Fast Exact Algorithms Using Hadamard Product of Polynomials

V. Arvind
Institute of Mathematical Sciences (HBNI), Chennai, India
arvind@imsc.res.in

Abhranil Chatterjee
Institute of Mathematical Sciences (HBNI), Chennai, India
abhranilc@imsc.res.in

Rajit Datta
Chennai Mathematical Institute, Chennai, India
rajit@cmi.ac.in

Partha Mukhopadhyay
Chennai Mathematical Institute, Chennai, India
partham@cmi.ac.in

Abstract

Let C be an arithmetic circuit of poly(n) size given as input that computes a polynomial $f \in \mathbb{F}[X]$, where $X = \{x_1, x_2, \ldots, x_n\}$ and $\mathbb{F}$ is any field where the field arithmetic can be performed efficiently. We obtain new algorithms for the following two problems first studied by Koutis and Williams [13, 22, 14].

(k,n)-MLC: Compute the sum of the coefficients of all degree-k multilinear monomials in the polynomial f.

k-MMD: Test if there is a nonzero degree-k multilinear monomial in the polynomial f.

Our algorithms are based on the fact that the Hadamard product $f \circ S_{n,k}$, is the degree-k multilinear part of f, where $S_{n,k}$ is the k^{th} elementary symmetric polynomial.

- For (k,n)-MLC problem, we give a deterministic algorithm of run time $O^*(n^{k/2+c\log k})$ (where c is a constant), answering an open question of Koutis and Williams [14, ICALP'09]. As corollaries, we show $O^*(\binom{n}{\downarrow k/2})$-time exact counting algorithms for several combinatorial problems: k-Tree, t-Dominating Set, m-Dimensional k-Matching.
- For k-MMD problem, we give a randomized algorithm of run time $4.32^k \cdot \text{poly}(n,k)$. Our algorithm uses only poly(n,k) space. This matches the run time of a recent algorithm [8] for k-MMD which requires exponential (in k) space.

Other results include fast deterministic algorithms for (k,n)-MLC and k-MMD problems for depth three circuits.

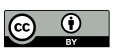
2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation

Keywords and phrases Hadamard Product, Multilinear Monomial Detection and Counting, Rectangular Permanent, Symmetric Polynomial

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2019.9

Related Version A full version of the paper is available at `https://arxiv.org/abs/1807.04496`.

Acknowledgements We thank anonymous reviewers for their comments on an earlier version of this paper. We are particularly grateful to an anonymous reviewer for pointing out the combinatorial applications of Theorem 1 in exact counting.



1 Introduction

Koutis and Williams [13, 22, 14] introduced and studied two algorithmic problems on arithmetic circuits. Given as input an arithmetic circuit C of $\mathrm{poly}(n)$ size computing a polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$, the (k,n)-MlC problem is to compute the sum of the coefficients of all degree-k multilinear monomials in the polynomial f, and the k-MmD problem is to test if f has a nonzero degree-k multilinear monomial.

These problems are natural generalizations of the well-studied k-path detection and counting problems in a given graph [13] as well as several other combinatorial problems like k-Tree, t-Dominating Set, m-Dimensional k-Matching [14], well-studied in the parameterized complexity, reduce to these problems. In fact, the first randomized FPT algorithms for the decision version of these combinatorial problems were obtained from an $O^*(2^k)$ [1] algorithm for k-MmD for monotone circuits using group algebras [13, 22, 14]. Recently, Brand et al. [8] have given the first randomized FPT algorithm for k-MmD for general circuits that runs in time $O^*(4.32^k)$. Their method is based on exterior algebra and color coding [1].

In general, the exact counting versions of these problems are #W[1]-hard. For these counting problems, improvements to the trivial $O^*(n^k)$ time exhaustive search algorithm are known only in some cases (like counting k-paths) [6]. Since an improvement for (k,n)-MlC over exhaustive search will yield faster exact counting algorithms for all these problems, Koutis and Williams [14] pose this as an interesting open problem. They give an algorithm of run time $O^*(n^{k/2})$ to compute the *parity* of the sum of coefficients of degree-k multilinear monomials.

The techniques based on group algebra [13, 14] and exterior algebra [8] can be broadly classified as *multilinear algebra* techniques. We give a new approach to the k-MmD, (k,n)-MlC problems, and related problems. Our algorithm is based on computing the *Hadamard product of polynomials*. The Hadamard product (also known as Schur product) generally refers to Hadamard product of matrices and is used in matrix analysis. We consider the Hadamard product of polynomials (e.g., see [3]). Given polynomials $f, g \in \mathbb{F}[X]$, their Hadamard product is defined as $f \circ g = \sum_m ([m]f \cdot [m]g)m$, where $[m]f$ denotes the coefficient of monomial m in f.

The Hadamard product is a useful tool in *noncommutative* computation [3, 5]. A contribution of the present paper is to develop an efficient way to implement Hadamard product in the *commutative* setting which is useful for designing FPT and exact algorithms. As mentioned above, the Hadamard product has been useful in arithmetic circuit complexity results, e.g., showing hardness of the noncommutative determinant [5]. Transferring techniques from circuit complexity to algorithm design is an exciting area of research. We refer the reader to the survey article of Williams [21], see also [23].

This paper. We apply the Hadamard product of polynomials in the setting of commutative computation. This is achieved by combining earlier ideas [3, 5] with a symmetrization trick shown in Section 2. We then use it to design efficient algorithms for (k,n)-MlC, k-MmD and related problems.

Consider the elementary symmetric polynomial $S_{n,k}$ of degree k over the n variables $x_1, x_2, \ldots, x_n$. By definition, $S_{n,k}$ is the sum of all the degree-k multilinear monomials. Computing the Hadamard product of $S_{n,k}$ and a polynomial f *sieves out* precisely the degree-k multilinear part of f. This connection with the symmetric polynomial gives the following result.

[1] The O^* notation suppresses polynomial factors.

▶ **Theorem 1.** *The* (k,n)-MLC *problem for any arithmetic circuit* C *of* $\mathrm{poly}(n)$ *size, has a deterministic* $O^*(n^{k/2+c\log k})$ *time algorithm where* c *is a constant.*

The field $\mathbb{F}$ could be any field where the field operations can be efficiently computable. The above run time $O^*(n^{k/2+c\log k})$ (where c is a constant) beats the naive $O^*(n^k)$ bound, answering the question asked by Koutis and Williams [14].

An ingredient of the proof is a result in [5] that allows us to efficiently compute the Hadamard product of a noncommutative algebraic branching program (ABP) with a noncommutative polynomial f, even with only black-box access to f that allows evaluating f on matrix-valued inputs. The other ingredient is an algorithm of Björklund et al. [7] for evaluating rectangular permanent over noncommutative rings, that can be viewed as an algorithm for evaluating $S^*_{n,k}$ (a symmetrized noncommutative version of $S_{n,k}$) over matrices. Now, applying the routine conversion of a commutative circuit to an ABP, which incurs only a quasi-polynomial blow-up, we get a faster algorithm for (k,n)-MLC of general circuits. As applications of Theorem 1 we obtain improved counting algorithms for k-Tree, t-Dominating Set, and m-Dimensional k-Matching.

The next algorithmic result we obtain is the following.

▶ **Theorem 2.** *The* k-MMD *problem for any arithmetic circuit* C *of* $\mathrm{poly}(n)$ *size, has a randomized* $O^*(4.32^k)$ *time and polynomial space-bounded algorithm.*

Again, the field $\mathbb{F}$ could be any field where the field operation can be efficiently computable. We briefly sketch the proof idea. Suppose that C is the input arithmetic circuit computing a homogeneous polynomial f of degree k. We essentially show that k-MMD is reducible to checking if the Hadamard product $f \circ C'$ is nonzero for some circuit C' from a collection of homogeneous degree-k depth two circuits. This collection of depth two circuits arises from the application of color coding [1]. Furthermore, the commutative Hadamard product $f \circ C'$ turns out to be computable in $O^*(2^k)$ time by a symmetrization trick combined with Ryser's formula for the permanent. The overall running time (because of trying several choices for C') turns out to be $O^*(4.32^k)$. Finally, checking if $f \circ C'$ is nonzero reduces to an instance of polynomial identity testing which can be solved in randomized polynomial time using Demillo-Lipton-Schwartz-Zippel Lemma [9, 24, 18]. The technique based on Hadamard product seems to be quite different than the exterior algebra based technique. Another difference is that, our algorithm uses $\mathrm{poly}(n,k)$ space whereas the algorithm in [8] takes exponential space.

Next, we state the results showing fast deterministic algorithms for depth-three circuits. We use the notation $\Sigma^{[s]}\Pi^{[k]}\Sigma$ to denote depth three circuits of top Σ gate fan-in s and the Π gates compute the product of k homogeneous linear forms over X.

▶ **Theorem 3.** *Given any homogeneous depth three* $\Sigma^{[s]}\Pi^{[k]}\Sigma$ *circuit of degree* k*, the* (k,n)-MLC *problem can be solved in deterministic* $O^*(2^k)$ *time. Over* $\mathbb{Z}$*, the* k-MMD *problem can be solved in deterministic* $O^*(4^k)$ *time. Over finite fields,* k-MMD *problem can be solved in deterministic* $e^k k^{O(\log k)}(2^{ck}+2^k)\cdot\mathrm{poly}(n,k,s)$ *time, where* $c \leq 5$*.*

Here the key observation is that we can efficiently compute the commutative Hadamard product of a depth three circuit with *any* circuit. It is well-known that the elementary symmetric polynomial $S_{n,k}$ can be computed using an algebraic branching program of size $\mathrm{poly}(n,k)$.

We compute the Hadamard product of the given depth three circuit with that homogeneous branching program for $S_{n,k}$, and check whether the resulting depth three circuit is identically zero or not. The same idea yields the algorithm to compute the sum of the coefficients of the multilinear terms as well.

Related Work. Soon after the first version of our paper [2] appeared in ArXiv, an independent work [16, v1] [2] also considers the k-MmD and (k,n)-MlC problems. The main ingredient of [16] is the application of a nontrivial Waring decomposition over rationals of symmetric polynomials [15] which does not have any known analogue for small finite fields. The algorithms obtained for k-MmD and (k,n)-MlC are faster ($O^*(4.08^k)$ time for k-MmD and $O^*(n^{k/2})$ for (k,n)-MlC). In comparison, our algorithms also work for all finite fields. As already mentioned, the algorithm of Koutis and Williams [14] for (k,n)-MlC works over $\mathbb{F}_2$ and the run time is $O^*(n^{k/2})$. In this sense, our algorithm for (k,n)-MlC can also be viewed as a generalization that does not depend on the characteristic of the ground field. It is to be noted that, over fields of small characteristic a Waring decomposition of the input polynomial may not be available. For example, over $\mathbb{F}_2$ the polynomial xy has no Waring decomposition.

Organization. The paper is organized as follows. In Section 2 we explain the Hadamard product framework. The proof of Theorem 1 and its consequences are given in Section 3. Section 4 contains the the proof of Theorem 2. The proof of Theorem 3 can be found in the full version in ArXiv.

2 Hadamard Product Framework

Computing the Hadamard product of two commutative polynomials is, in general, computationally hard. This can be observed from the fact that the Hadamard product of the determinant polynomial with itself is the permanent polynomial. Nevertheless, we develop a method for some special cases, that is efficient with degree k as the fixed parameter, for computing the *scaled* Hadamard product of commutative polynomials.

▶ **Definition 4.** *The* scaled *Hadamard product of polynomials $f, g \in \mathbb{F}[X]$ is defined as*

$$f \circ^s g = \sum_m (m! \cdot [m]f \cdot [m]g)\ m,$$

where for monomial $m = x_{i_1}^{e_1} x_{i_2}^{e_2} \dots x_{i_r}^{e_r}$ we define $m! = e_1! \cdot e_2! \cdots e_r!$.

Computing the scaled Hadamard product is key to our algorithmic results for k-MmD and (k,n)-MlC. Broadly, it works as follows: we transform polynomials f and g to suitable *noncommutative* polynomials. We compute their (noncommutative) Hadamard product efficiently [3, 5], and we finally recover the scaled commutative Hadamard product $f \circ^s g$ (or evaluate it at a desired point $\vec{a} \in \mathbb{F}^n$).

Suppose $f \in \mathbb{F}[x_1, x_2, \dots, x_n]$ is a homogeneous degree-k polynomial given by a circuit C. We can define its noncommutative version C^{nc} which computes the noncommutative homogeneous degree-k polynomial $\hat{f} \in \mathbb{F}\langle y_1, y_2, \dots, y_n\rangle$ as follows.

▶ **Definition 5.** *Given a commutative circuit C computing a polynomial in $\mathbb{F}[x_1, x_2, ..., x_n]$, the noncommutative version of C, C^{nc} is the noncommutative circuit obtained from C by fixing an ordering of the inputs to each product gate in C and replacing x_i by the noncommuting variable $y_i : 1 \le i \le n$.*

[2] See the final version [16] to be appeared in FOCS 2019.

Let X_k denote the set of all degree-k monomials over X. As usual, Y^k denotes all degree-k noncommutative monomials (i.e., words) over Y. Each monomial $m \in X_k$ can appear as different noncommutative monomials $\hat{m}$ in $\hat{f}$. We use the notation $\hat{m} \to m$ to denote that $\hat{m} \in Y^k$ will be transformed to $m \in X_k$ by substituting x_i for $y_i, 1 \leq i \leq n$. Then, we observe the following, $[m]f = \sum_{\hat{m}\to m}[\hat{m}]\hat{f}$.

The noncommutative circuit C^{nc} is not directly useful for computing Hadamard product. However, the following symmetrization helps. We first explain how permutations $\sigma \in S_k$ act on the set Y^k of degree-k monomials (and hence, by linearity, act on homogeneous degree k polynomials).

For each monomial $\hat{m} = y_{i_1}y_{i_2}\cdots y_{i_k}$, the permutation $\sigma \in S_k$ maps $\hat{m}$ to the monomial $\hat{m}^\sigma$ defined as $\hat{m}^\sigma = y_{i_{\sigma(1)}}y_{i_{\sigma(2)}}\cdots y_{i_{\sigma(k)}}$. By linearity, $\hat{f} = \sum_{\hat{m}\in Y^k}[\hat{m}]\hat{f}\cdot\hat{m}$ is mapped by σ to the polynomial, $\hat{f}^\sigma = \sum_{\hat{m}\in Y^k}[\hat{m}]\hat{f}\cdot\hat{m}^\sigma$.

The *symmetrized polynomial* of f, f^*, is degree-k homogeneous polynomial $f^* = \sum_{\sigma\in S_k}\hat{f}^\sigma$. We now explain the use of symmetrization in computing the scaled Hadamard product $f \circ^s g$.

▶ **Proposition 6.** *For a homogeneous degree-k commutative polynomial $f \in \mathbb{F}[X]$ given by circuit C, and its noncommutative version C^{nc} computing polynomial $\hat{f} \in \mathbb{F}\langle Y\rangle$, consider the* symmetrized *noncommutative polynomial $f^* = \sum_{\sigma\in S_k}\hat{f}^\sigma$. Then for each monomial $m \in X_k$ and each word $m' \in Y^k$ such that $m' \to m$, we have: $[m']f^* = m!\cdot[m]f$.*

Proof. Let $f = \sum_m [m]f\cdot m$ and $\hat{f} = \sum_{\hat{m}}[\hat{m}]\hat{f}\cdot\hat{m}$. Notice that $[m]f = \sum_{\hat{m}\to m}[\hat{m}]\hat{f}$. Now, we write $f^* = \sum_{m'}[m']f^*\cdot m'$. The group S_k acts on Y^k (degree k words in Y) by permuting the positions. Suppose $m = x_{i_1}^{e_1}\cdots x_{i_q}^{e_q}$ is a type $\boldsymbol{e} = (e_1,\ldots,e_q)$ degree k monomial over X and $m' \to m$. Then, by the *Orbit-Stabilizer lemma* the orbit $O_{m'}$ of m' has size $\frac{k!}{m!}$. It follows that

$$[m']f^* = \sum_{\hat{m}\in O_{m'}} m!\cdot[\hat{m}]\hat{f} = m!\sum_{\hat{m}\to m}[\hat{m}]\hat{f} = m!\cdot[m]f.$$

It is important to note that for some $\hat{m} \in Y^k$ such that $\hat{m} \to m$, even if $[\hat{m}]\hat{f} = 0$ then also $[\hat{m}]f^* = m!\cdot[m]f$. ◀

Next, we show how to use Proposition 6 to compute scaled Hadamard product in the commutative setting via noncommutative Hadamard product. We note that given a commutative circuit C computing f, the noncommutative polynomial $\hat{f}$ depends on the circuit structure of C. However, f^* depends only on the polynomial f.

▶ **Lemma 7.** *Let C be a circuit for a homogeneous degree-k polynomial $g \in \mathbb{F}[X]$. For any homogeneous degree-k polynomial $f \in \mathbb{F}[X]$, to compute a circuit for $f \circ^s g$ efficiently, it suffices to compute a circuit for $f^* \circ \hat{g}$ efficiently where $\hat{g}$ is the polynomial computed by the noncommutative circuit C^{nc}. Moreover, given any point $\vec{a} \in \mathbb{F}^n$, $(f \circ^s g)(\vec{a}) = (f^* \circ \hat{g})(\vec{a})$.*

Proof. We write $f = \sum_m [m]f\cdot m$ and $g = \sum_{m'}[m']g\cdot m'$, and notice that $f \circ^s g = \sum_m m!\cdot[m]f\cdot[m]g\cdot m$.

Suppose the polynomial computed by C^{nc} is $\hat{g}(Y) = \sum_{m\in X_k}\sum_{\hat{m}\to m}[\hat{m}]\hat{g}\cdot\hat{m}$. By Proposition 6, the noncommutative polynomial $f^*(Y) = \sum_{m\in X_k}\sum_{\hat{m}\to m} m!\cdot[m]f\cdot\hat{m}$. Hence,

$$(f^*\circ\hat{g})(Y) = \sum_{m\in X_k}\sum_{\hat{m}\to m} m!\cdot[m]f\cdot[\hat{m}]g\cdot\hat{m} = \sum_{m\in X_k} m!\cdot[m]f\sum_{\hat{m}\to m}[\hat{m}]g\cdot\hat{m}.$$

Therefore, using any commutative substitution (i.e. by substituting the Y variables by X variables), we get back a commutative circuit for $f \circ^s g$. Moreover, given a point $\vec{a} \in \mathbb{F}^n$,

$$(f^* \circ \hat{g})(\vec{a}) = \sum_{m \in X_k} m! \cdot [m]f \sum_{\hat{m} \to m} [\hat{m}]g \cdot \hat{m}(\vec{a}) = \sum_{m \in X_k} m! \cdot [m]f \cdot m(\vec{a}) \sum_{\hat{m} \to m} [\hat{m}]g.$$

From the definition, $[m]g = \sum_{\hat{m} \to m} \hat{m}[\hat{g}]$. Hence, $(f^* \circ \hat{g})(\vec{a}) = \sum_{m \in X_k} m! \cdot [m]f \cdot m(\vec{a})[m]g = (f \circ^s g)(\vec{a})$. ◀

3 The Sum of Coefficients of Multilinear Monomials

In this section we prove Theorem 1. As already sketched in Section 1, the main conceptual step is to apply the symmetrization trick to reduce the (k,n)-MLC problem to evaluating rectangular permanent over a suitable matrix ring. Then we use a result of [7] to solve the instance of rectangular permanent evaluation problem. As corollaries of our technique, we improve the running time of exact counting of several combinatorial problems studied in [14].

Before we prove the theorem, let us recall the definition of an ABP. An *algebraic branching program* (ABP) is a directed acyclic graph with one in-degree-0 vertex called *source*, and one out-degree-0 vertex called *sink*. The vertex set of the graph is partitioned into layers $0, 1, \ldots, \ell$, with directed edges only between adjacent layers (i to $i+1$). The source and the sink are at layers zero and ℓ respectively. Each edge is labeled by a linear form over variables $x_1, x_2, \ldots, x_n$. The polynomial computed by the ABP is the sum over all source-to-sink directed paths of the product of linear forms that label the edges of the path. An ABP is *homogeneous* if all edge labels are homogeneous linear forms. ABPs can be defined in both commutative and noncommutative settings. Equivalently, a homogeneous ABP of width w computing a degree-k polynomial over X can be thought of as the $(1,w)^{th}$ entry of the product of $w \times w$ matrices $M_1 \cdots M_k$ where entries of each M_i are homogeneous linear forms over X. By $[x_j]M_i$, we denote the $w \times w$ matrix over $\mathbb{F}$, such that $(p,q)^{th}$ entry of the matrix,$([x_j]M_i)(p,q) = [x_j](M_i(p,q))$, the coefficient of x_j in the linear form of the $(p,q)^{th}$ entry of M_i.

We now define the permanent of a rectangular matrix. The permanent of a rectangular $k \times n$ matrix $A = (a_{ij})$, with $k \leq n$ is defined as $\mathrm{rPer}(A) = \sum_{\sigma \in I_{k,n}} \prod_{i=1}^{k} a_{i,\sigma(i)}$ where $I_{k,n}$ is the set of all injections from $[k]$ to $[n]$. Also, we define the noncommutative polynomial $S^*_{n,k}$ as $S^*_{n,k}(y_1, y_2, \ldots, y_n) = \sum_{T \subseteq [n], |T|=k} \sum_{\sigma \in S_k} \prod_{i \in T} y_{\sigma(i)}$ which is the symmetrized version of the elementary symmetric polynomial $S_{n,k}$ as defined in Proposition 6. Given a set of matrices $M_1, \ldots, M_n$ define the rectangular matrix $A = (a_{i,j})_{i \in [k], j \in [n]}$ such that $a_{i,j} = M_j$. Now we make the following crucial observation.

▶ **Observation 8.**

$$S^*_{n,k}(M_1, \ldots, M_n) = \mathrm{rPer}(A).$$

We use a result from [7], that shows that over *any* ring R, the permanent of a rectangular $k \times n$ matrix can be evaluated using $O^*(k\binom{n}{\downarrow k/2})$ ring operations. In particular, if R is a matrix ring $M_s(\mathbb{F})$, the algorithm runs in time $O(k\binom{n}{\downarrow k/2}\,\mathrm{poly}(n,s))$. Now we are ready to prove Theorem 1.

Proof. Let us first proof a special case of the theorem when the polynomial f is given by an ABP B of width s. Notice that, we can compute the sum of the coefficients of the degree-k multilinear terms by evaluating $(f \circ S_{n,k})(\vec{1})$. Now to compute the Hadamard product efficiently, we transfer the problem to the noncommutative domain. Let B^{nc} defines

the noncommutative version of the commutative ABP B computing the polynomial f. From Lemma 7, it suffices to compute $(B^{nc} \circ S^*_{n,k})(\vec{1})$. Now, the following lemma reduces this to evaluating $S^*_{n,k}$ over matrix ring. We recall the following result from [5].

▶ **Lemma 9** (Theorem 2 of [4]). *Let f be a homogeneous degree-k noncommutative polynomial in $\mathbb{F}\langle Y\rangle$ and B be an ABP of width w computing a homogeneous degree-k polynomial $g = (M_1 \cdots M_k)(1, w)$ in $\mathbb{F}\langle Y\rangle$. Then $(f \circ g)(\vec{1}) = (f(A^B_1, \ldots, A^B_n))(1, (k+1)w)$ where for each $i \in [n]$, A^B_i is the following $(k+1)w \times (k+1)w$ block superdiagonal matrix,*

$$A^B_i = \begin{bmatrix} 0 & [y_i]M_1 & 0 & \ldots & 0 \\ 0 & 0 & [y_i]M_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & [y_i]M_k \\ 0 & 0 & 0 & \ldots & 0 \end{bmatrix}.$$

To see the proof, for any monomial $m = y_{i_1} y_{i_2} \cdots y_{i_k} \in Y^k$,

$$(A^B_{i_1} A^B_{i_2} \cdots A^B_{i_k})(1, (k+1)w) = ([y_{i_1}]M_1 \cdot [y_{i_2}]M_2 \cdots [y_{i_k}]M_k)(1, w) = [m]g,$$

from the definition. Hence, we have,

$$\begin{aligned} f(A^B_1, A^B_2, \ldots, A^B_n)(1, (k+1)w) &= \sum_{m \in Y^k} [m]f \cdot m(A^B_{i_1}, A^B_{i_2}, \ldots, A^B_{i_k})(1, (k+1)w) \\ &= \sum_{m \in Y^k} [m]f \cdot [m]g. \end{aligned}$$

Now, we construct a $k \times n$ rectangular matrix $A = (a_{i,j})_{i \in [k], j \in [n]}$ from the given ABP B^{nc} setting $a_{i,j} = A^{B^{nc}}_j$ as defined. Using Observation 8, we now have,

$$\mathrm{rPer}(A)(1, (k+1)s) = S^*_{n,k}(A^{B^{nc}}_1, \ldots, A^{B^{nc}}_n)(1, (k+1)s) = (S^*_{n,k} \circ B^{nc})(\vec{1}) = (S_{n,k} \circ^s B)(\vec{1}).$$

Hence combining the algorithm of Björklund et al. for evaluating rectangular permanent over noncommutative ring [7] with Lemma 9, we can evaluate the sum of the coefficients deterministically in time $O(k\binom{n}{\downarrow k/2} \mathrm{poly}(s, n))$.

Now, we are ready to prove the general case. It uses the following standard transformation from circuit to ABP [20, 19] and reduces the problem to the ABP case again. Given an arithmetic circuit of size s' computing a polynomial f of degree k, f can also be computed by a homogeneous ABP of size $s'^{O(\log k)}$. Hence given a polynomial f by a $\mathrm{poly}(n)$ sized circuit, we first get a circuit of $\mathrm{poly}(n)$ size for degree-k part of f using standard method of homogenization [19]. Then we convert the circuit to a homogeneous ABP of size $n^{O(\log k)}$. Then, we apply the first part of the proof on the newly constructed ABP. Notice that the entire computation can be done in deterministic $O^*(n^{k/2 + c \log k})$ for some constant c. ◀

Some Applications

As immediate consequence of Theorem 1, we improve the counting complexity of several combinatorial problems studied in [14]. To the best of our knowledge, nothing better than the trivial exhaustive search algorithm were known for the counting version of these problems. We start with the k-Tree problem.

▶ **Corollary 10.** *Given a tree T of k nodes and a graph G of n nodes, we can count the number of (not necessarily induced) copies of T in G in deterministic $O^*(\binom{n}{\downarrow k/2})$ time.*

Proof. Let us define $Q = \sum_{j \in V(T), i \in V(G)} C_{T,i,j}$, following [14] where if $|V(T)| = 1$, we define $C_{T,i,j} = x_j$ and if $|V(T)| > 1$, let $T_{i,1}, \ldots, T_{i,\ell}$ be the connected subtrees of T remaining after node$i$ is removed from T. For each $t \in [\ell]$, let $n_{i,t} \in [k]$ be the (unique) node in $T_{i,t}$ that is a neighbour of i in T, then we define

$$C_{T,i,j} = \prod_{t=1}^{\ell} \left(\sum_{j':(j,j') \in E(G)} x_j \cdot C_{T_{i,t},n_{i,t},j'} \right).$$

By the result of [14], it is known that to solve the k-Tree problem it is sufficient to count the number of multilinear terms in Q. Following Theorem 1, it suffices to show that Q has a $\text{poly}(n,k)$ sized ABP. It is enough to show that $C_{T,i,j}$ has a $\text{poly}(n,k)$ sized ABP and the ABP for Q follows easily. We construct an ABP for each $C(T,i,j)$ of size $\text{poly}(n,k)$ by induction on size of T. Suppose $C_{T,i,j}$ has such small ABP for $|V(T)| \leq p$. Then, for $V(T) = p+1$, it is clear from the definition that $C(T,i,j)$ will also have a small ABP. Therefore, the polynomial Q will also have an ABP of size $\text{poly}(n,k)$. ◀

The second application is for t-Dominating Set problem.

▶ **Corollary 11.** *Given a graph $G = (V,E)$, we can count the number of sets S of size k that dominates at least t nodes in G in $O^*(\binom{n}{\downarrow t/2})$ deterministic time.*

Proof. Following [14], define

$$P(X,z) = \left(\sum_{i \in V} \left((1+zx_i) \prod_{j:(i,j) \in E} (1+zx_j) \right) \right)^k.$$

We inspect $[z^t]P(X,z)$ which is a homogeneous degree t polynomial over X, call it $Q(X)$. As $P(X,z)$ has a small ABP of $\text{poly}(n,k)$ size substituing z by any scalar, we obtain an ABP of size $\text{poly}(n,k)$ for $Q(X)$ also by interpolation. Then, we use the standard method to homogenize the ABP and apply Theorem 1 to count the number of multilinear terms. This is sufficient to solve the problem by the result of [14]. ◀

The final application is regarding m-Dimensional k-Matching problem.

▶ **Corollary 12.** *Given mutually disjoint sets U_i, $i \in [m]$, and a collection C of m-tuples from $U_1 \times \cdots \times U_m$, we can count the number of sub-collection of k mutually disjoint m-tuples in C in deterministic $O^*(\binom{n}{\downarrow (m-1)k/2})$ time.*

Proof. Following [14] , encode each element u in $U = \cup_{i=2}^{m} U_i$ by a variable $x_u \in X$. Encode each m-tuple $t = (u_1, \ldots, u_m) \in C \subseteq U_1 \times \cdots \times U_m$ by the monomial $M_t = \prod_{i=2}^{m} x_{u_i}$. Assume $U_1 = \{u_{1,1}, \ldots, u_{1,n}\}$, and let $T_j \subseteq C$ denote the subset of m-tuples whose first coordinate is $u_{1,j}$. Consider the polynomial

$$P(X,z) = \prod_{j=1}^{n} \left(1 + \sum_{t \in T_j} (z \cdot M_t) \right).$$

Clearly, $P(X,z)$ has an ABP of size $\text{poly}(n,m)$. Let $Q(X) = [z^k]P(X,z)$, we can obtain a small ABP of size $\text{poly}(n,m,k)$ for $Q(X)$ by interpolation. Now, we homogenize the ABP and apply Theorem 1 to count the number of multilinear terms which is sufficient by the result of [14]. ◀

Hardness for Computing Rectangular Permanent over *any* Ring

In [7], it is shown that a $k \times n$ rectangular permanent can be evaluated over commutative rings and commutative semirings in $O(h(k) \cdot \text{poly}(n,k))$ time for some computable function h . In other words, the problem is in FPT parameterized by the number of rows. An interesting question is to ask whether one can get any FPT algorithm when the entries are from noncommutative rings (in particular, matrix rings). We prove that such an algorithm is unlikely to exist. We show that counting the number of k-paths in a graph G, a well-known #W[1]-complete problem, reduces to this problem. So, unless ETH fails we do not have such an algorithm [10].

▶ **Theorem 13.** *Given a $k \times n$ matrix X with entries $X_{ij} \in \mathbb{M}_{t\times t}(\mathbb{Q})$, computing the rectangular permanent of X is #W[1]-hard with k as the parameter where $t = (k+1)n$ under polynomial time many-one reductions.*

Proof. If we have an algorithm to compute the permanent of a $k \times n$ matrix over noncommutative rings which is FPT in parameter k, that yields an algorithm which is FPT in k for evaluating the polynomial $S^*_{n,k}$ on matrix inputs. This follows from Observation 8. Now, given a graph G we can compute a homogeneous ABP of width n and k layers for the graph polynomial C_G defined as follows. Let $G(V,E)$ be a directed graph with n vertices where $V(G) = \{v_1, v_2, \ldots, v_n\}$. A k-walk is a sequence of k vertices $v_{i_1}, v_{i_2}, \ldots, v_{i_k}$ where $(v_{i_j}, v_{i_{j+1}}) \in E$ for each $1 \le j \le k-1$. A k-path is a k-walk where no vertex is repeated. Let A be the adjacency matrix of G, and let $y_1, y_2, \ldots, y_n$ be noncommuting variables. Define an $n \times n$ matrix B

$$B[i,j] = A[i,j] \cdot y_i, \quad 1 \le i,j \le n.$$

Let $\vec{1}$ denote the all 1's vector of length n. Let $\vec{y}$ be the length n vector defined by $\vec{y}[i] = y_i$. The *graph polynomial* $C_G \in \mathbb{F}\langle Y\rangle$ is defined as

$$C_G(Y) = \vec{1}^T \cdot B^{k-1} \cdot \vec{y}.$$

Let W be the set of all k-walks in G. The following observation is folklore.

▶ **Observation 14.**

$$C_G(Y) = \sum_{v_{i_1} v_{i_2} \ldots v_{i_k} \in W} y_{i_1} y_{i_2} \cdots y_{i_k}.$$

Hence, G contains a k-path if and only if the graph polynomial C_G contains a multilinear term.

Clearly the number of k-paths in G is equal to $(C_G \circ S_{n,k})(\vec{1})$. By Lemma 7, we know that it suffices to compute $({C_G}^{nc} \circ S^*_{n,k})(\vec{1})$. We construct $kn \times kn$ matrices $A_1, \ldots, A_n$ from the ABP of ${C_G}^{nc}$ following Lemma 9. Then from Lemma 9, we know that $({C_G}^{nc} \circ S^*_{n,k})(\vec{1}) = S^*_{n,k}(A_1, \ldots, A_n)(1,t)$ where $t = (k+1)n$. So if we have an algorithm which is FPT in k for evaluating $S^*_{n,k}$ over matrix inputs, we also get an algorithm to count the number of k-paths in G in FPT(k) time. ◀

4 Multilinear Monomial Detection

In this section, we prove Theorem 2. Apart from being a new technique, the Hadamard product based algorithm runs in polynomial space and does not depend on the characteristic of the ground field. This is in contrast with the exterior algebra based approach [8] and Waring rank based approach [16].

We first recall that the Hadamard product of a noncommutative circuit and a noncommutative ABP can be computed efficiently. The proof is similar to the proof of Lemma 9.

▶ **Lemma 15** (Corollary 4 of [5]). *Given a homogeneous noncommutative circuit of size S' for $f \in \mathbb{F}\langle y_1, y_2, \ldots, y_n\rangle$ and a homogeneous noncommutative ABP of size S for $g \in \mathbb{F}\langle y_1, y_2, \ldots, y_n\rangle$, we can compute a noncommutative circuit of size $O(S^3S')$ for $f \circ g$ in deterministic $S^3S' \cdot \mathrm{poly}(n,k)$ time where k is the degree upper bound for f and g.*

Now we give an algorithm for computing the Hadamard product for a special case in the commutative setting. Any depth two $\Pi^{[k]}\Sigma$ circuit computes the product of k homogeneous linear forms over the input set of variables X.

▶ **Lemma 16.** *Given an arithmetic circuit C of size s computing $g \in \mathbb{F}[X]$, and a homogeneous $\Pi^{[k]}\Sigma$ circuit computing $f \in \mathbb{F}[X]$, and any point $\vec{a} \in \mathbb{F}^n$, we can evaluate $(f \circ^s g)(\vec{a})$ in $O^*(2^k)$ time and in polynomial space.*

Proof. By standard homogenization technique [19] we can extract the homogeneous degree-k component of C and thus we can assume that C computes a homogeneous degree-k polynomial. Write $f = \prod_{j=1}^{k} L_j$, for homogeneous linear forms L_j. The corresponding noncommutative polynomial $\hat{f}$ is defined by the natural order of the j indices (and replacing x_i by y_i for each i).

▷ Claim 17. The noncommutative polynomial f^* has a (noncommutative) $\Sigma^{[2^k]}\Pi^{[k]}\Sigma$ circuit, which we can write as $f^* = \sum_{i=1}^{2^k} C_i$, where each C_i is a (noncommutative) $\Pi^{[k]}\Sigma$ circuit.

Before we prove the claim, we show that it easily yields the desired algorithm: First we notice that

$$C^{nc} \circ f^* = \sum_{i=1}^{2^k} C^{nc} \circ C_i.$$

Now, by Lemma 15, we can compute in $\mathrm{poly}(n,s,k)$ time a $\mathrm{poly}(n,s,k)$ size circuit for the (noncommutative) Hadamard product $C^{nc} \circ C_i$. As argued in the proof of Lemma 7, for any $\vec{a} \in \mathbb{F}^n$ we have

$$(g \circ^s f)(\vec{a}) = (C \circ^s f)(\vec{a}) = (C^{nc} \circ f^*)(\vec{a}).$$

Thus, we can evaluate $(g \circ^s f)(\vec{a})$ by incrementally computing $(C^{nc} \circ C_i)(\vec{a})$ and adding up for $1 \le i \le 2^k$. This can be clearly implemented using only polynomial space. ◀

Now, we prove the above claim. Consider $f = L_1 L_2 \cdots L_k$. Then $\hat{f} = \hat{L}_1 \hat{L}_2 \cdots \hat{L}_k$, where $\hat{L}_j$ is obtained from L_j by replacing variables x_i with the noncommutative variable y_i for each i. We will require the following observation.

▶ **Observation 18.**

$$f^* = \sum_{\sigma \in S_k} \hat{L}_{\sigma(1)} \hat{L}_{\sigma(2)} \cdots \hat{L}_{\sigma(k)}.$$

Proof. Let us prove the claim, monomial by monomial. Fix a monomial m' in f^* such that $m' \to m$. Suppose $m' = y_{i_1} y_{i_2} \ldots y_{i_k}$. Note that, $m = x_{i_1} x_{i_2} \ldots x_{i_k}$. Recall from Proposition 6, $[m']f^* = m! \cdot [m]f$. Now, the coefficient of m' in $\sum_{\sigma \in S_k} \prod_{j=1}^{k} \hat{L}_{\sigma(j)}$ is

$$[m'] \left(\sum_{\sigma \in S_k} \prod_{j=1}^{k} \hat{L}_{\sigma(j)} \right) = \sum_{\sigma \in S_k} \prod_{j=1}^{k} [y_{i_j}] \hat{L}_{\sigma(j)}.$$

Let us notice that, $[y_{i_j}]\hat{L}_{\sigma(j)} = [x_{i_j}]L_{\sigma(j)}$. Hence,

$$[m']\left(\sum_{\sigma\in S_k}\prod_{j=1}^{k}\hat{L}_{\sigma(j)}\right) = \sum_{\sigma\in S_k}\prod_{j=1}^{k}[x_{i_j}]L_{\sigma(j)}. \qquad \blacktriangleleft$$

Now we observe the following easy fact.

▶ **Observation 19.** *For a degree k monomial $m = x_{i_1}x_{i_2}\cdots x_{i_k}$ (where the variables can have repeated occurrences) and a homogeneous $\Pi^{[k]}\Sigma$ circuit $C = \prod_{j=1}^{k} L_j$, the coefficient of monomial m in C is given by $m!\cdot[m]C = \sum_{\sigma\in S_k}\prod_{j=1}^{k}([x_{i_j}]L_{\sigma(j)})$.*

Proof of Claim 17. Now, the claim directly follows from Observation 19 as $\sum_{\sigma\in S_k}\prod_{j=1}^{k}[x_{i_j}]L_{\sigma(j)} = m!\cdot[m]f$.

Now define the $k\times k$ matrix T such that each row of T_i is just the linear forms $\hat{L}_1, \hat{L}_2, \ldots, \hat{L}_k$ appearing in f. The (noncommutative) permanent of T is given by $\mathrm{Perm}(T) = \sum_{\sigma\in S_k}\prod_{j=1}^{k}\hat{L}_{\sigma(j)}$, which is just f^*.

We now apply Ryser's formula [17] (noting the fact that it holds for the noncommutative permanent too), to express $\mathrm{Perm}(T)$ as a depth-3 homogeneous noncommutative $\Sigma^{[2^k]}\Pi^{[k]}\Sigma$ formula. It follows that $f^* = \mathrm{Perm}(T)$ has a $\Sigma^{[2^k]}\Pi^{[k]}\Sigma$ noncommutative formula. ◁

We include a proof of Observation 19 for completeness.

Proof. We assume, without loss of generality, that repeated variables are adjacent in the monomial $m = x_{i_1}x_{i_2}\cdots x_{i_k}$. More precisely, suppose the first e_1 variables are x_{j_1}, and the next e_2 variables are x_{j_2} and so on until the last e_q variables are x_{j_q}, where the q variables $x_{j_k}, 1\le k\le q$ are all distinct.

We notice that the monomial m can be generated in C by first fixing an order $\sigma : [k]\mapsto[k]$ for multiplying the k linear forms as $L_{\sigma(1)}L_{\sigma(2)}\cdots L_{\sigma(k)}$, and then multiplying the coefficients of variable $x_{i_j}, 1\le j\le k$ picked successively from linear forms $L_{\sigma(j)}, 1\le j\le k$. However, these $k!$ orderings repeatedly count terms.

We claim that each distinct product of coefficients term is counted exactly $m!$ times. Let $E_j\subseteq[k]$ denote the interval $E_j = \{\ell \mid e_{j-1}+1\le\ell\le e_j\}, 1\le j\le q$, where we set $e_0 = 0$.

Now, to see the claim we only need to note that two permutations $\sigma,\tau\in S_k$ give rise to the same product of coefficients term if and only if $\sigma(E_j) = \tau(E_j), 1\le j\le q$. Thus, the number of permutations τ that generate the same term as σ is $m!$.

Therefore the sum of product of coefficients $\sum_{\sigma\in S_k}\prod_{j=1}^{k}([x_{i_j}]L_{\sigma(j)})$ is same as $m!\cdot[m]C$, which completes the proof. ◀

▶ Remark 20. Over rationals, computing $f\circ^s g$, when g is a $\Pi^{[k]}\Sigma$ circuit, can also be done by computing g^* using Fischer's identity [11]. However, Lemma 16 also works over finite fields.

Now we are ready to prove Theorem 2.

Proof. By homogenization, we can assume that C computes a homogeneous degree k polynomial f.

We go over a collection of colorings $\{\zeta_i : [n]\to[k]\}$ chosen uniformly at random and define a $\Pi^{[k]}\Sigma$ formula $P_i = \prod_{j=1}^{k}\sum_{\ell:\zeta_i(\ell)=j} x_\ell$ for each colouring ζ_i. A monomial is *covered* by a coloring ζ_i if the monomial is nonzero in P_i. The probability that a random coloring covers a given degree-k multilinear monomial is $\frac{k!}{k^k}\approx e^{-k}$. Hence, for a collection of $O^*(e^k)$ many colorings $\{\zeta_i : [n]\to[k]\}$ chosen uniformly at random, with constant probability all the multilinear terms of degree k are covered.

For each coloring ζ_i, we construct a circuit $C'_i = C \circ^s P_i$.

Notice that we are interested only in multilinear monomials and for each such monomial m, the additional multiplicative factor $m! = 1$. Also, the coefficient of each monomial is exactly 1 in each P_i, and if f contains a multilinear term then it will be covered by *some* P_i. Now we perform PIT test on each C'_i using Demillo-Lipton-Schwartz-Zippel Lemma [9, 24, 18] in randomized polynomial time to complete the procedure. More precisely, we pick a random $\vec{a} \in \mathbb{F}^n$ and evaluate C'_i on that point. Notice that, by the proof of Lemma 16, it is easy to see that $C'_i(\vec{a})$ can be computed deterministically in time $2^k \cdot \text{poly}(n, s)$ time and $\text{poly}(n, k)$ space.[3]

To improve the run time from $O^*((2e)^k)$ to $O^*(4.32^k)$, we can use the idea of Hüffner et al. [12][4]. The key idea is that, using more than k colors we would reduce the number of colorings and hence the number of $\Pi\Sigma$ circuits, but it would increase the formal degree of each P_i. Following [12], we use $1.3k$ many colors and each P_i will be a $\Pi^{[1.3k]}\Sigma$ circuit. For each coloring $\zeta_i : [n] \to [1.3k]$ chosen uniformly at random, we define the following $\Pi^{[1.3k]}\Sigma$ circuit, $P_i(x_1, x_2, \ldots, x_n, z_1, \ldots, z_{1.3k}) = \prod_{j=1}^{1.3k} \left(\sum_{\ell:\zeta_i(\ell)=j} x_\ell + z_j \right)$.

Since each P_i is of degree $1.3k$, we need to modify the circuit C to another circuit C' of degree $1.3k$ in order to apply Hadamard products. The key idea is to define the circuits $C' \in \mathbb{F}[X, Z]$ as follows:

$$C'(X, Z) = C(X) \cdot S_{1.3k,0.3k}(z_1, \ldots, z_{1.3k})$$

where $S_{1.3k,0.3k}(z_1, \ldots, z_{1.3k})$ is the elementary symmetric polynomial of degree $0.3k$ over the variables $z_1, \ldots, z_{1.3k}$. By the result of [12], for $O^*(1.752^k)$ many random colorings with high probability each multilinear monomial in C will be covered by the monomials of some P_i (over the X variables).

Now to compute $C'^{nc} \circ P_i^*$ for each i, we symmetrize the polynomial P_i, the symmetrization happens over the X variables as well as over the Z variables. But in C'^{nc} we are only interested in the monomials (or words) where the rightmost $0.3k$ variables are over Z variables. In the noncommutative circuit C'^{nc}, every sub-word $z_{i_1} z_{i_2} \ldots z_{i_{0.3k}}$ receives a natural ordering $i_1 < i_2 < \ldots < i_{0.3k}$.

Notice that

$$P_i^*(X, Z) = \sum_{\sigma \in S_{1.3k}} \prod_{j=1}^{1.3k} \left(\sum_{\ell:\zeta_i(\ell)=\sigma(j)} x_\ell + z_{\sigma(j)} \right).$$

Our goal is to understand the part of $P_i^*(X, Z)$ where each monomial ends with the sub-word $z_{i_1} z_{i_2} \ldots z_{i_{0.3k}}$ and the top k symbols are over the X variables. For a fixed set of indices $W = \{i_1 < i_2 < \ldots < i_{0.3k}\}$, define the set $T = [1.3k] \setminus W$. Let $S_{[k],T}$ be the set of permutations $\sigma \in S_{1.3k}$ such that $\sigma : [k] \to T$ and $\sigma(k + j) = i_j$ for $1 \leq j \leq 0.3k$. As we have fixed the last $0.3k$ positions, each $\sigma \in S_{[k],T}$ corresponds to some $\sigma' \in S_k$. Let $Z_W = z_{i_1} z_{i_2} \ldots z_{i_{0.3k}}$. Now the following claim is immediate.

[3] Since the syntactic degree of the circuit is not bounded here, and if we have to account for the bit level complexity (over $\mathbb{Z}$) of the scalars generated in the intermediate stage we may get field elements whose bit level complexity is exponential in the input size. So, a standard technique is to take a random prime of polynomial bit-size and evaluate the circuit modulo that prime.

[4] This is also used in [8].

▷ Claim 21. The part of $P_i^*(X,Z)$ where each monomial ends with the sub-word Z_W and the first k variables are from X, is $P^*_{i,W} \cdot Z_W$, where

$$P^*_{i,W}(X) = \sum_{\sigma \in S_{[k],T}} \prod_{j=1}^{k} \left(\sum_{\ell:\zeta_i(\ell)=\sigma(j)} x_\ell \right) = \sum_{\sigma' \in S_k} \prod_{j=1}^{k} \left(\sum_{\ell:\zeta_i(\ell)=\sigma'(j)} x_\ell \right).$$

Notice that, $\sum_{W\subseteq[1.3k]:|W|=0.3k} P^*_{i,W}$ contains all the *colourful* degree-k multilinear monomials over X. We now obtain the following.

$$(C'^{nc} \circ P_i^*)(X,Z) = \sum_{W\subseteq[1.3k]:|W|=0.3k} \left(C^{nc}(X) \circ P^*_{i,W}(X)\right) \cdot Z_W.$$

Setting each $z_i = 1$ and using distributivity of Hadamard product, we get $(C'^{nc} \circ P_i^*)(X, \vec{1}) = C^{nc}(X) \circ \sum_{W\subseteq[1.3k]:|W|=0.3k} P^*_{i,W}$ which is the *colourful* multilinear part of the input circuit.

We now consider $(C' \circ^s P_i)(X,Z)$ and substitute 1 for each Z variable and do a randomized PIT test on the X variables using Demillo-Lipton-Schwartz-Zippel Lemma [9, 24, 18]. By Lemma 16, for any random $\vec{a} \in \mathbb{F}^n$, $(C' \circ^s P_i)(\vec{a})$ can be computed in $O^*(2^{1.3k}) = O^*(2.46^k)$ time and poly(n,k) space. This suffices to check whether the resulting circuit is identically zero or not. We repeat the procedure for each coloring and obtain a randomized $O^*(4.32^k)$ algorithm. This completes the proof of Theorem 2. ◀

References

1. Noga Alon, Raphael Yuster, and Uri Zwick. Color-Coding. *J. ACM*, 42(4):844–856, 1995. doi:10.1145/210332.210337.
2. Vikraman Arvind, Abhranil Chatterjee, Rajit Datta, and Partha Mukhopadhyay. Fast Exact Algorithms Using Hadamard Product of Polynomials. *CoRR*, abs/1807.04496, 2018. arXiv:1807.04496.
3. Vikraman Arvind, Pushkar S. Joglekar, and Srikanth Srinivasan. Arithmetic Circuits and the Hadamard Product of Polynomials. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2009, December 15-17, 2009, IIT Kanpur, India*, pages 25–36, 2009. doi:10.4230/LIPIcs.FSTTCS.2009.2304.
4. Vikraman Arvind and Srikanth Srinivasan. On the hardness of the noncommutative determinant. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 677–686, 2010. doi:10.1145/1806689.1806782.
5. Vikraman Arvind and Srikanth Srinivasan. On the hardness of the noncommutative determinant. *Computational Complexity*, 27(1):1–29, 2018. doi:10.1007/s00037-016-0148-5.
6. Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Counting Paths and Packings in Halves. In Amos Fiat and Peter Sanders, editors, *Algorithms - ESA 2009*, pages 578–586, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
7. Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Evaluation of permanents in rings and semirings. *Inf. Process. Lett.*, 110(20):867–870, 2010. doi:10.1016/j.ipl.2010.07.005.
8. Cornelius Brand, Holger Dell, and Thore Husfeldt. Extensor-coding. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 151–164, 2018. doi:10.1145/3188745.3188902.
9. Richard A. Demillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193–195, 1978. doi:10.1016/0020-0190(78)90067-4.
10. Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

11 Ismor Fischer. Sums of Like Powers of Multivariate Linear Forms. *Mathematics Magazine*, 67(1):59–61, 1994. doi:10.1080/0025570X.1994.11996185.

12 Falk Hüffner, Sebastian Wernicke, and Thomas Zichner. Algorithm Engineering for Color-Coding with Applications to Signaling Pathway Detection. *Algorithmica*, 52(2):114–132, 2008. doi:10.1007/s00453-007-9008-7.

13 Ioannis Koutis. Faster Algebraic Algorithms for Path and Packing Problems. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games*, pages 575–586, 2008. doi:10.1007/978-3-540-70575-8_47.

14 Ioannis Koutis and Ryan Williams. LIMITS and applications of group algebras for parameterized problems. *ACM Trans. Algorithms*, 12(3):31:1–31:18, 2016. doi:10.1145/2885499.

15 Hwangrae Lee. Power Sum Decompositions of Elementary Symmetric Polynomials. In *Linear Algebra and its Applications*, volume 492. Elsevier, August 2015.

16 Kevin Pratt. Faster Algorithms via Waring Decompositions. *CoRR*, abs/1807.06194, 2018. arXiv:1807.06194.

17 H.J. Ryser. *Combinatorial mathematics*. Carus mathematical monographs. Mathematical Association of America; distributed by Wiley [New York, 1963. URL: https://books.google.co.in/books?id=wOruAAAAMAAJ.

18 Jacob T. Schwartz. Fast Probabilistic algorithm for verification of polynomial identities. *J. ACM.*, 27(4):701–717, 1980.

19 Amir Shpilka and Amir Yehudayoff. Arithmetic Circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010. doi:10.1561/0400000039.

20 Leslie G. Valiant, Sven Skyum, S. Berkowitz, and Charles Rackoff. Fast Parallel Computation of Polynomials Using Few Processors. *SIAM J. Comput.*, 12(4):641–644, 1983. doi:10.1137/0212043.

21 Richard Ryan Williams. The Polynomial Method in Circuit Complexity Applied to Algorithm Design (Invited Talk). In *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India*, pages 47–60, 2014. doi:10.4230/LIPIcs.FSTTCS.2014.47.

22 Ryan Williams. Finding paths of length k in $O^*(2^k)$ time. *Inf. Process. Lett.*, 109(6):315–318, 2009. doi:10.1016/j.ipl.2008.11.004.

23 Ryan Williams. Algorithms for Circuits and Circuits for Algorithms. In *IEEE 29th Conference on Computational Complexity, CCC 2014, Vancouver, BC, Canada, June 11-13, 2014*, pages 248–261, 2014. doi:10.1109/CCC.2014.33.

24 R. Zippel. Probabilistic algorithms for sparse polynomials. In *Proc. of the Int. Sym. on Symbolic and Algebraic Computation*, pages 216–226, 1979.