is the same as $S^\top (\prod_{i=1}^{d} D_i) T$, where $S, T \in \mathbb{F}^w$ and D_i is a $w \times w$ matrix for $1 \leq i \leq d$ such that

$$S(\ell) = W(u, v_{0,\ell}) \text{ for } 1 \leq \ell \leq w$$
$$D_i(k, \ell) = W(v_{i-1,k}, v_{i,\ell}) \text{ for } 1 \leq \ell, k \leq w \text{ and } 1 \leq i \leq d$$
$$T(k) = W(v_{d,k}, t) \text{ for } 1 \leq k \leq w$$

ROABP. An ABP is called a *read once oblivious ABP (ROABP)* if the edge weights in the different layers are univariate polynomials in distinct variables. Formally, the entries in D_i come from $\mathbb{F}[x_{\pi(i)}]$ for all $i \in [d]$, where π is a permutation on the set $[d]$.

sparse-factor ROABP. We call the ABP a *sparse-factor ROABP* if the edge weights in different layers are sparse polynomials in disjoint sets of variables. Formally, if there exists an unknown partition of the variable set $\mathbf{x}$ into d sets $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_d\}$ such that $D_i \in \mathbb{F}^{w \times w}[\mathbf{x}_i]$ is a s-sparse polynomial, for all $i \in [d]$, then the corresponding ROABP is called a *s-sparse-factor* ROABP. It is read once in the sense that in the corresponding ABP, any particular variable contributes to at most one edge on any path.

2.3. Kronecker Map. We will often use a weight function on the variables which separates a desired set of monomials. Let $w \colon \mathbf{x} \to \mathbb{N}$ be a weight function on the variables. Consider its natural extension to the set of all monomials $w \colon \mathbb{Z}_+^n \to \mathbb{N}$ as follows: $w(\Pi_{i=1}^{n} x_i^{\gamma_i}) = \sum_{i=1}^{n} \gamma_i w(x_i)$, where $\gamma_i \in \mathbb{Z}_+$, $\forall i \in [n]$.

LEMMA 4 (Efficient Kronecker map [Kro82, Agr05]). *Let $\mathcal{M}$ be the set of all monomials in n variables $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ with maximum individual degree δ. Let A be a set of pairs of monomials from $\mathcal{M}$. Then there exists a (constructible) set of N-many weight functions $w \colon \mathbf{x} \to [1, \ldots, N \log N]$, such that at least one of them separates all the pairs in A, i.e. for any $(m, m') \in A$, $w(m) \neq w(m')$, where $N := O(n|A| \log(\delta + 1))$.*

Proof. Since we want to separate the n-variate monomials with maximum individual degree δ, we use the naïve Kronecker map $W \colon x_i \mapsto (\delta + 1)^{i-1}$ for all $i \in [n]$. It can be easily seen that W will give distinct weights to any two monomials (with maximum individual degree δ). But, the weights given by W are exponentially high.

So, we take the weight function W modulo p, for many small primes p. Each prime p leads to a different weight function. That is our set of candidate weight functions. We need to bound the number N of primes that ensures that at least one of the weight functions separates all the monomial pairs in A. We choose the smallest N primes, say $\mathcal{P}$ is the set. By the effective version of the Prime Number Theorem, the highest value in the set $\mathcal{P}$ is $N \log N$.

To bound the number N of primes: We want a $p \in \mathcal{P}$ such that $\forall (m, m') \in A$, $W(m) - W(m') \not\equiv 0 \pmod{p}$. Which means,

$$\exists p \in \mathcal{P}, \ p \nmid \prod_{(m,m') \in A} (W(m) - W(m')).$$

In other words,

$$\prod_{p\in\mathcal{P}} p \nmid \prod_{(m,m')\in A} (W(m) - W(m'))\,.$$

This can be ensured by setting $\prod_{p\in\mathcal{P}} p > \prod_{(m,m')\in A} (W(m) - W(m'))$. There are $|A|$ such monomial pairs and each $W(m) < n\delta(\delta+1)^{n-1}$. Also, $\prod_{p\in\mathcal{P}} p > 2^N$. Hence, $N = O(n|A|\log(\delta + 1))$ suffices. $\square$

3. Hitting set for ROABP: Theorem 1. Like [ASS13] and [FSS14], we work with the vector polynomial. I.e. for a polynomial computed by a width-w ROABP, $C(\mathbf{x}) = S^\top(\prod_{i=1}^{d} D_i)T$, we see the product $D := \prod_{i=1}^{d} D_i$ as a polynomial over the matrix algebra $\mathbb{M}_{w\times w}(\mathbb{F})$. We can write the polynomial $C(\mathbf{x})$ as the dot product $R \cdot D$, where $R = ST^\top$. The vector space spanned by the coefficients of $D(\mathbf{x})$ is called the coefficient space of $D(\mathbf{x})$. This space will have dimension at most w^2. We essentially try to construct a small set of vectors, by evaluating $D(\mathbf{x})$, which can span the coefficient space of $D(\mathbf{x})$. Clearly, if $C \neq 0$ then the dot product of R with at least one of these spanning vectors will be nonzero. And thus, we get a hitting set.

Unlike [ASS13] and [FSS14], we directly work with the original polynomial $D(\mathbf{x})$, instead of shifting it and breaking it into subcircuits. Our approach for finding the hitting set is to come up with a weight function on the variables which can *isolate a basis* for the coefficients of the polynomial $D(\mathbf{x})$. This can be seen as a generalization of isolating a monomial for a polynomial in $\mathbb{F}[\mathbf{x}]$, which is a usual technique for PIT (e.g. sparse PIT [KS01]).

We present our results for polynomials over arbitrary algebra. Let $\mathbb{A}_k(\mathbb{F})$ be a k-dimensional algebra over the field $\mathbb{F}$. Let $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ be a set of variables and let $D(\mathbf{x})$ be a polynomial in $\mathbb{A}_k(\mathbb{F})[\mathbf{x}]$ with highest individual degree δ. Let $\mathcal{M}$ denote the set of all monomials over the variable set $\mathbf{x}$ with highest individual degree δ.

Now, we will define a basis isolating weight assignment for a polynomial $D \in \mathbb{A}_k(\mathbb{F})[\mathbf{x}]$ which would lead to a hitting set for the polynomial $C \in \mathbb{F}[\mathbf{x}]$, where $C = R \cdot D$, for some $R \in \mathbb{A}_k(\mathbb{F})$.

DEFINITION 5 (Basis Isolating Weight Assignment). *A weight function $w\colon \mathbf{x} \to \mathbb{N}$ is called a basis isolating weight assignment for a polynomial $D(\mathbf{x}) \in \mathbb{A}_k(\mathbb{F})[\mathbf{x}]$ if there exists a set of monomials $S \subseteq \mathcal{M}$ $(k' := |S| \leq k)$ whose coefficients form a basis for the coefficient space of $D(\mathbf{x})$, such that*

- *for any $m, m' \in S$, $w(m) \neq w(m')$ and*
- *for any monomial $m \in \mathcal{M} \setminus S$,*

$$\operatorname{coef}_D(m) \in \operatorname{span}\{\operatorname{coef}_D(m') \mid m' \in S,\ w(m') < w(m)\}\,.$$

The above definition is equivalent to saying that there exists a *unique minimum weight basis* (according to the weight function w) among the coefficients of D, and also the basis monomials have distinct weights. We skip the easy proof for this equivalence, as we will not need it. Note that a weight assignment, which gives distinct weights to all the monomials, is indeed a basis isolating weight assignment. But, it will involve exponentially large weights. To, find an efficient weight assignment one must use some properties of the given circuit. First, we show how such a weight assignment would lead to hitting set. We will actually show that it isolates a monomial in $C(\mathbf{x})$.

LEMMA 6. *Let $w\colon \mathbf{x} \to \mathbb{N}$ is a basis isolating weight assignment for a polynomial $D(\mathbf{x}) \in \mathbb{A}_k(\mathbb{F})[\mathbf{x}]$. And let $C = R \cdot D$ be a nonzero polynomial, for some $R \in \mathbb{A}_k(\mathbb{F})$.*

Then, after the substitution $x_i = t^{w(x_i)}$ for all $i \in [n]$, the polynomial C remains nonzero, where t is an indeterminate.

Proof. Let $D_m \in \mathbb{A}_k(\mathbb{F})$ denote the coefficient $\operatorname{coef}_D(m)$. It is easy to see that after the mentioned substitution, the new polynomial $C'(t)$ is equal to $\sum_{m \in \mathcal{M}}(R \cdot D_m)t^{w(m)}$.

Let us say that $S \subset \mathcal{M}$ is the set of monomials whose coefficients form the isolated basis for D. According to the definition of the basis isolating weight assignment, for any monomial $m \in \mathcal{M} \setminus S$,

$$D_m \in \operatorname{span}\{D_{m'} \mid m' \in S, \ w(m') < w(m)\}. \tag{1}$$

First, we claim that $\exists m' \in S$ such that $R \cdot D_{m'} \neq 0$. For the sake of contradiction, let us assume that $\forall m' \in S, \ R \cdot D_{m'} = 0$. Taking the dot product with R on both the sides of Equation (1), we get that for any monomial $m \in \mathcal{M} \setminus S$,

$$R \cdot D_m \in \operatorname{span}\{R \cdot D_{m'} \mid m' \in S, \ w(m') < w(m)\}.$$

Hence, $R \cdot D_m = 0, \ \forall m \in \mathcal{M}$. That means $C(\mathbf{x}) = 0$, which contradicts our assumption.

Now, let m^* be the minimum weight monomial in S whose coefficient gives a nonzero dot product with R, i.e. $m^* = \arg\min_{m \in S}\{w(m) \mid R \cdot D_m \neq 0\}$. There is a unique such monomial in S because all the monomials in S have distinct weights.

We claim that $\operatorname{coef}_{C'}(t^{w(m^*)}) \neq 0$ and hence $C'(t) \neq 0$. To see this, consider any monomial m, other than m^*, with $w(m) = w(m^*)$. The monomial m has to be in the set $\mathcal{M} \setminus S$, as the monomials in S have distinct weights. From Equation (1),

$$D_m \in \operatorname{span}\{D_{m'} \mid m' \in S, \ w(m') < w(m^*)\}.$$

Taking dot product with R on both the sides we get,

$$R \cdot D_m \in \operatorname{span}\{R \cdot D_{m'} \mid m' \in S, \ w(m') < w(m^*)\}.$$

But, by the choice of m^*, $R \cdot D_{m'} = 0$, for any $m' \in S$ with $w(m') < w(m^*)$. Hence, $R \cdot D_m = 0$, for any $m \neq m^*$ with $w(m) = w(m^*)$.

So, the coefficient $\operatorname{coef}_{C'}(t^{w(m^*)})$ can be written as

$$\sum_{\substack{m \in \mathcal{M} \\ w(m)=w(m^*)}} R \cdot D_m = R \cdot D_{m^*},$$

which, we know, is nonzero. □

To construct a hitting set for C, we can try many possible field values of t. The number of such values needed will be the degree of C after the substitution, which is at most $(n\delta \max_i w(x_i))$. Hence, the cost of the hitting set is dominated by the *cost of the weight function*, i.e. the maximum weight given to any variable and the time taken to construct the weight function.

In the next step, we show that such a basis isolating weight assignment can indeed be found for a sparse-factor ROABP, but with cost quasi-polynomial in the input size. First, we make the following observation that it suffices that the coefficients of the monomials not in S, linearly depend on any coefficients with strictly smaller weight, not necessarily coming from S.

OBSERVATION 7. *If, for a polynomial $D \in \mathbb{A}_k(\mathbb{F})[\mathbf{x}]$, there exists a weight function $w \colon \mathbf{x} \to \mathbb{N}$ and a set of monomials $S \subseteq \mathcal{M}$ $(k' := |S| \leq k)$ such that for any monomial $m \in \mathcal{M} \setminus S$,*

$$\operatorname{coef}_D(m) \in \operatorname{span}\{\operatorname{coef}_D(m') \mid m' \in \mathcal{M}, \ w(m') < w(m)\}.$$

then we can also conclude that for any monomial $m \in \mathcal{M} \setminus S$,

$$\text{coef}_D(m) \in \text{span}\{\text{coef}_D(m') \mid m' \in S, \ w(m') < w(m)\}.$$

Proof. We are given that for any monomial $m \in \overline{S} := \mathcal{M} \setminus S$,

$$\text{coef}_D(m) \in \text{span}\{\text{coef}_D(m') \mid m' \in \mathcal{M}, \ w(m') < w(m)\}.$$

Any coefficient $\text{coef}_D(m')$ on the right hand side of this equation, which corresponds to an index in $\overline{S}$, can be replaced with some other coefficients, which have further smaller weight. If we keep doing this, we will be left with the coefficients only corresponding to the set S, because in each step we are getting smaller and smaller weight coefficients.
□

In our construction of the weight function, we will create the set $\overline{S} := \mathcal{M} \setminus S$ incrementally, i.e. in each step we will make more coefficients depend on strictly smaller weight coefficients. Finally, we will be left with only k' (the rank of the coefficient space of D) many coefficients in S. We present the result for an arbitrary k-dimensional algebra $\mathbb{A}_k(\mathbb{F})$, instead of just the matrix algebra.

LEMMA 8 (Weight Construction). *Let $\mathbf{x}$ be given by a union of d disjoint sets of variables $\mathbf{x}_1 \sqcup \mathbf{x}_2 \sqcup \cdots \sqcup \mathbf{x}_d$, with $|\mathbf{x}| = n$. Let $D(\mathbf{x}) = P_1(\mathbf{x}_1)P_2(\mathbf{x}_2)\cdots P_d(\mathbf{x}_d)$, where $P_i \in \mathbb{A}_k(\mathbb{F})[\mathbf{x}_i]$ is a sparsity-s, individual degree-δ polynomial, for all $i \in [d]$. Then, we can construct a basis isolating weight assignment for $D(\mathbf{x})$ with the cost being $(\text{poly}(k,s,n,\delta))^{\log d}$.*

Proof. In our construction, the final weight function w will be a combination of $(\log d + 1)$-many different weight functions, say $(w_0, w_1, \ldots, w_{\log d})$. Let us say, their precedence is decreasing from left to right, i.e. w_0 has the highest precedence and $w_{\log d}$ has the lowest precedence. As mentioned earlier, we will build the set $\overline{S}$ (the set of monomials whose coefficients are in the span of strictly smaller weight coefficients than themselves) incrementally in $(\log d + 1)$ steps, using weight function w_i in the $(i+1)$-th step.

Iteration 0: Let $\mathcal{M}_{0,1}, \mathcal{M}_{0,2}, \ldots, \mathcal{M}_{0,d}$ be the sets of monomials and $\mathcal{C}_{0,1}, \mathcal{C}_{0,2}, \ldots, \mathcal{C}_{0,d}$ be the sets of coefficients in the polynomials $P_1, P_2, \ldots, P_d$ respectively.

NOTATION. *The product of two sets of monomials $\mathcal{M}_1$ and $\mathcal{M}_2$ is defined as $\mathcal{M}_1 \times \mathcal{M}_2 = \{m_1 m_2 \mid m_1 \in \mathcal{C}_1, \ m_2 \in \mathcal{C}_2\}$. The product of any two sets of coefficients $\mathcal{C}_1$ and $\mathcal{C}_2$ is defined as $\mathcal{C}_1 \times \mathcal{C}_2 = \{c_1 c_2 \mid c_1 \in \mathcal{C}_1, \ c_2 \in \mathcal{C}_2\}$.*

The crucial property of the polynomial D is that the set of coeffcients in D, $\mathcal{C}_0$, is just the product $\mathcal{C}_{0,1} \times \mathcal{C}_{0,2} \times \cdots \times \mathcal{C}_{0,d}$. Similary, the set of all the monomials in D, say $\mathcal{M}_0$, can be viewed as the product $\mathcal{M}_{0,1} \times \mathcal{M}_{0,2} \times \cdots \times \mathcal{M}_{0,d}$. Let $m := m_a m_{a+1} \cdots m_b$ be a monomial, where $1 \leq a \leq b \leq d$ and $m_j \in \mathcal{M}_{0,j}$, for $a \leq j \leq b$. Then D_m will denote the coefficient $\text{coef}_{P_a}(m_a) \, \text{coef}_{P_{a+1}}(m_{a+1}) \cdots \text{coef}_{P_b}(m_b)$.

Let us fix $w_0 \colon \mathbf{x} \to \mathbb{N}$ to be a weight function on the variables which gives distinct weights to all the s monomials in $\mathcal{M}_{0,i}$, for each $i \in [d]$. As w_0 assigns distinct weights to these monomials, so does the weight function w.

For each P_i we do the following:
- arrange the coefficients in $\mathcal{C}_{0,i}$ in increasing order of their weight according to w (or equivalently, according to w_0),
- choose a maximal set of linearly independent coefficients, in a greedy manner, going from lower weights to higher weights.

The fact that the weight functions $w_1, w_2, \ldots, w_{\log d}$ are not defined yet does not matter because w_0 has the highest precedence. The total order given to the monomials in

$\mathcal{M}_{0,i}$ by w_0 is the same as given by w, irrespective of what the functions $w_1, \ldots, w_{\log d}$ are chosen to be.

This gives us a basis for the coefficients of P_i, say $\mathcal{C}'_{0,i}$. Let $\mathcal{M}'_{0,i}$ denote the monomials in P_i corresponding to these basis coefficients. From the construction of the basis, it follows that for any monomial $m \in \mathcal{M}_{0,i} \setminus \mathcal{M}'_{0,i}$,

$$D_m \in \text{span}\{D_{m'} \mid m' \in \mathcal{M}'_{0,i}, \ w(m') < w(m)\}. \tag{2}$$

Now, consider any monomial $m \in \mathcal{M}$ which is not present in the set $\mathcal{M}'_0 := \mathcal{M}'_{0,1} \times \mathcal{M}'_{0,2} \times \cdots \times \mathcal{M}'_{0,d}$. Let $m = m_1 m_2 \cdots m_d$, where $m_i \in \mathcal{M}_{0,i}$ for all $i \in [d]$. We know that for at least one $j \in [d]$, $m_j \in \mathcal{M}_{0,j} \setminus \mathcal{M}'_{0,j}$. Then using Equation (2) we can write the following about $D_m = D_{m_1} D_{m_2} \cdots D_{m_d}$,

$$D_m \in \text{span}\{D_{m_1} \cdots D_{m_{j-1}} D_{m'_j} D_{m_{j+1}} \cdots D_{m_d} \mid m'_j \in \mathcal{M}'_{0,j}, \ w(m'_j) < w(m_j)\}.$$

This holds, because the algebra product is bilinear. Equivalently, for any monomial $m \in \mathcal{M}_0 \setminus \mathcal{M}'_0$,

$$D_m \in \text{span}\{D_{m'} \mid m' \in \mathcal{M}_0, \ w(m') < w(m)\}.$$

This is true because

$$w(m_1) + \cdots + w(m'_j) + \cdots + w(m_d) < w(m_1) + \cdots + w(m_j) + \cdots + w(m_d) = w(m).$$

Hence, all the monomials in $\mathcal{M}_0 \setminus \mathcal{M}'_0$ can be put into $\overline{S}$, i.e. their corresponding coefficients depend on strictly smaller weight coefficients.

Iteration 1: Now, let us consider monomials in the set $\mathcal{M}'_0 = \mathcal{M}'_{0,1} \times \mathcal{M}'_{0,2} \times \cdots \times \mathcal{M}'_{0,d}$. Let the corresponding set of coefficients be $\mathcal{C}'_0 := \mathcal{C}'_{0,1} \times \mathcal{C}'_{0,2} \times \cdots \times \mathcal{C}'_{0,d}$. Since, the underlying algebra $\mathbb{A}_k(\mathbb{F})$ has dimension at most k and the coefficients in $\mathcal{C}'_{0,i}$ form a basis for $\mathcal{C}_{0,i}$, $|\mathcal{M}'_{0,i}| \leq k$, for all $i \in [d]$. In the above product, let us make $d/2$ disjoint pairs of consecutive terms, and for each pair, multiply the two terms in it. Putting it formally, let us define $\mathcal{C}_{1,j}$ to be the product $\mathcal{C}'_{0,2j-1} \times \mathcal{C}'_{0,2j}$ and similarly $\mathcal{M}_{1,j} := \mathcal{M}'_{0,2j-1} \times \mathcal{M}'_{0,2j}$, for all $j \in [d/2]$ (if d is odd, we can make it even by multiplying the identity element of $\mathbb{A}_k(\mathbb{F})$ in the end). Now, let $\mathcal{C}_1 := \mathcal{C}'_0 = \mathcal{C}_{1,1} \times \mathcal{C}_{1,2} \times \cdots \times \mathcal{C}_{1,d_1}$, and $\mathcal{M}_1 := \mathcal{M}'_0 = \mathcal{M}_{1,1} \times \mathcal{M}_{1,2} \times \cdots \times \mathcal{M}_{1,d_1}$, where $d_1 := d/2$. For any $i \in [d_1]$, $\mathcal{M}_{1,i}$ has at most k^2 monomials.

Now, we fix the weight function $w_1 : \mathbf{x} \to \mathbb{N}$ such that it gives distinct weights to all the monomials in $\mathcal{M}_{1,i}$, for each $i \in [d_1]$. As w_1 separates these monomials, so does the weight function w. Now, we repeat the same procedure of constructing a basis in a greedy manner for $\mathcal{C}_{1,i}$ according to the weight function w, for each $i \in [d_1]$. Let the basis coefficients for $\mathcal{C}_{1,i}$ be $\mathcal{C}'_{1,i}$ and corresponding monomials be $\mathcal{M}'_{1,i}$.

As argued before, any coefficient in $\mathcal{C}_1$, which is outside the set $\mathcal{C}'_1 := \mathcal{C}'_{1,1} \times \mathcal{C}'_{1,2} \times \cdots \times \mathcal{C}'_{1,d_1}$, is in the span of strictly smaller weight (than itself) coefficients. So, we can also put the corresponding monomials $\mathcal{M}_1 \setminus \mathcal{M}'_1$ in $\overline{S}$ where $\mathcal{M}'_1 := \mathcal{M}'_{1,1} \times \mathcal{M}'_{1,2} \times \cdots \times \mathcal{M}'_{1,d_1}$.

Iteration r: We keep repeating the same procedure for $(\log d + 1)$-many rounds. After round r, say the set of monomials we are left with is given by the product $\mathcal{M}'_{r-1} = \mathcal{M}'_{r-1,1} \times \mathcal{M}'_{r-1,2} \times \cdots \times \mathcal{M}'_{r-1,d_{r-1}}$, where $\mathcal{M}_{r-1,i}$ has at most k monomials, for each $i \in [d_{r-1}]$ and $d_{r-1} = d/2^{r-1}$. In the above product, we make $d_{r-1}/2$ disjoint pairs of consecutive terms, and multiply the two terms in each pair. Let us say we get $\mathcal{M}_r := \mathcal{M}'_{r-1} = \mathcal{M}_{r,1} \times \mathcal{M}_{r,2} \times \cdots \times \mathcal{M}_{r,d_r}$, where $d_r = d_{r-1}/2$. Say, the

corresponding set of coefficients is given by $\mathcal{C}_r = \mathcal{C}_{r,1} \times \mathcal{C}_{r,2} \times \cdots \times \mathcal{C}_{r,d_r}$. Note that $|\mathcal{M}_{r,i}| \leq k^2$, for each $i \in [d_r]$.

We fix the weight function w_r such that it gives distinct weights to all the monomials in the set $\mathcal{M}_{r,i}$, for each $i \in [d_r]$. We once again mention that fixing of w_r does not affect the greedy basis constructed in earlier rounds and hence the monomials which were put in the set $\overline{S}$, because w_r has less precedence than any $w_{r'}$, for $r' < r$.

For each $\mathcal{C}_{r,i}$, we construct a basis in a greedy manner going from lower weight to higher weight (according to the weight function w). Let this set of basis coefficients be $\mathcal{C}'_{r,i}$ and corresponding monomials be $\mathcal{M}'_{r,i}$, for each $i \in [d_r]$. Let $\mathcal{C}'_r := \mathcal{C}'_{r,1} \times \mathcal{C}'_{r,2} \times \cdots \times \mathcal{C}'_{r,d_r}$ and $\mathcal{M}'_r := \mathcal{M}'_{r,1} \times \mathcal{M}'_{r,2} \times \cdots \times \mathcal{M}'_{r,d_r}$. Arguing similar as before we can say that each coefficient in $\mathcal{C}_{r,i} \setminus \mathcal{C}'_{r,i}$ is in the span of strictly smaller weight coefficients (from $\mathcal{C}'_{r,i}$) than itself. Hence, the same can be said about any coefficient in the set $\mathcal{C}_r \setminus \mathcal{C}'_r$. So, all the monomials in the set $\mathcal{M}_r \setminus \mathcal{M}'_r$ can be put into $\overline{S}$. Now, we are left with monomials $\mathcal{M}'_r = \mathcal{M}'_{r,1} \times \mathcal{M}'_{r,2} \times \cdots \times \mathcal{M}'_{r,d_r}$ for the next round.

Iteration $\log d$: As in each round, the number of terms in the product gets halved, after $\log d$ rounds we will be left with just one term, i.e. $\mathcal{M}_{\log d} = \mathcal{M}'_{\log d-1,1} \mathcal{M}'_{\log d-1,2} = \mathcal{M}_{\log d,1}$. Now, we will fix the function $w_{\log d}$ which separates all the monomials in $\mathcal{M}_{\log d,1}$. By arguments similar as above, we will be finally left with at most k' monomials in S, which will all have distinct weights. It is clear that for every monomial in $\overline{S}$, its coefficient will be in the span of strictly smaller weight coefficients than itself.

Now, let us look at the cost of this weight function. In the first round, w_0 needs to separate at most $O(ds^2)$ many pairs of monomials. For each $1 \leq r \leq \log d$, w_r needs to separate at most $O(dk^4)$ many pairs of monomials. From Lemma 4, to construct w_r, for any $0 \leq r \leq \log d$, one needs to try $\text{poly}(k,s,n,\delta)$-many weight functions each having highest weight at most $\text{poly}(k,s,n,\delta)$ (as d is bounded by n). To get the correct combination of the weight functions $(w_0, w_1, \ldots, w_{\log d})$ we need to try all possible combinations of these polynomially many choices for each w_r. Thus, we have to try $(\text{poly}(k,s,n,\delta))^{\log d}$ many combinations.

To combine these weight functions we can choose a large enough number B (greater than the highest weight a monomial can get in any of the weight functions), and define $w := w_0 B^{\log d} + w_1 B^{\log d-1} + \cdots + w_{\log d}$. The choice of B ensures that the different weight functions cannot interfere with each other, and they also get the desired precedence order.

The highest weight a monomial can get from the weight function w would be $(\text{poly}(k,s,n,\delta))^{\log d}$. Thus, the cost of w remains $(\text{poly}(k,s,n,\delta))^{\log d}$. □

Combining Lemma 8 with Observation 7 and Lemma 6, we can get a hitting set for ROABP.

THEOREM 1 (restated). *Let $C(\mathbf{x})$ be an n-variate polynomial computed by a width-w, s-sparse-factor ROABP, with individual degree bound δ. Then there is a $\text{poly}(w,s,n,\delta)^{\log n}$-time hitting set for $C(\mathbf{x})$.*

Proof. As mentioned earlier, $C(\mathbf{x})$ can be written as $R \cdot D(\mathbf{x})$, for some $R \in \mathbb{M}_{w \times w}(\mathbb{F})$, where $D(\mathbf{x}) \in \mathbb{M}_{w \times w}(\mathbb{F})[\mathbf{x}]$. The underlying matrix algebra $\mathbb{M}_{w \times w}(\mathbb{F})$ has dimension w^2. The hitting set size will be dominated by the cost of the weight function constructed in Lemma 8. As the parameter d in Lemma 8, i.e. the number of layers in the ROABP, is bounded by n, the hitting set size will be $\text{poly}(w,s,n,\delta)^{\log n}$. □

4. Sum of constantly many set-multilinear circuits: Theorem 2. To find a hitting set for a sum of constantly many set-multilinear circuits, we build some tools. The first is depth-3 multilinear circuits with 'small distance'. As it turns out,

a multilinear polynomial computed by a depth-3 Δ-distance circuit (top fan-in k) can also be computed by a width-$O(kn^\Delta)$ ROABP (Lemma 14). Thus, we get a poly$(nk)^{\Delta \log n}$-time hitting set for this class, from Theorem 1. Next, we use a general result about finding a hitting set for a class m-*base-sets*-C, if a hitting set is known for class C (Lemma 17). A polynomial is in m-base-sets-C, if there exists a partition of the variables into m base sets such that restricted to each base set (treat other variables as field constants), the polynomial is in class C. Finally, we show that a sum of constantly many set-multilinear circuits falls into the class m-base-sets-Δ-distance, for $m\Delta = o(n)$. Thus, we get Theorem 2.

4.1. Δ-distance circuits. Recall that each product gate in a depth-3 multilinear circuit induces a partition on the variables. Let these partitions be $\mathbb{P}_1, \mathbb{P}_2, \ldots, \mathbb{P}_k$.

DEFINITION 9 (Distance for a partition sequence). *Let* $\mathbb{P}_1, \mathbb{P}_2, \ldots, \mathbb{P}_k \in \mathrm{Part}([n])$ *be the k partitions of the variables* $\{x_1, x_2, \ldots, x_n\}$. *Then* $\mathrm{d}(\mathbb{P}_1, \mathbb{P}_2, \ldots, \mathbb{P}_k) = \Delta$ *if* $\forall i \in \{2, 3, \ldots, k\}, \forall colors\ Y_1 \in \mathbb{P}_i, \exists Y_2, Y_3, \ldots, Y_{\Delta'} \in \mathbb{P}_i\ (\Delta' \leq \Delta)$ *such that* $Y_1 \cup Y_2 \cup \cdots \cup Y_{\Delta'}$ *equals a union of some colors in* $\mathbb{P}_j, \forall j \in [i-1]$.

In other words, in every partition $\mathbb{P}_i$, each color Y_1 has a set of colors called 'friendly neighborhood', $\{Y_1, Y_2, \ldots, Y_{\Delta'}\}$, consisting of at most Δ colors, which is exactly partitioned in the 'upper partitions'. We call $\mathbb{P}_i$, an *upper* partition relative to $\mathbb{P}_j$ (and $\mathbb{P}_j$, a *lower* partition relative to $\mathbb{P}_i$), if $i < j$. For a color X_a of a partition $\mathbb{P}_j$, let $\mathrm{nbd}_j(X_a)$ denote its friendly neighborhood. The friendly neighborhood $\mathrm{nbd}_j(x_i)$ of a variable x_i in a partition $\mathbb{P}_j$ is defined as $\mathrm{nbd}_j(\mathrm{color}_j(x_i))$, where $\mathrm{color}_j(x_i)$ is the color in the partition $\mathbb{P}_j$ that contains the variable x_i.

DEFINITION 10 (Δ-distance circuits). *A multilinear depth-3 circuit C has Δ-distance if its product gates can be ordered to correspond to a partition sequence* $(\mathbb{P}_1, \ldots, \mathbb{P}_k)$ *with* $\mathrm{d}(\mathbb{P}_1, \mathbb{P}_2, \ldots, \mathbb{P}_k) \leq \Delta$.

Every depth-3 multilinear circuit is thus an n-distance circuit. A circuit with a partition sequence, where the partition $\mathbb{P}_i$ is a refinement of the partition $\mathbb{P}_{i+1}, \forall i \in [k-1]$, exactly characterizes a 1-distance circuit. All depth-3 multilinear circuits have distance between 1 and n. Also observe that the circuits with 1-distance strictly subsume set-multilinear circuits. E.g. a circuit, whose product gates induce two different partitions $\mathbb{P}_1 = \{\{1\}, \{2\}, \ldots, \{n\}\}$ and $\mathbb{P}_2 = \{\{1, 2\}, \{3, 4\}, \ldots, \{n-1, n\}\}$, has 1-distance but is not set-multilinear.

Friendly neighborhoods - To get a better picture, we ask: Given a color X_a of a partition $\mathbb{P}_j$ in a circuit $D(\mathbf{x})$, how do we find its friendly neighborhood $\mathrm{nbd}_j(X_a)$? Consider a graph G_j which has the colors of the partitions $\{\mathbb{P}_1, \mathbb{P}_2, \ldots, \mathbb{P}_j\}$, as its vertices. For all $i \in [j-1]$, there is an edge between the colors $X \in \mathbb{P}_i$ and $Y \in \mathbb{P}_j$ if they share at least one variable. Observe that if any two colors X_a and X_b of partition $\mathbb{P}_j$ are reachable from each other in G_j, then, they should be in the same neighborhood. As reachability is an equivalence relation, *the neighborhoods are equivalence classes of colors*.

Moreover, observe that for any two variables x_a and x_b, if their respective colors in partition $\mathbb{P}_j$, $\mathrm{color}_j(x_a)$ and $\mathrm{color}_j(x_b)$ are reachable from each other in G_j then their respective colors in partition $\mathbb{P}_{j+1}$, $\mathrm{color}_{j+1}(x_a)$ and $\mathrm{color}_{j+1}(x_b)$ are also reachable from each other in G_{j+1}. Hence,

OBSERVATION 11. *If at some partition, the variables x_a and x_b are in the same neighborhood, then, they will be in the same neighborhood in all of the lower partitions. I.e.* $\mathrm{nbd}_j(x_a) = \mathrm{nbd}_j(x_b) \implies \mathrm{nbd}_i(x_a) = \mathrm{nbd}_i(x_b), \forall i \geq j$. In other words, if we define a new sequence of partitions, such that the j-th partition has x_a and x_b in the same color if $\mathrm{nbd}_j(x_a) = \mathrm{nbd}_j(x_b)$, then the upper partitions are *refinements* of the

12

lower partitions.

4.1.1. Reduction to ROABP. Now, we show that any polynomial computed by a low-distance multilinear depth-3 circuit can also be computed by a small size ROABP. First we make the following observation about sparse polynomials.

OBSERVATION 12. *Any multilinear polynomial $C(\mathbf{x})$ with sparsity s can be computed by a width-s ROABP, in any variable order.*

Proof. Let $\mathcal{M}$ denote the set of monomials in C, and let C_m denote $\operatorname{coef}_C(m)$. Consider an ABP with $n+1$ layers of vertices $V_1, V_2, \ldots, V_{n+1}$ each having s vertices (one for each monomial in $\mathcal{M}$) together with a start vertex v_0 and an end vertex v_{n+2}. Let $v_{i,m}$ denote the m-th vertex of the layer V_i, for any $i \in [n+1]$ and any $m \in \mathcal{M}$.

The edge labels in the ABP are given as follows: For all $m \in \mathcal{M}$,

- The edge $(v_0, v_{1,m})$ is labelled by C_m,
- The edge $(v_{n+1,m}, v_{n+2})$ is labelled by 1,
- For all $i \in [n]$, the edge $(v_{i,m}, v_{i+1,m})$ is labelled by x_i if the monomial m contains x_i, otherwise by 1.

All other edges get labelled by 0. Clearly, the ABP constructed computes the polynomial $P(\mathbf{x})$ and it is an ROABP.

Also, note that this construction can be done with any desired variable order. □

Now, consider a depth-3 Δ-distance multilinear polynomial $P = \sum_{i=1}^{k} a_i Q_i$, where each $Q_i = \prod_{j=1}^{n_i} \ell_{ij}$ is a product of linear polynomials. We will construct an ROABP for each Q_i. We can combine these ROABPs to construct a single ROABP if they all have the same variable order. To achieve this we use the *refinement* property described above (from Observation 11).

LEMMA 13. *Let $P = \sum_{i=1}^{k} a_i Q_i$ be a polynomial computed by a Δ-distance circuit. Then we can make a width-$O(n^\Delta)$ ROABP for each Q_i, in the same variable order.*

Proof. Each Q_i is a product of linear forms in disjoint set of variables, say $Q_i = \prod_{j=1}^{n_i} \ell_{ij}$. Let the partition induced on the variable set, by the product Q_i, be $\mathbb{P}_i$, for all $i \in [k]$. Without loss of generality let the partition sequence $(\mathbb{P}_1, \mathbb{P}_2, \ldots, \mathbb{P}_k)$ have distance Δ. For each $i \in [k]$, let us define a new partition $\mathbb{P}'_i$, such that the union of colors in each neighborhood of $\mathbb{P}_i$ forms a color of $\mathbb{P}'_i$. This is a valid definition, as neighborhoods are equivalence classes of colors. From Observation 11, the partition $\mathbb{P}'_i$ is a refinement of partition $\mathbb{P}'_j$ for any $i < j$.

For a partition $\mathbb{P}$ of the variable set $\mathbf{x}$, an ordering on its colors $(c_1 < c_2 < \cdots < c_r)$ naturally induces a partial ordering on the variables, i.e. for any $x_i \in c_j$ and $x_{i'} \in c_{j'}$, $c_j < c_{j'} \implies x_i < x_{i'}$. The variables in the same color do not have any relation.

Let us say, a variable (partial) order $(<^*)$ *respects* a partition $\mathbb{P}$ with colors $\{c_1, c_2, \ldots, c_r\}$, if there exists an ordering of the colors $(c_{j_1} < c_{j_2} < \cdots < c_{j_r})$, such that its induced partial order $(<)$ on the variables can be extended to $<^*$. We claim that there exists a variable order $(<^*)$ which *respects* partition $\mathbb{P}'_i$, for all $i \in [k]$.

We build this variable order $(<^*)$ iteratively. We start with $\mathbb{P}'_k$. We give an arbitrary ordering to the colors in $\mathbb{P}'_k$, say $(c_{k,1} < c_{k,2} < \cdots < c_{k,r_k})$, which induces a partial order $(<_k)$ on the variables. For any $k > i \geq 1$, let us define a partial order $(<_i)$ inductively as follows: Let $(<_{i+1})$ be a partial order on the variables induced by an ordering on the colors of $\mathbb{P}'_{i+1}$. As mentioned earlier, the colors of $\mathbb{P}'_i$ are just further partitions of the colors of $\mathbb{P}'_{i+1}$. Hence, we can construct an ordering on the colors of $\mathbb{P}'_i$, such that the induced partial order $(<_i)$ is an extension of $(<_{i+1})$. To achieve that, we do the following: For each color c in $\mathbb{P}'_{i+1}$, fix an arbitrary ordering among those colors of $\mathbb{P}'_i$, whose union forms c.

13

Clearly, the partial order $(<_1)$ defined in such a way respects $\mathbb{P}'_i$ for all $i \in [k]$. We further fix an arbitrary ordering among variables belonging to the same color in $\mathbb{P}'_1$. Thus, we get a total order $(<^*)$, which is an extension of $<_1$ and hence respects $\mathbb{P}'_i$ for all $i \in [k]$.

Now, we construct an ROABP for each Q_i in the variable order $<^*$. First, we multiply out the linear forms which belong to the same neighborhood in each Q_i. That is, we write Q_i as the product $\prod_{j=1}^{r_i} Q_{ij}$, where r_i is the number of neighborhoods in $\mathbb{P}_i$ (number of colors in $\mathbb{P}'_i$) and each Q_{ij} is the product of linear forms (colors) which belong to the same neighborhood in $\mathbb{P}_i$. As, the partition sequence has distance Δ, the neighborhoods have at most Δ colors. So, the degree of each Q_{ij} is bounded by Δ and hence the sparsity is bounded by $O(n^\Delta)$. By Observation 12, we can construct a width-$O(n^\Delta)$ ROABP for Q_{ij} in the variable order given by $<^*$.

Let c_{ij} denote the color of $\mathbb{P}'_i$ corresponding to Q_{ij}. As the order $<^*$ respects $\mathbb{P}'_i$, it gives an order on its colors, say $c_{ij_1} < c_{ij_2} < \cdots < c_{ij_{r_i}}$. Now, we arrange the ROABPs for Q_{ij}'s in the order $Q_{ij_1} Q_{ij_2} \ldots Q_{ij_{r_i}}$, while identifying the end vertex of Q_{ij_a} with the start vertex of $Q_{ij_{a+1}}$, for all $a \in [r_i - 1]$. Clearly the ROABP thus constructed computes the polynomial Q_i and has variable order $<^*$.
□

Once we have ROABPs for the polynomials Q_i's in the same variable order, let us make a new start node and connect it with the start node of the ROABP for Q_i with label a_i, for all $i \in [k]$. Also, let us make a new end node and connect it with the end node of the ROABP for Q_i with label 1, for all $i \in [k]$. Clearly, the ROABP thus constructed computes the polynomial $P = \sum_{i=1}^{k} a_i Q_i$ and has width $O(kn^\Delta)$. Thus, we can write

Lemma 14 (Δ-distance to ROABP). *An n-variate polynomial computed by a depth-3, Δ-distance circuit with top fan-in k has a width-$O(kn^\Delta)$ ROABP.*

Hence, from Theorem 1 we get,

Theorem 15 (Δ-distance Hitting Set). *Let $C(\mathbf{x})$ be a depth-3, Δ-distance, n-variate multilinear circuit with top fan-in k. Then there is a $(nk)^{O(\Delta \log n)}$-time hitting-set for $C(\mathbf{x})$.*

4.2. Base sets with Δ-distance. In this section we describe our second tool towards finding a hitting set for sum of constantly many set-multilinear polynomials. We further generalize the class of polynomials, for which we can give an efficient test, beyond low-distance. Basically, it is enough to have low-distance "projections".

Definition 16. *A multilinear depth-3 circuit $C(\mathbf{x})$ is said to have m-base-sets-Δ-distance if there is a partition of the variable set $\mathbf{x}$ into base sets $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$ such that for any $i \in [m]$, restriction of C on the i-th base set (i.e. other variables are considered as field constants), has Δ-distance.*

We will show that there is an efficient hitting set for this class of polynomials. In fact, we can show a general easy result for a polynomial whose restriction on one base set falls into a class C, for which a hitting set is already known.

Lemma 17 (Hybrid Argument). *Let $\mathcal{H}$ be the hitting set for a class of (n-variate) polynomials C. Let $\mathbf{x}$ be a union of m disjoint sets of variables $\mathbf{x}_1 \sqcup \mathbf{x}_2 \sqcup \cdots \sqcup \mathbf{x}_m$, called base sets, each with size at most n. Let $C(\mathbf{x})$ be a polynomial such that its restriction to the base set $\mathbf{x}_i$ (i.e. the other variables are considered as field constants), is in class C, for all $i \in [m]$. Then there is a hitting set for $C(\mathbf{x})$ of size $|\mathcal{H}|^m$ (with the knowledge of the base sets).*

Proof. Let us assume that the set $\mathbf{x}_i$ has cardinality n, for all $i \in [m]$. If not, then we can introduce dummy variables. Now, we claim that if $C(\mathbf{x}) \neq 0$ then there

exists m points $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_m \in \mathcal{H}$, such that $C(\mathbf{x}_1 = \mathbf{h}_1, \mathbf{x}_2 = \mathbf{h}_2, \mathbf{x}_m = \mathbf{h}_m) \neq 0$.

We prove the claim inductively.

Base Case: The polynomial $C(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m) \neq 0$. It follows from the assumption.

Induction Hypothesis: There exists points $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_i \in \mathcal{H}$ such that the partially evaluated polynomial $C'(\mathbf{x_{i+1}}, \ldots, \mathbf{x}_m) := C(\mathbf{x}_1 = \mathbf{h}_1, \ldots, \mathbf{x}_i = \mathbf{h}_i, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_m) \neq 0$.

Induction Step: We show that there exists $\mathbf{h_{i+1}} \in \mathcal{H}$ such that the polynomial $C'(\mathbf{x}_{i+1} = \mathbf{h}_{i+1}, \mathbf{x}_{i+2}, \ldots, \mathbf{x}_m) \neq 0$.

The polynomial C' is nothing but the polynomial C evaluated at $\mathbf{x}_1, \ldots, \mathbf{x}_i$. Hence, the polynomial C' restricted to the set $\mathbf{x}_{i+1}$, is also in the class C. So, there must exist a point $\mathbf{h}_{i+1} \in \mathcal{H}$ such that $C'(\mathbf{x}_{i+1} = \mathbf{h}_{i+1}) \neq 0$.

Thus, the claim is true. Now, to construct a hitting set for C, one needs to substitute the set $\mathcal{H}$ for each base set $\mathbf{x}_i$, i.e. the cartesian product $\mathcal{H} \times \mathcal{H} \times \cdots \times \mathcal{H}$ (m times). Hence, we get a hitting set of size $|\mathcal{H}|^m$. □

Note that, in the above proof the knowledge of the base sets is crucial. This lemma, together with Theorem 15, gives us the following:

THEOREM 18 (m-base-sets-Δ-distance PIT). *If $C(\mathbf{x})$ is a depth-3 multilinear circuit, with top fan-in k, having m base sets (known) with Δ-distance, then there is a $(nk)^{O(m\Delta \log n)}$-time hitting-set for C.*

4.3. Sum of set-multilinear circuits reduces to m-base-sets-Δ-distance. In this section, we will reduce the PIT for sum of constantly many set-multilinear depth-3 circuits, to the PIT for depth-3 circuits with m-base-sets-Δ-distance, where $m\Delta = o(n)$. Thus, we get a subexponential time whitebox algorithm for this class (from Theorem 18). Note that a sum of constantly many set-multilinear depth-3 circuits is equivalent to a depth-3 multilinear circuit such that the number of distinct partitions, induced by its product gates, is constant.

We first look at the case of two partitions. For a partition $\mathbb{P}$ of $[n]$, let $\mathbb{P}|_B$ denote the restriction of $\mathbb{P}$ on a base set $B \subseteq [n]$. E.g., if $\mathbb{P} = \{\{1, 2\}, \{3, 4\}, \{5, 6, \ldots, n\}\}$ and $B = \{1, 3, 4\}$ then $\mathbb{P}|_B = \{\{1\}, \{3, 4\}\}$. Recall that $d(\mathbb{P}_1, \mathbb{P}_2, \ldots, \mathbb{P}_c)$ denotes the *distance* of the partition sequence $(\mathbb{P}_1, \mathbb{P}_2, \ldots, \mathbb{P}_c)$ (Definition 9). For a partition sequence $(\mathbb{P}_1, \mathbb{P}_2, \ldots \mathbb{P}_c)$, and a base set $B \subseteq [n]$, let $d_B(\mathbb{P}_1, \mathbb{P}_2, \ldots, \mathbb{P}_c)$ denote the distance of the partition sequence when restricted to the base set B, i.e. $d(\mathbb{P}_1|_B, \mathbb{P}_2|_B, \ldots, \mathbb{P}_c|_B)$.

LEMMA 19. *For any two partitions $\{\mathbb{P}_1, \mathbb{P}_2\}$ of the set $[n]$, there exists a partition of $[n]$, into at most $2\sqrt{n}$ base sets $\{B_1, B_2, \ldots, B_m\}$ ($m < 2\sqrt{n}$), such that for any $i \in [m]$, either $d_{B_i}(\mathbb{P}_1, \mathbb{P}_2) = 1$ or $d_{B_i}(\mathbb{P}_2, \mathbb{P}_1) = 1$.*

Proof. Let us divide the set of colors in the partition $\mathbb{P}_1$, into two types of colors: One with at least $\sqrt{n}$ elements and the other with less than $\sqrt{n}$ elements. In other words, $\mathbb{P}_1 = \{X_1, X_2, \ldots, X_r\} \cup \{Y_1, Y_2, \ldots, Y_q\}$ such that $|X_i| \geq \sqrt{n}$ and $|Y_j| < \sqrt{n}$, for all $i \in [r]$, $j \in [q]$. Let us make each X_i a base set, i.e. $B_i = X_i$, $\forall i \in [r]$. As $|X_i| \geq \sqrt{n}$, $\forall i \in [r]$, we get $r \leq \sqrt{n}$. Now, for any $i \in [r]$, $\mathbb{P}_1|_{B_i}$ has only one color. Hence, irrespective of what colors $\mathbb{P}_2|_{B_i}$ has, $d_{B_i}(\mathbb{P}_2, \mathbb{P}_1) = 1$, for all $i \in [r]$.

Now, for the other kind of colors, we will make base sets which have exactly one element from each color Y_j. More formally, let $Y_j = \{y_{j,1}, y_{j,2}, \ldots, y_{j,r_j}\}$, for all $j \in [q]$. Let $r' = \max\{r_1, r_2, \ldots, r_q\}$ ($r' < \sqrt{n}$). Now define base sets $B'_1, B'_2, \ldots, B'_{r'}$ such that for any $a \in [r']$, $B'_a = \{y_{j,a} \mid j \in [q], |Y_j| \geq a\}$. In other words, all those Y_js which have at least a elements, contribute their a-th element to B'_a. Now for any

15

$a \in [r']$, $\mathbb{P}_1|_{B'_a} = \{\{y_{j,a}\} \mid j \in [q], |Y_j| \geq a\}$, i.e. it has exactly one element in each color. Clearly, irrespective of what colors $\mathbb{P}_2|_{B'_a}$ has, $d_{B'_a}(\mathbb{P}_1, \mathbb{P}_2) = 1$, for all $a \in [r']$.

$\{B_1, B_2, \ldots, B_r\} \cup \{B'_1, B'_2, \ldots, B'_{r'}\}$ is our final set of base sets. Clearly, they form a partition of $[n]$. The total number of base sets, $m = r + r' < 2\sqrt{n}$.

□

Now, we generalize Lemma 19 to any constant number of partitions, by induction.

LEMMA 20 (Reduction to m-base-sets-1-distance). *For any set of c partitions* $\{\mathbb{P}_1, \mathbb{P}_2, \ldots, \mathbb{P}_c\} \subseteq \mathrm{Part}([n])$, *there exists a partition of the set $[n]$, into m base sets* $\{B_1, B_2, \ldots, B_m\}$ *with $m < 2^{c-1} \cdot n^{1-(1/2^{c-1})}$ such that for any $i \in [m]$, there exists a permutation of the partitions, $(\mathbb{P}_{i_1}, \mathbb{P}_{i_2}, \ldots, \mathbb{P}_{i_c})$ with $d_{B_i}(\mathbb{P}_{i_1}, \mathbb{P}_{i_2}, \ldots, \mathbb{P}_{i_c}) = 1$.*

Proof. Let $f(c, n) := 2^{c-1} \cdot n^{1-(1/2^{c-1})}$. The proof is by induction on the number of partitions.

Base case: For $c = 2$, $f(c, n)$ becomes $2\sqrt{n}$. Hence, the statement follows from Lemma 19.

Induction hypothesis: The statement is true for any $c - 1$ partitions.

Induction step: Like in Lemma 19, we divide the set of colors in $\mathbb{P}_1$ into two types of colors. Let $\mathbb{P}_1 = \{X_1, X_2, \ldots, X_r\} \cup \{Y_1, Y_2, \ldots, Y_q\}$ such that $|X_i| \geq \sqrt{n}$ and $|Y_j| < \sqrt{n}$, for all $i \in [r]$, $j \in [q]$. Let us set $B_i = X_i$ and let $n_i := |B_i|$, $\forall i \in [r]$. Our base sets will be further subsets of these B_is. For a fixed $i \in [r]$, let us define $\mathbb{P}'_h = \mathbb{P}_h|_{B_i}$, as a partition of the set B_i, for all $h \in [c]$. Clearly, $\mathbb{P}'_1$ has only one color. Now, we focus on the partition sequence $(\mathbb{P}'_2, \mathbb{P}'_3, \ldots, \mathbb{P}'_c)$. From the inductive hypothesis, there exists a partition of B_i into m_i base sets $\{B_{i,1}, B_{i,2}, \ldots, B_{i,m_i}\}$ $(m_i \leq f(c-1, n_i))$ such that for any $u \in [m_i]$, there exists a permutation of $(\mathbb{P}'_2, \mathbb{P}'_3, \ldots, \mathbb{P}'_c)$, given by $(\mathbb{P}'_{i_2}, \mathbb{P}'_{i_3}, \ldots, \mathbb{P}'_{i_c})$, with $d_{B_{i,u}}(\mathbb{P}'_{i_2}, \mathbb{P}'_{i_3}, \ldots, \mathbb{P}'_{i_c}) = 1$. As $\mathbb{P}'_1$ has only one color, so does $\mathbb{P}'_1|_{B_{i,u}}$. Hence, $d_{B_{i,u}}(\mathbb{P}'_{i_2}, \mathbb{P}'_{i_3}, \ldots, \mathbb{P}'_{i_c}, \mathbb{P}'_1)$ is also 1. From this, we easily get $d_{B_{i,u}}(\mathbb{P}_{i_2}, \mathbb{P}_{i_3}, \ldots, \mathbb{P}_{i_c}, \mathbb{P}_1) = 1$. The above argument can be made for all $i \in [r]$.

Now for the other colors, we proceed as in Lemma 19. Let $Y_j = \{y_{j,1}, y_{j,2}, \ldots, y_{j,r_j}\}$, for all $j \in [q]$. Let $r' = \max\{r_1, r_2, \ldots, r_q\}$ $(r' < \sqrt{n})$. Now define sets $B'_1, B'_2, \ldots, B'_{r'}$ such that for any $a \in [r']$, $B'_a = \{y_{j,a} \mid j \in [q], |Y_j| \geq a\}$. In other words, all those Y_js which have at least a elements, contribute their a-th element to B'_a. Let $n'_a := |B'_a|$, for all $a \in [r']$. Our base sets will be further subsets of these B'_as. For a fixed $a \in [r']$, let us define $\mathbb{P}'_h = \mathbb{P}_h|_{B'_a}$, as a partition of the set B'_a, for all $h \in [c]$. Clearly, $\mathbb{P}'_1$ has exactly one element in each of its colors. Now, we focus on the partition sequence $(\mathbb{P}'_2, \mathbb{P}'_3, \ldots, \mathbb{P}'_c)$. From the inductive hypothesis, there exists a partition of B'_a into m'_a base sets $\{B'_{a,1}, B'_{a,2}, \ldots, B'_{a,m'_a}\}$ $(m'_a \leq f(c-1, n'_a))$ such that for any $u \in [m'_a]$, there exists a permutation of $(\mathbb{P}'_2, \mathbb{P}'_3, \ldots, \mathbb{P}'_c)$, given by $(\mathbb{P}'_{i_2}, \mathbb{P}'_{i_3}, \ldots, \mathbb{P}'_{i_c})$, with $d_{B'_{a,u}}(\mathbb{P}'_{i_2}, \mathbb{P}'_{i_3}, \ldots, \mathbb{P}'_{i_c}) = 1$. As $\mathbb{P}'_1$ has exactly one element in each of its colors, so does $\mathbb{P}'_1|_{B'_{a,u}}$. Hence, $d_{B'_{a,u}}(\mathbb{P}'_1, \mathbb{P}'_{i_2}, \mathbb{P}'_{i_3}, \ldots, \mathbb{P}'_{i_c})$ is also 1. From this, we easily get $d_{B'_{a,u}}(\mathbb{P}_1, \mathbb{P}_{i_2}, \mathbb{P}_{i_3}, \ldots, \mathbb{P}_{i_c}) = 1$. The above argument can be made for all $a \in [r']$.

Our final set of base sets will be $\{B_{i,u} \mid i \in [r], u \in [m_i]\} \cup \{B'_{a,u} \mid a \in [r'], u \in [m'_a]\}$. As argued above, when restricted to any of these base sets, the given partitions have a sequence, which has distance 1. Now, we need to bound the number of these base sets,

$$m = \sum_{i \in [r]} m_i + \sum_{a \in [r']} m'_a.$$

From the bounds on m_i and m'_a, we get

$$m \leq \sum_{i \in [r]} f(c-1, n_i) + \sum_{a \in [r']} f(c-1, n'_a).$$

Recall that $n_i \geq \sqrt{n}$. We break the second sum, in the above equation, into two parts. Let $R_1 = \{a \in [r'] \mid n_a' \geq \sqrt{n}\}$ and $R_2 = \{a \in [r'] \mid n_a' < \sqrt{n}\}$.

$$m \leq \sum_{i \in [r]} f(c-1, n_i) + \sum_{a \in R_1} f(c-1, n_a') + \sum_{a \in R_2} f(c-1, n_a'). \tag{3}$$

Let us first focus on the third sum. Note that $|R_2| \leq r' < \sqrt{n}$. For $a \in R_2$, $n_a' < \sqrt{n}$ and hence $f(c-1, n_a') < f(c-1, \sqrt{n}) = 2^{c-2} \cdot n^{1/2-(1/2^{c-1})}$. So,

$$\sum_{a \in R_2} f(c-1, n_a') < \sqrt{n} \cdot 2^{c-2} \cdot n^{1/2-(1/2^{c-1})} = 2^{c-2} \cdot n^{1-(1/2^{c-1})}. \tag{4}$$

Now, we focus on first two sums in Equation (3). As, $n_i \geq \sqrt{n}$, $\forall i \in [r]$ and $n_a' \geq \sqrt{n}$, $\forall a \in R_1$, we combine these two sums (with an abuse of notation) and write the sum as follows,

$$\sum_{i \in [r'']} f(c-1, n_i),$$

where $r'' = r + |R_1|$, and $n_i \geq \sqrt{n}$, $\forall i \in [r'']$. As each $n_i \geq \sqrt{n}$, we know $r'' < \sqrt{n}$ (as $\sum n_i \leq n$).

Observe that $f(c-1, z)$, as a function of z, is a concave function (its derivative is monotonically decreasing, when $z > 0$). From the properties of a concave function, we know,

$$\frac{1}{r''} \sum_{i \in [r'']} f(c-1, n_i) \leq f\left(c-1, \frac{1}{r''} \sum_{i \in [r'']} n_i\right).$$

Now, $\sum_{i \in [r'']} n_i \leq n$ and $f(c-1, z)$ is an increasing function (when $z > 0$). Hence,

$$\frac{1}{r''} \sum_{i \in [r'']} f(c-1, n_i) \leq f\left(c-1, \frac{1}{r''} n\right).$$

Equivalently,

$$\sum_{i \in [r'']} f(c-1, n_i) \leq r'' \cdot 2^{c-2} \cdot (n/r'')^{1-(1/2^{c-2})}$$

$$= 2^{c-2} \cdot n^{1-(1/2^{c-2})} \cdot (r'')^{1/2^{c-2}}$$

$$< 2^{c-2} \cdot n^{1-(1/2^{c-2})} \cdot n^{1/2^{c-1}}$$

$$= 2^{c-2} \cdot n^{1-(1/2^{c-1})}.$$

Using this with Equation (4) and substituting in Equation (3), we get

$$m < 2^{c-1} \cdot n^{1-(1/2^{c-1})}.$$

□

Now, we combine these results with our hitting-sets for depth-3 circuits having m base sets with Δ-distance.

THEOREM 2 (restated). *Let $C(\mathbf{x})$ be a n-variate polynomial, which can be computed by a sum of c set-multilinear depth-3 circuits, each having top fan-in k. Then there is a $(nck)^{O(2^{c-1}n^{1-\epsilon}\log n)}$-time whitebox PIT test for C, where $\epsilon := 1/2^{c-1}$.*

Proof. As mentioned earlier, the polynomial $C(\mathbf{x})$ can be viewed as being computed by a depth-3 multilinear circuit, such that its product gates induce at most c-many distinct partitions. From Lemma 20, we can partition the variable set into m base sets, such that for each of these base sets, the partitions can be sequenced to have distance 1, where $m := 2^{c-1}n^{1-\epsilon}$. Hence, the polynomial C has m base sets with 1-distance and top fan-in ck. Moreover, from the proof of Lemma 20, it is clear that such base sets can be computed in $n^{O(c)}$-time. From Theorem 18, we know that there is $(nck)^{O(m\log n)}$-time whitebox PIT test for such a circuit. Substituting the value of m, we get the result. $\square$

Tightness of this method. Lemma 19 can be put in other words as: Any two partitions have m-base-sets-Δ-distance with $m\Delta = O(\sqrt{n})$. We can, in fact, show that this result is tight.

Showing the lower bound: Let $d(\mathbb{P}_1, \mathbb{P}_2) = \Delta$. Then each color of $\mathbb{P}_2$ has a friendly neighborhood (of at most Δ colors) which is exactly partitioned in $\mathbb{P}_1$. Now construct Δ base sets such that i-th base set takes the variables of i-th color from every neighborhood of $\mathbb{P}_2$. Clearly, when restricted to one of these bases sets, $d(\mathbb{P}_1, \mathbb{P}_2)$ is 1. In other words $\mathbb{P}_1$ and $\mathbb{P}_2$ have Δ-base-sets-1-distance. Similarly, one can argue that if $\mathbb{P}_1$ and $\mathbb{P}_2$ have m-base-sets-Δ-distance then they also have $m\Delta$-base-sets-1-distance. Now, we will show that if we want m-base-sets-1-distance for two partitions then $m = \Omega(\sqrt{n})$.

Consider the following example (assuming n is a square):

$\mathbb{P}_1 = \{\{1, 2, \ldots \sqrt{n}\}, \{\sqrt{n}+1, \sqrt{n}+2, \ldots, 2\sqrt{n}\}, \ldots, \{\sqrt{n}(\sqrt{n}-1)+1, \sqrt{n}(\sqrt{n}-1)+2, \ldots, n\}\}$ and

$\mathbb{P}_2 = \{\{1, \sqrt{n}+1, \ldots, n-\sqrt{n}+1\}, \{2, \sqrt{n}+2, \ldots, n-\sqrt{n}+2\}, \ldots, \{\sqrt{n}, 2\sqrt{n}, \ldots, n\}\}$. Basically, $\mathbb{P}_2$ has the residue classes (mod $\sqrt{n}$).

OBSERVATION 21. *A base set B, such that $d_B(\mathbb{P}_1, \mathbb{P}_2) = 1$, has at most $\sqrt{n}$ variables.*

Proof. Suppose it has more than $\sqrt{n}$ variables. Then, there is at least one color in $\mathbb{P}_1$ which contributes two variables to B. These two variables have to be in two different colors of $\mathbb{P}_2$ (because of our design of $\mathbb{P}_1$ and $\mathbb{P}_2$). So, $d_B(\mathbb{P}_1, \mathbb{P}_2)$ is at least 2. We get a contradiction. $\square$

The number of such base sets has to be at least $\sqrt{n}$. Combining this with the reduction from m-base-sets-Δ-distance to $m\Delta$-base-sets-1-distance, we get $m\Delta = \Omega(\sqrt{n})$.

It is not clear if Lemma 20 is tight. We conjecture that for any set of partitions, $m\Delta = O(\sqrt{n})$ can be achieved.

5. Sparse-Invertible Width-w ROABP: Theorem 3. As mentioned in Section 2, a polynomial $C(\mathbf{x})$ computed by s-sparse-factor width-w ROABP can be written as $D_0^\top (\prod_{i=1}^{d} D_i) D_{d+1}$, where $D_i \in \mathbb{F}^{w\times w}[\mathbf{x}_i]$ is an s-sparse polynomial for all $i \in [d]$, and $\mathbf{x}_1, \mathbf{x}_2 \ldots, \mathbf{x}_d$ are disjoint sets of variables.

We will show a hitting-set for a sparse-factor ROABP $D_0(\prod_{i=1}^{d} D_i) D_{d+1}$ with D_i being an *invertible* matrix, for all $i \in [d]$. Hence, we name this model *sparse-invertible-factor ROABP*. To be more general, we take D_0 and D_{d+1} also to be polynomials in some sets of variables disjoint from $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_d$.

For a polynomial D, let its sparsity $\mathsf{s}(D)$ be the number of monomials in D with nonzero coefficients and let $\mu(D)$ be the maximum support of any monomial in D.

THEOREM 3 (restated). *Let* $\mathbf{x} = \mathbf{x}_0 \sqcup \cdots \sqcup \mathbf{x}_{d+1}$, *with* $|\mathbf{x}| = n$. *Let* $C(\mathbf{x}) = D_0^\top D D_{d+1} \in \mathbb{F}[\mathbf{x}]$ *be a polynomial with* $D(\mathbf{x}) = \prod_{i=1}^d D_i(\mathbf{x}_i)$, *where* $D_0 \in \mathbb{F}^w[\mathbf{x}_0]$ *and* $D_{d+1} \in \mathbb{F}^w[\mathbf{x}_{d+1}]$ *and for all* $i \in [d]$, $D_i \in \mathbb{F}^{w \times w}[\mathbf{x}_i]$ *is an invertible matrix. For all* $i \in \{0, 1, \ldots, d+1\}$, D_i *has degree bounded by* δ, $\mathsf{s}(D_i) \leq s$ *and* $\mu(D_i) \leq \mu$. *Let* $\ell := 1 + 2 \min\{\lceil \log(w^2 \cdot s) \rceil, \mu\}$. *Then there is a hitting-set of size* $\text{poly}((n\delta s)^{\ell w^2})$ *for* $C(\mathbf{x})$.

REMARK. *If* $\mu = 1$, *e.g. each* D_i *is either a univariate or a linear polynomial, then we get poly-time for constant* w. *Also if both* w *and the sparsity-bound* s *are constant, we get poly-time.*

Like [ASS13] and [FSS14], we find a hitting-set by showing a *low-support concentration*. Low support concentration in the polynomial $D(\mathbf{x}) = \prod_{i=1}^d D_i$ means that the coefficients of the low support monomials in $D(\mathbf{x})$ span the whole coefficient space of $D(\mathbf{x})$.

Let $\mathbf{x}$ be $\{x_1, x_2 \ldots, x_n\}$. For any $e \in \mathbb{Z}_+^n$, support of the monomial $\mathbf{x}^e$ is defined as $\mathsf{S}(e) := \{i \in [n] \mid e_i \neq 0\}$ and support size is defined as $\mathsf{s}(e) := |\mathsf{S}(e)|$. Now, we define ℓ-*concentration* for a polynomial $D(\mathbf{x}) \in \mathbb{F}^{w \times w}[\mathbf{x}]$.

DEFINITION 22 (ℓ-concentration). *Polynomial* $D(\mathbf{x}) \in \mathbb{F}^{w \times w}[\mathbf{x}]$ *is* ℓ-*concentrated if* $\text{rank}_{\mathbb{F}}\{\text{coef}_D(\mathbf{x}^e) \mid e \in \mathbb{Z}_+^n, \ \mathsf{s}(e) < \ell\} = \text{rank}_{\mathbb{F}}\{\text{coef}_D(\mathbf{x}^e) \mid e \in \mathbb{Z}_+^n\}$.

We will later see that the low support concentration in polynomial $D(\mathbf{x})$ implies low support concentration in polynomial $C(\mathbf{x})$ (defined similarly). In other words, $C(\mathbf{x})$ will have a nonzero coefficient for at least one of the low support monomials. Thus, we get a hitting set by testing these low support coefficients. We use the following lemma from [ASS13].

LEMMA 23. *If* $C(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ *is an* n-*variate,* ℓ-*concentrated polynomial with highest individual degree* δ, *then there is a* $(n\delta)^{O(\ell)}$-*time hitting-set for* $C(\mathbf{x})$.

Proof. ℓ-concentration for $C(\mathbf{x})$ simply means that it has at least one $(< \ell)$-support monomial with nonzero coefficient. We will construct a hitting set which essentially will test all these $(< \ell)$-support coefficients. We go over all subsets S of $\mathbf{x}$ with size $\ell - 1$ and do the following: Substitute 0 for all the variables outside the set S. There will be at least one choice of S, for which the polynomial $C(\mathbf{x})$ remains nonzero after the substitution. Now, it is an $(\ell - 1)$-variate nonzero polynomial. We take the usual hitting set $\mathcal{H}^{\ell-1}$ for this, where $\mathcal{H} \subseteq \mathbb{F}$ is a set of size $\delta + 1$ (see, for example, [SY10, Fact 4.1]). In other words, each of these $\ell - 1$ variables are assigned values from the set $\mathcal{H}$.

The number of sets S we need to try are $\binom{n}{\ell-1}$. Hence, the overall hitting set size is $(n\delta)^{O(\ell)}$. ∎

Now, we move on to show how to achieve low support concentration in $D(\mathbf{x}) = \prod_{i=1}^d D_i$. To achieve that we will use some efficient shift. By shifting by a point $\alpha := (\alpha_1, \alpha_2, \ldots, \alpha_n)$, we mean replacement of x_i with $x_i + \alpha_i$. Note that $D(\mathbf{x} + \alpha) \neq 0$ if and only if $D(\mathbf{x}) \neq 0$. Hence, a hitting set for $D(\mathbf{x} + \alpha)$ gives us a hitting set for $D(\mathbf{x})$. Instead of constants, we will be actually shifting $D(\mathbf{x})$ by univariate polynomials, say, given by the map $\phi \colon \mathbf{t} \to \{t^a\}_{a \geq 0}$, where $\mathbf{t} := \{t_1, t_2, \ldots, t_n\}$. The ϕ is said to be an efficient map if $\phi(t_i)$ is efficiently computable, for each $i \in [n]$.

Proof Idea- As all the matrices in the matrix product $D(\mathbf{x}) = \prod_{i=1}^d D_i(\mathbf{x}_i)$ are over disjoint sets of variables, any coefficient in the polynomial $D(\mathbf{x})$ can be uniquely written as a product of d factors, each coming from one D_i. We start with the assumption that the constant term of each polynomial D_i, denoted by $D_{i\mathbf{0}}$, is an

invertible matrix. Using this we define a notion of *parent* and *child* between all the coefficients (also see Figure 1): If a coefficient can be obtained from another coefficient by replacing one of its constant factors $D_{i\mathbf{0}}$ with another term (with non-trivial support) from D_i, then former is called a parent of the latter. Observe that if we want to do this replacement by a multiplication of some matrix, then $D_{i\mathbf{0}}$ should be invertible. Moreover, all the factors on its right side (or its left side) also need to be constant terms in their respective matrices (this is because of non-commutativity). For a coefficient, the set of matrices D_i which contribute a non-trivial factor to it, is said to form the *block-support* of the coefficient.

Our next step is to show that if a coefficient linearly depends on its descendants then the dependence can be lifted to its parent (by dividing and multiplying appropriate factors), i.e. its parent also linearly depends on its descendants. As the dimension of the matrix algebra is constant, if we take an appropriately large (constant) child-parent chain, there will be a linear dependence among the coefficients in the chain. As the dependencies lift to the parent, they can be lifted all the way up. By an inductive argument it follows that every coefficient depends on the coefficients with low-block-support. Now, this can be translated to low-support concentration in D, if a low-support concentration is assumed in each D_i.

To achieve low-support concentration in each D_i, we use an appropriate shift. The sparsity of D_i is used crucially in this step. To make $D_{i\mathbf{0}}$ invertible, again an appropriate shift is used. Note that $D_{i\mathbf{0}}$ can be made invertible by a shift only when D_i itself is invertible, hence the invertible-factor assumption.

5.1. Building the Proof of Theorem 3. Our first focus will be on the matrix product $D(\mathbf{x}) := \prod_{i=1}^d D_i$ which belongs to $\mathbb{F}^{w \times w}[\mathbf{x}]$. We will show low-support concentration in $D(\mathbf{x})$ over the matrix algebra $\mathbb{F}^{w \times w}$ (which is non-commutative!).

5.1.1. Low Block-Support. Let the matrix product $D(\mathbf{x}) := \prod_{i=1}^d D_i$ correspond to an ROABP such that $D_i \in \mathbb{F}^{w \times w}[\mathbf{x}_i]$ for all $i \in [d]$. Let n_i be the cardinality of $\mathbf{x}_i$ and let $n = \sum_{i=1}^d n_i$. For an exponent $e = (e_1, e_2, \ldots, e_m) \in \mathbb{Z}_+^m$, and for a set of variables $\mathbf{y} = \{y_1, y_2, \ldots, y_m\}$, $\mathbf{y}^e$ will denote $y_1^{e_1} y_2^{e_2} \ldots y_m^{e_m}$.

Viewing D_i as belonging to $\mathbb{F}^{w \times w}[\mathbf{x}_i]$, one can write $D_i := \sum_{e \in \mathbb{Z}_+^{n_i}} D_{ie} \mathbf{x}_i^e$, where $D_{ie} \in \mathbb{F}^{w \times w}$, for all $e \in \mathbb{Z}_+^{n_i}$. In particular $D_{i\mathbf{0}}$ refers to the constant part of the polynomial D_i.

For any $e \in \mathbb{Z}_+^n$, support of the monomial $\mathbf{x}^e$ is defined as $\mathrm{S}(e) := \{i \in [n] \mid e_i \neq 0\}$ and support size is defined as $\mathrm{s}(e) := |\mathrm{S}(e)|$. In this section, we will also define block-support of a monomial. Any monomial $\mathbf{x}^e$ for $e \in \mathbb{Z}_+^n$, can be seen as a product $\prod_{i=1}^d \mathbf{x}_i^{e_i}$, where $e_i \in \mathbb{Z}_+^{n_i}$ for all $i \in [d]$, such that $e = (e_1, e_2, \ldots, e_d)$. We define *block-support* of e, $\mathrm{bS}(e)$ as $\{i \in [d] \mid e_i \neq \mathbf{0}\}$ and *block-support size* of e, $\mathrm{bs}(e) = |\mathrm{bS}(e)|$.

Next, we will show low block-support concentration of $D(\mathbf{x})$ when *each $D_{i\mathbf{0}}$ is invertible*.

As each D_i is a polynomial over a different set of variables, we can easily see that the coefficient of any monomial $\mathbf{x}^e = \prod_{i=1}^d \mathbf{x}_i^{e_i}$ in $D(\mathbf{x})$ is

$$D_e := \prod_{i=1}^d D_{ie_i}. \tag{5}$$

Now, we will define a relation of *parent* and *children* between these coefficients.

DEFINITION 24. *For $e^*, e \in \mathbb{Z}_+^n$, D_{e^*} is called a parent of D_e if $\exists j \in [d]$, $j > \max \mathrm{bS}(e)$ or $j < \min \mathrm{bS}(e)$, such that $\mathrm{bS}(e^*) = \mathrm{bS}(e) \cup \{j\}$ and $e_i^* = e_i$, $\forall i \in [d]$ with*
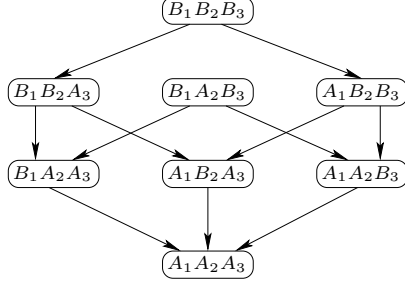
FIG. 1. *An edge represents the child-parent relationship among the coefficients. The arrow points towards the child.*

$i \neq j$.

If D_{e^*} is a parent of D_e then D_e is a *child* of D_{e^*}. Note that *a coefficient has at most two children*, on the other hand it can have many parents. In the case when $j > \max \mathrm{bS}(e)$ we call e, the *left* child of e^* and in the other case we call it the *right* child. Figure 1 shows this relationship between the coefficients for the polynomial $(A_1 + B_1 x_1)(A_2 + B_2 x_2)(A_3 + B_3 x_3)$, where $A_i, B_i \in \mathbb{F}^{w \times w}$, for all $i \in \{1, 2, 3\}$.

To motivate this definition, observe that if $j > \max \mathrm{bS}(e)$ then by Equation (5) we can write $D_{e^*} = D_e A^{-1} B$, where $A := \prod_{i=j}^{d} D_{i\mathbf{0}}$ and $B := D_{je_j^*} \prod_{i=j+1}^{d} D_{i\mathbf{0}}$. We will denote the product $A^{-1}B$ as $D_{e^{-1}e^*}$. Similarly, if $j < \min \mathrm{bS}(e)$ then one can write $D_{e^*} = BA^{-1}D_e$, where $A := \prod_{i=1}^{j} D_{i\mathbf{0}}$ and $B := \left(\prod_{i=1}^{j-1} D_{i\mathbf{0}} \right) D_{je_j^*}$. In this case we will denote the product BA^{-1} as $D_{e^* e^{-1}}$. Note that the invertibility of $D_{i\mathbf{0}}$s is crucial here.

We also define *descendants* of a coefficient D_e as $\mathrm{descend}(D_e) := \{D_f \mid f \in \mathbb{Z}_+^n, \ \mathrm{bS}(f) \subset \mathrm{bS}(e)\}$. Note that, the set of descendants of a coefficient could be bigger than the set of its children, grand-children, etc. Now, we will view the coefficients as $\mathbb{F}$-vectors and look at the linear dependence between them. The following lemma shows how these dependencies lift to the parent.

LEMMA 25 (Child to parent). *Let D_{e^*} be a parent of D_e. If D_e is linearly dependent on its descendants, then D_{e^*} is linearly dependent on its descendants.*

Proof. Let D_e be the left child of D_{e^*} (the other case is similar). So, we can write

$$D_{e^*} = D_e D_{e^{-1}e^*}. \tag{6}$$

Let the dependence of D_e on its descendants be the following:

$$D_e = \sum_{\substack{f \\ \mathrm{bS}(f) \subset \mathrm{bS}(e)}} \alpha_f D_f.$$

Using Equation (6) we can write,

$$D_{e^*} = \sum_{\substack{f \\ \mathrm{bS}(f) \subset \mathrm{bS}(e)}} \alpha_f D_f D_{e^{-1}e^*}.$$

Now, we just need to show that for any D_f with $\mathrm{bS}(f) \subset \mathrm{bS}(e)$, $D_f D_{e^{-1}e^*}$ is a valid coefficient of some monomial in $D(\mathbf{x})$ and also that it is a descendant of D_{e^*}. Recall that $D_{e^{-1}e^*} = A^{-1}B$, where $A := \prod_{i=j}^{d} D_{i\mathbf{0}}$ and $B := D_{je_j^*} \prod_{i=j+1}^{d} D_{i\mathbf{0}}$ and

21

$\mathrm{bS}(e^*) = \mathrm{bS}(e) \cup \{j\}$. We know that $j > \max\{\mathrm{bS}(e)\}$. Hence, $j > \max\{\mathrm{bS}(f)\}$ as $\mathrm{bS}(f) \subset \mathrm{bS}(e)$. So, it is clear that $D_f D_{e^{-1}e^*}$ is the coefficient of $\mathbf{x}^{f^*} := \mathbf{x}^f \mathbf{x}_j^{e_j^*}$. It is easy to see that $\mathrm{bS}(f^*) = \mathrm{bS}(f) \cup \{j\} \subset \mathrm{bS}(e^*)$. Hence, $D_{f^*} = D_f D_{e^{-1}e^*}$ is a descendant of D_{e^*}.
□

Clearly, if the descendants are more than $\dim_{\mathbb{F}} \mathbb{F}^{w \times w}$, then there will be a linear dependence among them. So,

LEMMA 26. *Any coefficient D_e, with $\mathrm{bs}(e) = w^2$, $\mathbb{F}$-linearly depends on its descendants.*

Proof. First of all, we show that if a coefficient D_{f^*} is nonzero then so are its children. Let us consider its left child D_f (the other case is similar). Recall that we can write $D_{f^*} = D_f D_{f^{-1}f^*}$. Hence if D_f is zero, so is D_{f^*}.

Let $k := w^2$. Now, consider a chain of coefficients $D_{e_0}, D_{e_1}, \ldots, D_{e_k} = D_e$, such that for any $i \in [k]$, $D_{e_{i-1}}$ is a child of D_{e_i}. Clearly, $\mathrm{bs}(e_i) = i$ for $0 \le i \le k$. All the vectors in this chain are nonzero because of our above argument, as D_e is nonzero (The case of $D_e = 0$ is trivial). These $k + 1$ vectors lie in $\mathbb{F}^k$, hence, there exists an $i \in [k]$ such that D_{e_i} is linear dependent on $\{D_{e_0}, \ldots, D_{e_{i-1}}\}$. As descendants include children, grand-children, etc., we can say that D_{e_i} is linearly dependent on its descendants. Now, by applying Lemma 25 repeatedly, we conclude $D_{e_k} = D_e$ is dependent on its descendants. □

Note that, for a coefficient D_e with $\mathrm{bs}(e) = i$, its descendants have block-support strictly smaller than i. So, Lemma 26 means that coefficients with block-support w^2 depend on coefficients with block-support $\le w^2 - 1$. Now, we show w^2-block-support-concentration in $D(\mathbf{x})$, i.e. any coefficient is dependent on the coefficients with block-support $\le w^2 - 1$.

LEMMA 27 (w^2-Block-concentration). *Let $D(\mathbf{x}) = \prod_{i=1}^d D_i(\mathbf{x}_i) \in \mathbb{F}^{w \times w}[\mathbf{x}]$ be a polynomial with $D_{i\mathbf{0}}$ being invertible for each $i \in [d]$. Then $D(\mathbf{x})$ has w^2-block-support concentration.*

Proof. Let $k := w^2$. We will actually show that for any coefficient D_e with $\mathrm{bs}(e) \ge k$ (the case when $\mathrm{bs}(e) < k$ is trivial),

$$D_e \in \mathrm{span}\{D_f \mid f \in \mathbb{Z}_+^n, \ \mathrm{bS}(f) \subset \mathrm{bS}(e) \text{ and } \mathrm{bs}(f) \le k - 1\}.$$

We will prove the statement by induction on the block-support of D_e, $\mathrm{bs}(e)$.

Base case: When $\mathrm{bs}(e) = k$, it has been already shown in Lemma 26.

Induction Hypothesis: For any coefficient D_e with $\mathrm{bs}(e) = i - 1$ for $i - 1 \ge k$,

$$D_e \in \mathrm{span}\{D_f \mid f \in \mathbb{Z}_+^n, \ \mathrm{bS}(f) \subset \mathrm{bS}(e) \text{ and } \mathrm{bs}(f) \le k - 1\}.$$

Induction step: Let us take a coefficient D_e with $\mathrm{bs}(e) = i$. Consider any child of D_e, denoted by $D_{e'}$. As $\mathrm{bs}(e') = i - 1$, by our induction hypothesis, $D_{e'}$ is linearly dependent on its descendants. So, from Lemma 25, D_e is linearly dependent on its descendants. In other words,

$$D_e \in \mathrm{span}\{D_f \mid \mathrm{bS}(f) \subset \mathrm{bS}(e) \text{ and } \mathrm{bs}(f) \le i - 1\}. \tag{7}$$

Again, by our induction hypothesis, for any coefficient D_f, with $\mathrm{bs}(f) \le i - 1$,

$$D_f \in \mathrm{span}\{D_g \mid \mathrm{bS}(g) \subset \mathrm{bS}(f) \text{ and } \mathrm{bs}(g) \le k - 1\}. \tag{8}$$

Combining Equations (7) and (8), we get

$$D_e \in \mathrm{span}\{D_g \mid \mathrm{bS}(g) \subset \mathrm{bS}(e) \text{ and } \mathrm{bs}(g) \le k - 1\}.$$

□

Now, we show low block-support concentration in the actual polynomial computed by an ROABP, i.e. in $C(\mathbf{x}) = D_0^\top(\prod_{i=1}^d D_i)D_{d+1}$, where $D_0, D_{d+1} \in F^w[\mathbf{x}]$. Note that in context of C, the definition of block support is appropriately modified. Block support of a monomial now is a subset of $\{0, 1, \ldots, d+1\}$. As before it will contain the index i, if the monomial has a non-trivial support from $\mathbf{x}_i$, for $0 \leq i \leq d+1$.

LEMMA 28. *Let* $\mathbf{x} = \mathbf{x}_0 \sqcup \mathbf{x}_1 \sqcup \cdots \sqcup \mathbf{x}_{d+1}$. *Let* $D(\overline{x}) \in \mathbb{F}^{w \times w}[\mathbf{x}_1, \ldots, \mathbf{x}_d]$ *be a polynomial described in Lemma 27. Let* $C(\mathbf{x}) = D_0^\top D D_{d+1} \in \mathbb{F}[\mathbf{x}]$ *be a polynomial with* $D_0 \in \mathbb{F}^w[\mathbf{x}_0]$, $D_{d+1} \in \mathbb{F}^w[\mathbf{x}_{d+1}]$. *Then* $C(\mathbf{x})$ *has* (w^2+2)*-block-support concentration.*

Proof. Let $k := w^2$. Lemma 27 shows that $D(\mathbf{x})$ has k-block-support concentration. The coefficient of $\mathbf{x}^e$ in C is $C_e := D_{0e_0} \prod_{i=1}^d D_{ie_i} D_{(d+1)e_{d+1}}$, where $e = (e_0, e_1, \ldots, e_d, e_{d+1})$. Let $D_e := \prod_{i=1}^d D_{ie_i}$. By k-block-support concentration of $D(\mathbf{x})$,

$$D_e \in \text{span}\{D_f \mid \text{bs}(f) \leq k - 1\}.$$

Which implies,

$$C_e \in \text{span}\{D_{0e_0} D_f D_{(d+1)e_{d+1}} \mid \text{bs}(f) \leq k - 1\}.$$

Clearly, $D_{0e_0} D_f D_{(d+1)e_{d+1}}$ is the coefficient of the monomial $x_0^{e_0} x_1^{f_1} \cdots x_d^{f_d} x_{d+1}^{e_{d+1}}$. Hence, $C_e \in \text{span}\{C_f \mid \text{bs}(f) \leq k + 1\}$. □

5.2. Low-support concentration. Now, we want to show that if $C(\mathbf{x}) = D_0^\top(\prod_{i=1}^d D_i)D_{d+1}$ has low block-support concentration and moreover if each D_i has low-support concentration then $C(\mathbf{x})$ has an appropriate low-support concentration.

LEMMA 29 (Composition). *Let* $C(\mathbf{x})$ *be a polynomial* $D_0^\top D D_{d+1}$ *as described in Lemma 28. If* $C(\mathbf{x})$ *has* ℓ*-block-support concentration and* $D_i(\mathbf{x}_i)$ *has* ℓ'*-support concentration for all* $i \in [d]$ *then* $C(\mathbf{x})$ *has* $\ell\ell'$*-support concentration.*

Proof. Recall that as D_i's are polynomials over disjoint sets of variables, any coefficient C_f in $C(\mathbf{x})$ can be written as $D_{0f_0}^\top(\prod_{i=1}^d D_{if_i})D_{(d+1)f_{d+1}}$, where $f = (f_0, f_1, f_2, \ldots, f_{d+1})$ and D_{if_i} is the coefficient corresponding to the monomial $\mathbf{x}_i^{f_i}$ in D_i for all $0 \leq i \leq d+1$. From the definition of $\text{bS}(f)$, we know that $f_i = 0$, for any $i \notin \text{bS}(f)$. From ℓ'-support concentration of $D_i(\mathbf{x}_i)$, we know that for any coefficient D_{if_i},

$$D_{if_i} \in \text{span}\{D_{ig_i} \mid g_i \in \mathbb{Z}_+^{n_i}, \text{s}(g_i) \leq \ell' - 1\}.$$

Using this, we can write

$$C_f \in \text{span}\left\{ D_{0g_0}^\top \prod_{i=1}^d D_{ig_i} D_{(d+1)g_{d+1}} \mid g_i \in \mathbb{Z}_+^{n_i}, \text{s}(g_i) \leq \ell' - 1, \forall i \in [[d+1]] \right. \tag{9}$$

$$\left. \text{and } g_i = \mathbf{0}, \forall i \notin \text{bS}(f) \right\}.$$

Note that the product $D_{0g_0}^\top \prod_{i=1}^d D_{ig_i} D_{(d+1)g_{d+1}}$ will be the coefficient of a monomial $\mathbf{x}^g$ such that $\text{bS}(g) \subseteq \text{bS}(f)$ because $g_i = \mathbf{0}, \forall i \notin \text{bS}(f)$. Clearly, if $\text{s}(g_i) \leq \ell' - 1, \forall i \in \text{bS}(f)$ then $\text{s}(g) \leq (\ell' - 1) \text{bs}(f)$. So, one can write

$$C_f \in \text{span}\{C_g \mid g \in \mathbb{Z}_+^n, \text{s}(g) \leq (\ell' - 1) \text{bs}(f)\}. \tag{10}$$

23

From ℓ-block-support concentration of $C(\mathbf{x})$, we know that for any coefficient C_e of $C(\mathbf{x})$,

$$C_e \in \mathrm{span}\{C_f \mid f \in \mathbb{Z}_+^n, \; \mathrm{bs}(f) \leq \ell - 1\}. \tag{11}$$

Using Equations (10) and (11), we can write for any coefficient C_e of $C(\mathbf{x})$,

$$C_e \in \mathrm{span}\{C_g \mid g \in \mathbb{Z}_+^n, \; \mathrm{s}(g) \leq (\ell' - 1)(\ell - 1)\}.$$

Hence, $C(\mathbf{x})$ has $((\ell - 1)(\ell' - 1) + 1)$-support concentration and hence $\ell\ell'$-support concentration. $\square$

Now, we just need to show low-support concentration of each D_i. To achieve that we will use some efficient shift. Shifting will serve a dual purpose. Recall that for Lemma 27, we need invertibility of the constant term in D_i, i.e. $D_{i\mathbf{0}}$, for all $i \in [d]$. In case $D_{i\mathbf{0}}$ is not invertible for some $i \in [d]$, after a shift it might become invertible, since D_i is assumed invertible in the sparse-invertible model. For the shifted polynomial $D_i'(\mathbf{x}_i) := D_i(\mathbf{x}_i + \phi(\mathbf{t}_i))$, its constant term $D_{i\mathbf{0}}'$ is just an evaluation of $D_i(\mathbf{x})$, i.e. $D_i|_{\mathbf{x}_i = \phi(\mathbf{t}_i)}$. Now, we want a shift for D_i which would ensure that $\det(D_{i\mathbf{0}}') \neq 0$ and that D_i' has low-support concentration. For both the goals we use the sparsity of the polynomial.

For a polynomial D, let its sparsity set $\mathsf{S}(D)$ be the set of monomials in D with nonzero coefficients and $\mathsf{s}(D)$ be its sparsity, i.e. $\mathsf{s}(D) = |\mathsf{S}(D)|$. Let, for a polynomial $D(\mathbf{x}) \in \mathbb{F}^{w \times w}[\mathbf{x}]$, $S = \mathsf{S}(D)$ and $s = |S|$. Then it is easy to see that for its determinant polynomial $\mathsf{S}(\det(D)) \subseteq S^w$, where $S^w := \{m_1 m_2 \cdots m_w \mid m_i \in S, \; \forall i \in [w]\}$. Hence $\mathsf{s}(\det(D)) \leq s^w$. Now, suppose $\det(D) \neq 0$. We will describe an efficient shift which will make the constant term, of the shifted polynomial, invertible. Let $\phi \colon \mathbf{t} \to \{t^i\}_{i=0}^{\infty}$ be a monomial map which separates all the monomials in $\det(D(\mathbf{t}))$, i.e. for any two $\mathbf{t}^{e_1}, \mathbf{t}^{e_2} \in \mathsf{S}(\det(D(\mathbf{t})))$, $\phi(\mathbf{t}^{e_1}) \neq \phi(\mathbf{t}^{e_2})$. It is easy to see that if we shift each x_i by $\phi(t_i)$ to get $D'(\mathbf{x}) = D(\mathbf{x} + \phi(\mathbf{t}))$ then $\det(D_{i\mathbf{0}}') = \det(D|_{\mathbf{x} = \phi(\mathbf{t})}) \neq 0$.

For sparse polynomials, Agrawal et al. [ASS13, Lemma 16] have given an efficient shift to achieve low-support concentration. Here, we rewrite their lemma. The map $\phi_{\ell'} \colon \mathbf{t} \to \{t^i\}_{i=0}^{\infty}$ is said to be separating ℓ'-support monomials of degree δ, if for any two monomials $\mathbf{t}^{e_1}$ and $\mathbf{t}^{e_2}$ which have support bounded by ℓ' and degree bounded by δ, $\phi_{\ell'}(\mathbf{t}^{e_1}) \neq \phi_{\ell'}(\mathbf{t}^{e_2})$. For a polynomial $D(\mathbf{x})$, let $\mu(D)$ be the maximum support of a monomial in D, i.e. $\mu(D) := \max_{\mathbf{x}^e \in \mathsf{S}(D)} \mathsf{s}(e)$.

LEMMA 30 ([ASS13]). *Let V be a $\mathbb{F}$-vector space of dimension k. Let $D(\mathbf{x}) \in V[\mathbf{x}]$ be a polynomial with degree bound δ. Let $\ell := 1 + 2\min\{\lceil \log(k \cdot \mathsf{s}(D)) \rceil, \mu(D)\}$ and ϕ_ℓ be a monomial map separating ℓ-support monomials of degree δ. Then $D(\mathbf{x} + \phi_\ell(\mathbf{t}))$ has ℓ-concentration over $\mathbb{F}(t)$.*

The [ASS13] version of the Lemma 30 gave a concentration result about sparse polynomials over $\mathbb{H}_k(\mathbb{F})$. But observe that the process of shifting and the definition of concentration only deal with the additive structure of $\mathbb{H}_k(\mathbb{F})$, and the multiplication structure is irrelevant. Hence, the result is true over any $\mathbb{F}$-vector space, in particular, over the matrix algebra. By combining these observations, we have the following.

LEMMA 31. *Let $D(\mathbf{x}) = \prod_{i=1}^{d} D_i(\mathbf{x}_i)$ be a polynomial in $\mathbb{F}^{w \times w}[\mathbf{x}]$ with $\det(D) \neq 0$ such that for all $i \in [d]$, D_i has degree bounded by δ, $\mathsf{s}(D_i) \leq s$ and $\mu(D_i) \leq \mu$. Let $\ell := 1 + 2\min\{\lceil \log(w^2 \cdot s) \rceil, \mu\}$ and $M := \mathrm{poly}(s^w(n\delta)^\ell)$. Then there is a set of M monomial maps with degree bounded by $M \log M$ such that for at least one of the maps ϕ, $C' := C(\mathbf{x} + \phi(\mathbf{t}))$ has $\ell(w^2 + 2)$-concentration.*

Proof. Let $\phi \colon \mathbf{t} \to \{t^i\}_{i=0}^{\infty}$ be a map such that it separates all the monomials in $\mathsf{S}(\det(D_i(\mathbf{t}_i)))$, for all $i \in [d]$. There are ds^{2w} such monomial pairs. Also assume

24

that ϕ separates all monomials of support bounded by ℓ. There are $(n\delta)^{O(\ell)}$ such monomials. Hence, total number of monomial pairs which need to be separated are $s^{O(w)} + (n\delta)^{O(\ell)}$. From Lemma 4, we know that there is a set of M monomial maps $(t_i \mapsto t^{w(t_i)})$ with highest degree $M \log M$ such that at least one of the maps ϕ separates the desired monomials, where $M = \text{poly}(s^w(n\delta)^\ell)$. As the map ϕ separates all the monomials in $\mathsf{S}(\det(D_i(\mathbf{t}_i)))$, $\det(D_i(\phi(\mathbf{t}_i))) \neq 0$ and hence, $D'_{i\mathbf{0}}$ is invertible for all $i \in [d]$. So, $C'(\mathbf{x})$ has $(w^2 + 2)$-block-support concentration from Lemma 27.

From Lemma 30, $D'_i(\mathbf{x}_i)$ has ℓ-concentration for all $0 \leq i \leq d+1$. Hence, from Lemma 29, $C'(\mathbf{x})$ has $\ell(w^2 + 2)$-concentration. $\square$

Now, we come back to the proof of Theorem 3 (restated in this section). Combining Lemma 31 with Lemma 23 we get a hitting set for $C'(\mathbf{x}) = C(\mathbf{x} + \phi(\mathbf{t}))$ of size $(n\delta)^{O(\ell w^2)}$. Each of these evaluations of C will be a polynomial in t with degree at most $\text{poly}(s^w(n\delta)^\ell)$. Hence, total time complexity becomes $\text{poly}(s^w(n\delta)^{\ell w^2})$.

5.3. Width-2 Read Once ABP. In the previous section, the crucial part in finding a hitting-set for an ROABP, is the assumption that the matrix product $D(\mathbf{x})$ is invertible. Now, we will show that for width-2 ROABP, this assumption is not required. Via a factorization property of 2×2 matrices, we will show that PIT for width-2 sparse-factor ROABP reduces to PIT for width-2 sparse-invertible-factor ROABP.

LEMMA 32 (2×2 invertibility). *Let* $C(\mathbf{x}) = D_0^\top \left(\prod_{i=1}^d D_i \right) D_{d+1}$ *be a polynomial computed by a width-2 sparse-factor ROABP. Then we can write* $\alpha(\mathbf{x})C(\mathbf{x}) = C_1(\mathbf{x})C_2(\mathbf{x}) \cdots C_{m+1}(\mathbf{x})$, *for some nonzero* $\alpha \in \mathbb{F}[\mathbf{x}]$ *and some* $m \leq d$, *where* $C_i(\mathbf{x})$ *is a polynomial computed by a width-2 sparse-invertible-factor ROABP, for all* $i \in [m+1]$.

Proof. Let us say, for some $i \in [d]$, $D_i(\mathbf{x}_i)$ is not invertible. Let $D_i = \begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix}$ with $a_i, b_i, c_i, d_i \in \mathbb{F}[\mathbf{x}_i]$ and $a_i d_i = b_i c_i$. Without loss of generality, at least one of $\{a_i, b_i, c_i, d_i\}$ is nonzero. Let us say $a_i \neq 0$ (other cases are similar). Then we can write,

$$\begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix} = \frac{1}{a_i} \begin{bmatrix} a_i \\ c_i \end{bmatrix} \begin{bmatrix} a_i & b_i \end{bmatrix}.$$

In other words, we can write $\alpha_i D_i = A_i B_i^\top$, where $A_i, B_i \in \mathbb{F}^2[\mathbf{x}_i]$ and $0 \neq \alpha_i \in \{a_i, b_i, c_i, d_i\}$. Note that $\mathsf{s}(\alpha_i), \mathsf{s}(A_i), \mathsf{s}(B_i) \leq \mathsf{s}(D_i)$. Let us say that the set of non-invertible D_is is $\{D_{i_1}, D_{i_2}, \ldots, D_{i_m}\}$. Writing all of them in the above form we get,

$$C(\mathbf{x}) \prod_{j=1}^m \alpha_{i_j} = \prod_{j=1}^{m+1} C_j,$$

where

$$C_j := \begin{cases} D_0^\top \left(\prod_{i=1}^{i_1-1} D_i \right) A_{i_1} & \text{if } j = 1, \\ B_{i_{j-1}}^\top \left(\prod_{i=i_{j-1}+1}^{i_j-1} D_i \right) A_{i_j} & \text{if } 2 \leq j \leq m, \\ B_{i_m}^\top \left(\prod_{i=i_m+1}^d D_i \right) D_{d+1} & \text{if } j = m+1. \end{cases}$$

Clearly, for all $j \in [m+1]$, C_j can be computed by a sparse-invertible-factor ROABP. $\square$

Now, from the above lemma it is easy to construct a hitting-set. First we write a general result about hitting-sets for a product of polynomials from some class [SY10, Observation 4.1].

LEMMA 33 (Lagrange interpolation). *Suppose $\mathcal{H}$ is a hitting-set for a class of polynomials $\mathcal{C}$. Let $C(\mathbf{x}) = C_1(\mathbf{x})C_2(\mathbf{x})\cdots C_m(\mathbf{x})$, where $C_i \in \mathcal{C}$ and has degree bounded by δ, for all $i \in [m]$. There is a hitting-set of size $m\delta|\mathcal{H}| + 1$ for $C(\mathbf{x})$.*

Proof. Let $h = |\mathcal{H}|$ and $\mathcal{H} = \{\alpha_1, \alpha_2, \ldots, \alpha_h\}$. Let $B := \{\beta_i\}_{i=1}^{h}$ be a set of constants. The Lagrange interpolation $\alpha(u)$ of the points in $\mathcal{H}$ is defined as follows

$$\alpha(u) := \sum_{i=1}^{h} \frac{\prod_{j \neq i}(u - \beta_j)}{\prod_{j \neq i}(\beta_i - \beta_j)} \alpha_i.$$

The key property of the interpolation is that when we put $u = \beta_i$, $\alpha(\beta_i) = \alpha_i$ for all $i \in [h]$. For any $a \in [m]$, we know that $C_a(\alpha_i) \neq 0$, for some $i \in [h]$. Hence, $C_a(\alpha(u))$ as a polynomial in u is nonzero because $C_a(\alpha(\beta_i)) = C_a(\alpha_i) \neq 0$. So, we can say $C(\alpha(u)) \neq 0$ as a polynomial in u. Degree of $\alpha(u)$ is h. So, degree of $C(\alpha(u))$ in u is bounded by $m\delta h$. We can put $(m\delta h + 1)$-many distinct values of u to get a hitting-set for $C(\alpha(u))$. $\square$

Note that a hitting-set for $\alpha(\mathbf{x})C(\mathbf{x})$ is also a hitting-set for $C(\mathbf{x})$ if α is a nonzero polynomial. Recall that we get a hitting-set for invertible ROABP from Theorem 3. Lemma 32 tells us how to write a width-2 ROABP as a product of width-2 invertible ROABPs. Combining these results with Lemma 33 we directly get the following.

THEOREM 34. *Let $C(\mathbf{x}) = D_0^{\top}(\mathbf{x}_0)(\prod_{i=1}^{d} D_i(\mathbf{x}_i))D_{d+1}(\mathbf{x}_{d+1})$ be a polynomial in $\mathbb{F}[\mathbf{x}]$ computed by a width-2 ROABP such that for all $0 \leq i \leq d + 1$, D_i has degree bounded by δ, $\mathsf{s}(D_i) \leq s$ and $\mu(D_i) \leq \mu$. Let $\ell := 1 + 2\min\{\lceil\log(4 \cdot s)\rceil, \mu\}$. Then there is a hitting-set of size $\mathrm{poly}((n\delta s)^{\ell})$.*

We remark again that when all D_is are constant-variate or linear polynomials, the hitting-set is polynomial-time.

6. Discussion. The first open problem is to do basis isolation for ROABP with only a polynomially large weight assignment. Also, our technique of finding a basis isolating weight assignment seems general. It needs to be explored, for what other general classes can it be applied. In particular, can it be used to solve depth-3 multilinear circuits? An easier question, perhaps, could be to improve Theorem 18 to get a truly blackbox PIT for the 2-base-sets-1-distance model.

Another question is whether we can find a similar result in the boolean setting, i.e. get a psuedorandom generator for unknown order ROBP with seed length same as the known order case.

In the case of constant width ROABP, we could show constant-support concentration, but only after assuming that the factor matrices are invertible. It seems that the invertibility assumption restricts the computing power of ROABP significantly. It is desirable to have low-support concentration without the assumption of invertibility.

As in the case of invertible ROABP and width-2 ROABP, analogous results hold in the boolean setting, it will be interesting to see if there is some connection, at the level of techniques, between pseudorandom generators for boolean and arithmetic models.

REFERENCES

[AGKS13]   Manindra Agrawal, Rohit Gurjar, Arpita Korwar, and Nitin Saxena, *Hitting-sets for low-distance multilinear depth*-3, Electronic Colloquium on Computational Complexity (ECCC) **20** (2013), 174.

[Agr05]   Manindra Agrawal, *Proving lower bounds via pseudo-random generators.*, FSTTCS, Lecture Notes in Computer Science, vol. 3821, 2005, pp. 92–105.

[ASS13]   Manindra Agrawal, Chandan Saha, and Nitin Saxena, *Quasi-polynomial hitting-set for set-depth- formulas*, STOC, 2013, pp. 321–330.

[ASSS12]   Manindra Agrawal, Chandan Saha, Ramprasad Saptharishi, and Nitin Saxena, *Jacobian hits circuits: hitting-sets, lower bounds for depth-d occur-k formulas & depth-3 transcendence degree-k circuits*, STOC, 2012, pp. 599–614.

[BDVY13]   Andrej Bogdanov, Zeev Dvir, Elad Verbin, and Amir Yehudayoff, *Pseudorandomness for width-2 branching programs*, Theory of Computing **9** (2013), 283–293.

[BOC92]   Michael Ben-Or and Richard Cleve, *Computing algebraic formulas using a constant number of registers*, SIAM J. Comput. **21** (1992), no. 1, 54–58.

[De11]   Anindya De, *Pseudorandomness for permutation and regular branching programs*, IEEE Conference on Computational Complexity, 2011, pp. 221–231.

[DS07]   Zeev Dvir and Amir Shpilka, *Locally decodable codes with two queries and polynomial identity testing for depth 3 circuits*, SIAM J. Comput. **36** (2007), no. 5, 1404–1434.

[FS12]   Michael A. Forbes and Amir Shpilka, *On identity testing of tensors, low-rank recovery and compressed sensing*, STOC, 2012, pp. 163–172.

[FS13]   ———, *Quasipolynomial-time identity testing of non-commutative and read-once oblivious algebraic branching programs*, FOCS, 2013, pp. 243–252.

[FSS14]   Michael A. Forbes, Ramprasad Saptharishi, and Amir Shpilka, *Pseudorandomness for multilinear read-once algebraic branching programs, in any order*, STOC, 2014.

[GKKS13]   Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi, *Arithmetic circuits: A chasm at depth three*, FOCS (2013).

[IMZ12]   Russell Impagliazzo, Raghu Meka, and David Zuckerman, *Pseudorandomness from shrinkage*, 2013 IEEE 54th Annual Symposium on Foundations of Computer Science **0** (2012), 111–119.

[JQS10a]   Maurice J. Jansen, Youming Qiao, and Jayalal Sarma, *Deterministic black-box identity testing \$pi\$-ordered algebraic branching programs*, FSTTCS, 2010, pp. 296–307.

[JQS10b]   ———, *Deterministic identity testing of read-once algebraic branching programs*, Electronic Colloquium on Computational Complexity (ECCC) **17** (2010), 84.

[KNP11]   Michal Koucký, Prajakta Nimbhorkar, and Pavel Pudlák, *Pseudorandom generators for group products: extended abstract*, STOC, 2011, pp. 263–272.

[Kro82]   Leopold Kronecker, *Grundzuge einer arithmetischen theorie der algebraischen grossen*, Berlin, G. Reimer, 1882.

[KS01]   Adam Klivans and Daniel A. Spielman, *Randomness efficient identity testing of multivariate polynomials*, STOC, 2001, pp. 216–223.

[KS07]   Neeraj Kayal and Nitin Saxena, *Polynomial identity testing for depth 3 circuits*, Computational Complexity **16** (2007), no. 2, 115–138.

[KS09]   Neeraj Kayal and Shubhangi Saraf, *Blackbox polynomial identity testing for depth 3 circuits*, FOCS, 2009, pp. 198–207.

[KS11]   Zohar Shay Karnin and Amir Shpilka, *Black box polynomial identity testing of generalized depth-3 arithmetic circuits with bounded top fan-in*, Combinatorica **31** (2011), no. 3, 333–364.

[Mul12a]   Ketan D. Mulmuley, *The gct program toward the p vs. np problem*, Commun. ACM **55** (2012), no. 6, 98–107.

[Mul12b]   ———, *Geometric complexity theory V: Equivalence between blackbox derandomization of polynomial identity testing and derandomization of Noether's normalization lemma*, FOCS, 2012, pp. 629–638.

[MV97]   Meena Mahajan and V. Vinay, *Determinant: Combinatorics, algorithms, and complexity*, Chicago J. Theor. Comput. Sci. **1997** (1997).

[Nis90]   N. Nisan, *Pseudorandom generators for space-bounded computations*, Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '90, ACM, 1990, pp. 204–212.

[Nis91]   Noam Nisan, *Lower bounds for non-commutative computation (extended abstract)*, Proceedings of the 23rd ACM Symposium on Theory of Computing, ACM Press, 1991, pp. 410–418.

[RS05]   Ran Raz and Amir Shpilka, *Deterministic polynomial identity testing in non-*

	commutative models., Computational Complexity **14** (2005), no. 1, 1–19.
[RSV13]	Omer Reingold, Thomas Steinke, and Salil P. Vadhan, *Pseudorandomness for regular branching programs via fourier analysis*, APPROX-RANDOM, 2013, pp. 655–670.
[RY09]	Ran Raz and Amir Yehudayoff, *Lower bounds and separations for constant depth multilinear circuits*, Computational Complexity **18** (2009), no. 2, 171–207.
[Sax08]	Nitin Saxena, *Diagonal circuit identity testing and lower bounds*, ICALP, Lecture Notes in Computer Science, vol. 5125, Springer, 2008, pp. 60–71.
[Sax09]	———, *Progress on polynomial identity testing*, Bulletin of the EATCS **99** (2009), 49–79.
[Sax14]	———, *Progress on polynomial identity testing - 2*, CoRR **abs/1401.0976** (2014).
[Sch80]	Jacob T. Schwartz, *Fast probabilistic algorithms for verification of polynomial identities*, J. ACM **27** (1980), no. 4, 701–717.
[SS11]	Nitin Saxena and C. Seshadhri, *An almost optimal rank bound for depth-3 identities*, SIAM J. Comput. **40** (2011), no. 1, 200–224.
[SS12]	———, *Blackbox identity testing for bounded top-fanin depth-3 circuits: The field doesn't matter*, SIAM J. Comput. **41** (2012), no. 5, 1285–1298.
[SS13]	———, *From sylvester-gallai configurations to rank bounds: Improved blackbox identity test for depth-3 circuits*, J. ACM **60** (2013), no. 5, 33.
[SSS09]	Chandan Saha, Ramprasad Saptharishi, and Nitin Saxena, *The power of depth 2 circuits over algebras*, FSTTCS, 2009, pp. 371–382.
[SSS13]	———, *A case of depth-3 identity testing, sparse factorization and duality.*, Computational Complexity **22** (2013), no. 1, 39–69.
[Ste12]	Thomas Steinke, *Pseudorandomness for permutation branching programs without the group theory.*, Electronic Colloquium on Computational Complexity (ECCC) **19** (2012), 83.
[SVW14]	Thomas Steinke, Salil P. Vadhan, and Andrew Wan, *Pseudorandomness and fourier growth bounds for width 3 branching programs*, CoRR **abs/1405.7028** (2014).
[SY10]	Amir Shpilka and Amir Yehudayoff, *Arithmetic circuits: A survey of recent results and open questions*, Foundations and Trends in Theoretical Computer Science **5** (2010), no. 3-4, 207–388.
[Tod91]	Seinosuke Toda, *Counting problems computationally equivalent to computing the determinant*, 1991.