

Aalto University  
School of Science  
Bachelor's Programme in Science and Technology

# **Detecting multilinear monomials with algebraic fingerprints**

**Bachelor's Thesis**

**May 1, 2023**

**Onni Miettinen**

AM: prefer  
 $\cdot$ , \*  
usually  
indicates  
some other  
operation  
AM: use poly  
from  
complexity  
package  
AM: First  
sentence of  
abstract uses  
a bit too  
much jargon  
AM: two  
"however" in  
third  
paragraph

<b>Author:</b>	Onni Miettinen
<b>Title of thesis:</b>	Detecting multilinear monomials with algebraic fingerprints
<b>Date:</b>	May 1, 2023
<b>Pages:</b>	30
<b>Major:</b>	Computer Science
<b>Code:</b>	SCI3027
<b>Supervisor:</b>	Prof. Eero Hyvönen
<b>Instructor:</b>	D.Sc. Augusto Modanese (Department of Computer Science)
<p>This thesis studies how parameterized combinatorial problems can be solved by detecting a <math>k</math>-multilinear monomial in a multivariate polynomial by algebraic means. More specifically, the thesis focuses on the technique of algebraic fingerprinting by Koutis and Williams. Moreover, the thesis overviews the technique as a general framework for parameterized combinatorial problems, and discusses its limits. Finally, the thesis collects some ideas for improving the framework or even surpassing the lower limit of the general framework.</p> <p>With algebraic fingerprints, a <math>k</math>-multilinear monomial can be detected in time <math>\mathcal{O}(2^k * \text{poly}(n))</math>, where <math>n</math> is the number of variables in the multivariate polynomial. This gives a general framework for solving parameterized combinatorial problems efficiently, since they can in general be reduced to instances of <math>k</math>-multilinear monomial detection problems. Especially algorithms that rely on color coding can be accelerated in time by a factor of <math>e^k</math> with the use of algebraic fingerprints. Currently, this technique underlies fastest algorithms for many parameterized combinatorial problems.</p> <p>However, <math>\Omega(2^k * \text{poly}(n))</math> is the lower limit of the technique: multilinear monomials cannot be detected faster with the general technique of algebraic fingerprints by Koutis and Williams. However, the algebraic framework solves the very general multilinear monomial detection problem; if the underlying combinatorics of the problem are well understood and taken use of in the reduction to multilinear monomial detection, a much faster algorithm based on algebraic fingerprinting may be found for a specific problem. Using ideas from algebraic fingerprinting, Björklund managed to find an algorithm that finds a Hamiltonian path in time <math>\mathcal{O}(1.657^n * \text{poly}(n))</math>.</p>	
<b>Keywords:</b>	multilinear, monomial, detection, algebraic, fingerprints, fingerprinting, parameterized, combinatorial
<b>Language:</b>	English

<b>Tekijä:</b>	Onni Miettinen
<b>Työn nimi:</b>	Detecting multilinear monomials with algebraic fingerprints
<b>Päiväys:</b>	1. toukokuuta 2023
<b>Sivumäärä:</b>	30
<b>Pääaine:</b>	Tietotekniikka
<b>Koodi:</b>	SCI3027
<b>Vastuopettaja:</b>	Prof. Eero Hyvönen
<b>Työn ohjaaja(t):</b>	D.Sc. Augusto Modanese (Tietotekniikan laitos)
<p>Tässä työssä tutkitaan, kuinka parametrisoituja kombinatorisia ongelmia voidaan ratkaista löytämällä <math>k</math>-asteinen multilineaarinen monomi monimuuttujaisesta polynomista algebrallisin keinoin. Kandidaatintyö keskittyy tarkemmin Williamsin ja Koutisin algebrallisia sormenjälkiä hyödyntävään tekniikkaan, jolla voidaan havaita multilineaarinen monomi tehokkaasti. Lisäksi tutkitaan tämän tekniikan suomaa yleistä kehystä parametrisoitujen ongelmien ratkaisemiseksi ja sen mahdollisia rajoja. Viimeiseksi pohditaan keinoja, joilla kehystä voitaisiin kehittää tai jopa ohittaa kehyksen antama aliraja algoritmien tehokkuudelle.</p> <p>Williamsin ja Koutisin algebrallisten sormenjälkien avulla voidaan havaita <math>k</math>-asteen multilineaarinen monomi ajassa <math>\mathcal{O}(2^k * \text{poly}(n))</math>, jossa <math>n</math> on polynomin muuttujien lukumäärä. Tämä johtaa yleiseen parametrisoitujen ongelmien kehykseen, jolla niitä voidaan ratkaista tehokkaasti. Etenkin color-coding—metodiin nojaavia algoritmeja voidaan nopeuttaa tekijällä <math>e^k</math> hyödyntämällä algebrallisia sormenjälkiä. Tällä hetkellä algebrallisten sormenjälkien tekniikka pohjaa nopeimpia algoritmeja monille parametrisoiduille kombinatorisille ongelmille.</p> <p><math>\Omega(2^k * \text{poly}(n))</math> on kuitenkin kehyksen alaraja: Williamsin ja Koutisin algebrallisten sormenjälkien avulla ei voida havaita multilineaarisia monomeja nopeammin. Toisaalta heidän kehys perustuu yleiseen multilineaaristen monomien havaitsemisongelmaan; jos hyödynnetään ongelmakohtaisia kombinatorisia ominaisuuksia ongelman redusoinnissa multilineaaristen monomien havaitsemiseksi, voidaan löytää algoritmi, joka suoriutuu nopeammin kyseisellä ongelmalla. Björklund kykeni kehittämään algebrallisten sormenjälkien avulla algoritmin, joka löytää Hamiltonisen polun ajassa <math>\mathcal{O}(1.657^n * \text{poly}(n))</math>.</p>	
<b>Avainsanat:</b>	multilineaarinen, monomi, havaitseminen, algebrallinen, sormenjälki, parametrisoitu, kombinatorinen
<b>Kieli:</b>	Englanti

# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
1.1	Research scope and thesis structure . . . . .	5
1.2	Algebrization of combinatorial problems . . . . .	6
1.3	Detecting a multilinear monomial . . . . .	8
1.4	Related problems . . . . .	9
<b>2</b>	<b>Preliminaries</b>	<b>10</b>
2.1	Algebra . . . . .	10
2.2	Arithmetic circuits . . . . .	13
2.3	Notation and other terminology . . . . .	13
<b>3</b>	<b>General multilinear monomial detection</b>	<b>15</b>
3.1	Algebraic fingerprinting . . . . .	15
3.1.1	Specifications for the algebraic structure . . . . .	16
3.1.2	Using group algebras of $\mathbb{Z}_2^k$ . . . . .	17
3.1.3	Fingerprints to prevent unwanted cancelation . . . . .	18
3.1.4	Polynomial identity testing . . . . .	19
3.2	Time and space complexity . . . . .	21
<b>4</b>	<b>Limits of algebraic fingerprinting</b>	<b>22</b>
4.1	Algebraic optimization . . . . .	22
4.2	Derandomization . . . . .	22
4.3	Finding the solution . . . . .	23
<b>5</b>	<b>Improving from algebraic fingerprinting</b>	<b>23</b>
5.1	Fingerprinting for cancelation of non-solutions . . . . .	24
5.2	Other research in detecting multilinear monomials . . . . .	25
<b>6</b>	<b>Conclusion</b>	<b>26</b>
	<b>References</b>	<b>27</b>

# 1 Introduction

In the past fifteen years, there have been rapid advances in the algorithms for combinatorial problems. This has been greatly sparked by the development in algebraic techniques for solving the multilinear monomial detection problem, that is, deciding whether a multivariate (multiple variables) polynomial contains a multilinear monomial<sup>1</sup>. Multilinear monomial detection for parameterized combinatorial problems was first introduced by Koutis in an early work [Kou05], where a set packing problem is reduced to multilinear monomial detection.

AM: (Namely,) the technique of *algebraic fingerprinting*, introduced by Koutis [Kou08] and further developed by Williams [Wil09], has found great success for many combinatorial problems. For instance with algebraic fingerprinting, the  $k$ -path problem that previously could be solved in  $\mathcal{O}^*(4^k)^2$  time by Chen et al. [Che+07], can now be solved in  $\mathcal{O}^*(2^{3k/2})$  time [Kou08]. This result was quickly improved by Williams [Wil09], who gave an  $\mathcal{O}^*(2^k)$  algorithm.

Most famously, Björklund [Bjö14] published an algorithm that solved the famous Hamiltonian path problem (Hamiltonicity) in  $\mathcal{O}^*(1.657^n)$  time with clever utilizations of this novel algebraic technique. The fastest algorithms for Hamiltonicity before this ran in  $\mathcal{O}^*(2^n)$  time, and were known since 1962 [HK62], [Bel62]. This was a significant improvement on a problem that had seen no progress in nearly fifty years! Soon enough for  $k$ -path, an  $\mathcal{O}^*(1.657^k)$  algorithm was found [Bjö+17]. AM: and date from 1962 [HK62], [Bel62]. AM: thereafter, a  $\mathcal{O}^*(1.657^k)$  algorithm was found for the related (?)  $k$ -path problem [Bjö+17].

## 1.1 Research scope and thesis structure

The goals of this bachelor's thesis are to find out how multilinear monomial detection is relevant in combinatorial problems, and how algebraic fingerprints can be utilized to design faster algorithms for these problems. For the reader with background in computer science, this thesis overviews a clever use of algebra in theoretical computer science, that is the technique of algebraic fingerprinting.

In the next subsections AM: rest of this introduction, we discuss the relation of combinatorics and polynomials, discuss the algebrization of combinatorial problems, and define the multilinear monomial detection problem as a prelude for the thesis.

In Section 1.4, we overview the area of problems where this technique proves useful. Section 2 gives necessary preliminaries for the reader. In Section 3, we dive into the technique of algebraic fingerprints, and solve the multilinear monomial detection problem

---

<sup>1</sup>i.e., a term of multiple variables where each variable appears at most once. See also Section 2.1

<sup>2</sup> $\mathcal{O}^*$  may hide factors polynomial to input size. See Section 2.3

AM: I do see this style of footnotes from time to time, but some people are not fond of it. Parentheses might be a better choice. AM: Move footnote elsewhere

AM: used twice

with algebraic means. Section 5 briefly overviews ideas for improvement from the general algebraic technique. Section 6 concludes the thesis.

## 1.2 Algebrization of combinatorial problems

A combinatorial problem asks whether a given finite set of objects satisfies some given constraints. For example, the  $k$ -path problem asks for, given a finite set of vertices and edges, a simple<sup>3</sup> path of  $k$  vertices. The solutions and non-solutions (solution candidate space) to combinatorial problems can be thought of as combinations of the given objects. The solution candidate space for the  $k$ -path problem consists of combinations of  $k - 1$  edges, where a valid solution combination contains only disjoint edges.

Valiant [Val92] observed that multivariate polynomials have natural combinatorial interpretations. Now, we see how combinatorics may be represented with multivariate polynomials. We may think of a polynomial in its sum of monomials form<sup>4</sup> as a set of combinations. For instance, imagine a set of elements  $M = \{a, b, c\}$ . A AM: The polynomial  $C = a + b + c$  represents all the possible choices if we are to pick an element from  $M$ . Following the standard idea behind polynomial multiplication, we see that a polynomial

$$C^2 = (a + b + c)(a + b + c) = aa + ab + ac + ba + bb + bc + ca + cb + cc$$

represents all the possible combinations if we are to pick twice from  $M$ . Indeed, a polynomial  $C^k$  represents all combinations of  $k$  elements from  $M$ .

Utilizing this idea, Koutis [Kou05] managed to reduce a combinatorial problem into an algebraic one, that is, multilinear monomial detection. In multilinear monomial detection, we represent the combinatorial problem as a polynomial where the solution candidates are encoded as follows: the non-solutions are non-multilinear and solutions are multilinear terms.

AM: You already discuss it in Section 1.3, but you should also mention somewhere around here that complexity of problem is sensitive to how the polynomial is presented (e.g., in sum of monomials form it is trivial) and that the presentation is going to be very succinct (namely algebraic circuit). 1 sentence is enough.

For an instance of  $k$ -path problem, we may use the set of edges as  $M$  where an edge is a product of two indeterminates defined uniquely for each vertex. If we look for  $k - 1$  combinations of elements from  $M$ , we see that a multilinear term in the resulting polynomial  $C^{k-1}$  corresponds to a solution to the  $k$ -path problem: as the edges are disjoint, each indeterminate appears only once.

---

<sup>3</sup>i.e., a path where each vertex is visited only once

<sup>4</sup>an expression where there is no multiplication after addition of monomials. See also Section 2.3

Thus, the task of finding a satisfying combination to the combinatorial problem has been reduced to finding a multilinear monomial from the multivariate polynomial. It follows that a decision problem is answered by the existence of a multilinear monomial, and a counting problem by the number of multilinear monomials. This thesis, however, focuses on the decision problem <sup>5</sup>. AM: This thesis focuses on the former (although we give a brief overview of actually finding a solution in Section 4.3)

Appropriate definitions for the indeterminates for  $M$  are problem-specific. In what follows, this thesis gives another example reduction to multilinear monomial detection: we algebrize the  $k$ -3D matching problem.

The  $k$ -3D matching problem is defined as follows:

$k$ -3D MATCHING

**Input:** Three disjoint sets  $A$ ,  $B$  and  $C$ , and a set of triples  $T \subseteq A \times B \times C$ .

**Question:** Is there a subset  $M \subseteq T$ , such that  $|M| = k$  and the all the triples  $m \in M$  are disjoint?

We begin by defining new indeterminates corresponding to the elements in  $A$ ,  $B$  and  $C$ , labeled as  $a_i$ ,  $b_j$  and  $c_k$ , respectively, where  $i \in [|A|]$ ,  $j \in [|B|]$  and  $k \in [|C|]$  such that  $[n] = \{1, \dots, n\}$ .

For every triple  $t \in T$ , we define a multilinear monomial  $x$  that is a product of the elements in  $t$ . We denote the set of those monomials as  $X$ . Thus,  $(a, b, c) \in T \implies abc \in X$ . From an algebraic perspective, we form monomials in the commutative polynomial ring  $\mathbb{Z}[X]$  (i.e., polynomials that have coefficients from  $\mathbb{Z}$  and variables from  $X$ . See also Section 2.1).

Next, we define multivariate polynomials  $P_1, P_k \in \mathbb{Z}[X]$  AM:  $P_k \in \mathbb{Z}[X]$  for  $k \in \mathbb{N}$  iteratively as follows:

$$P_1 = \sum_{x \in X} x, \quad P_k = P_1^k$$

Following this construction, we observe that  $P_k$ , when expanded into a sum of multivariate monomials, contains a multilinear term if and only if the original  $k$ -3D matching instance can be answered in the positive. Thus, we have a successful reduction to multilinear monomial detection for the  $k$ -3D matching.

For instance with  $A = \{a_1, a_2\}$ ,  $B = \{b_1, b_2\}$ ,  $C = \{c_1, c_2, c_3\}$  and

$$T = \{(a_1, b_1, c_1), (a_1, b_2, c_2), (a_2, b_2, c_3)\}$$

(see Figure TODO: diagram of the 3D matching problem with the matchings), we have  $P_1 = a_1b_1c_1 + a_1b_2c_2 + a_2b_2c_3$ . For  $k = 2$ , we get

---

<sup>5</sup>Section 4.3 briefly overviews actually finding a solution

OM: I'm not sure what to put for  $\mathbb{Z}$  here, since we don't have any coefficients here yet

AM: = after line break

$$\begin{aligned}
P_2 &= (a_1b_1c_1 + a_1b_2c_2 + a_2b_2c_3)^2 \\
&= a_1^2b_1^2c_1^2 + a_1^2b_2^2c_2^2 + a_2^2b_2^2c_3^2 + 2a_1^2b_1b_2c_1c_2 + 2a_1a_2b_1b_2c_1c_3 + 2a_1a_2b_2^2c_2c_3.
\end{aligned}$$

We may see that the multilinear monomial  $2a_1a_2b_1b_2c_1c_3$  corresponds to a solution to our problem: the triples  $(a_1, b_1, c_1)$  and  $(a_2, b_2, c_3)$  are indeed disjoint, as seen in Figure TODO.

### 1.3 Detecting a multilinear monomial

We have now reached multilinear monomial detection from the perspective of combinatorial problems. Furthermore, we may reach this point from *parameterized* problems as well; instead of picking  $k$  triples or edges, we may pick, say,  $k - 3$  combinations by only computing  $P_{k-3}$  instead of  $P_k$ . Thus, the multilinear monomial detection approach works well with parameterization. The general, parameterized multilinear monomial detection problem is defined as follows:

*k*-MULTILINEAR MONOMIAL DETECTION

**Input:** A polynomial  $P_1 \in \mathbb{Z}[X]$  with the solution candidates encoded as monomials.

**Question:** Does the polynomial  $P_k = P_1^k$  extended as a sum of monomials contain a multilinear monomial of degree  $k$  <sup>6</sup>?

AM:  
footnote  
after  
question  
mark

Clearly, an upper bound for solving the problem is given by a naive expansion of  $P_1^k$  into a sum of monomials and scanning all the terms. However, this is not optimal: an  $N$ -degree polynomial will have  $e^N$  <sup>7</sup> possible monomials, and most problems using this algebrization (see Section 1.4) have been solved with faster algorithms. This motivates the detection of multilinear monomials without fully expanding  $P_k$  into a sum of monomials.

Since only multilinear terms are important in  $P_k$ , any squared variable can be instantly discarded as soon as it is formed during the computation of  $P_1^k$ . This can be achieved with dynamic programming to create a polynomial  $P'_k$  that only contains multilinear monomials. Since there are  $2^n$  multilinear monomials given  $n$  variables, this method results in a slightly faster algorithm than with naive expansion.

However, the underlying problems are usually FPT <sup>8</sup>. This implies that scaling exponentially with the number of variables is far from optimal. In order for the complexity of the algorithm to scale with the parameter  $k$ , we can reduce the number of variables by mapping  $\rho: X \rightarrow Y$ , where  $|X| \geq |Y|$  and  $|Y| = \Theta(k)$ , and dynamically expand  $(P'_1)^k \in \mathbb{Z}[Y]$  instead of  $P_k$ , where  $P'_1 \in \mathbb{Z}[Y]$  represents  $P_1 \in \mathbb{Z}[X]$  mapped with  $\rho$ . AM:

<sup>6</sup>a multilinear monomial with degree  $k$  has  $k$  indeterminates, see also Section 2.1

<sup>7</sup> $e^N$  is an approximation from  $\binom{n+N}{N}$ , where  $n$  is the number of variables

<sup>8</sup>fixed parameter tractable, i.e., polynomial to  $n$  in time if  $k$  is fixed. See Section 2.3



$P'_1 = \Phi_\rho(P_1) \in \mathbb{Z}[Y]$  and  $\Phi_\rho$  is the canonical extension of  $\rho$  to a ring homomorphism  $\mathbb{Z}[X] \rightarrow \mathbb{Z}[Y]$  (i.e., simply replacing every  $x \in X$  in  $P_1$  with  $\rho(x)$ )

However, since  $|X| \geq |Y|$ , a multilinear monomial in  $P_k$  may not be multilinear in  $P'_k$  since we may map two different variables of a multilinear monomial in  $X$  to the same variable in  $Y$ . For an algorithm to reliably detect a multilinear monomial, it is necessary to use different mappings from  $X$  into  $Y$  until a multilinear monomial survives the mapping. However, the probability that any given multilinear monomial survives [this mapping](#) [AM: a uniformly chosen mapping \(?\)](#) is around  $e^{-|Y|}$  [KW15]. This implies that  $\Omega(e^{|Y|})$  random mappings must be tried for a multilinear monomial to survive with a reasonable constant probability. Thus, a  $k$ -multilinear monomial is detected with a randomized algorithm in  $\mathcal{O}^*((2e)^{|Y|})$  time, where  $|Y| = \Theta(k)$ .

This is essentially the idea behind color coding introduced by Alon, Yuster, and Zwick [AYZ95]. Although here given in this algebraic form for the  $k$ -multilinear monomial detection, color coding is purely combinatorial, and does not rely on algebraic techniques. However, since  $k$ -multilinear monomial detection is a purely algebraic problem, it is reasonable to conjecture that there is an algebraic method for solving it.

Indeed, a faster algebraic method exists; the technique of algebraic fingerprinting first introduced by Koutis [Kou08] and further developed by Williams [Wil09] solves  $k$ -multilinear monomial detection in  $\mathcal{O}^*(2^k)$  time. Since the technique focuses on the abstract  $k$ -multilinear monomial detection, to which many parameterized problems can be reduced to, it gives a general framework for solving parameterized problems [KW15].

## 1.4 Related problems

The importance of multilinear monomial detection lies in the fact that many parameterized combinatorial problems, as we see in this section, can be reduced to instances of it. Thus, solving the general  $k$ -multilinear monomial detection problem efficiently has great interest. Currently, the technique of algebraic fingerprints underlies the fastest algorithms for all the problems mentioned here. Note that only time complexity is discussed here.

Koutis was the first one to introduce and apply multilinear monomial detection for combinatorial problems, namely the  $k$ -path and  $m$ -set  $k$ -packing problems [Kou08]. In their work [KW09], Koutis and Williams reduced the  $k$ -tree,  $k$ -leaf spanning tree, and  $t$ -dominating set problems to  $k$ -multilinear monomial detection, and gave  $\mathcal{O}^*(2^k)$  algorithms for these problems.

Björklund, Kaski, and Kowalik [BKK16] showed reductions for the  $k$ -sized graph motif problem and for several optimization variants such as *closest graph motif* and *maximum graph motif* problems which are relevant in e.g. bioinformatics. Cadena, Ekanayake, and

Vullikanti [CEV17] applied algebraic fingerprinting to general *graph scan statistics* which is a methodology that detects anomalies in a graph. This is relevant in general network design problems.

## 2 Preliminaries

It is necessary to recall basic algebraic concepts and common notation before further discussing multilinear monomial detection and algebraic fingerprinting. Section 2.1 gives definitions for some necessary algebras and algebraic concepts. Section 2.2 outlines arithmetic circuits.

Some lesser-known notation is also used. FPT or *fixed parameter tractable* is a class of parameterized problems that can be solved in polynomial time if the parameter is fixed.  $\text{poly}(n)$  refers to a polynomial function in  $n$ .  $\mathcal{O}^*$  hides factors polynomial to the input size, e.g.,  $\mathcal{O}(n^3 k^n) = \mathcal{O}^*(k^n)$ .

### 2.1 Algebra

We generally refer to the following algebraic objects as *algebraic structures*, i.e., objects that contain a set of elements and operations on them.

**Groups.** A *group*  $\mathbf{G}$  is a tuple  $(G, +)$ , where  $G$  is a set of elements,  $+: G \times G \rightarrow G$  is a binary operation closed under the elements in  $G$ ,  $+$  is associative, every element  $g \in G$  has an inverse  $g^{-1} \in G$ , and  $G$  contains an identity element  $e$  such that  $g + e = g$ ,  $g + g^{-1} = e$  and  $e = e^{-1}$ . Moreover,  $\mathbf{G}$  is called *Abelian* if  $+$  is also commutative. A group is called *multiplicative* if the operation is written as multiplication.

**Rings.** A *ring*  $\mathbf{R}$  is a triple  $(R, +, \cdot)$ , where  $(R, +)$  is an Abelian group, and  $\cdot: R \times R \rightarrow R$  is a binary operation closed under  $R$ . We call the binary operations  $+$  and  $\cdot$  addition and multiplication, respectively. A *commutative ring* has commutative multiplication.

Note, from here on we may use a bold typeface to represent either the set of elements or the algebraic structure itself, i.e., we may use expressions such as  $u \in \mathbf{R}$ , where  $\mathbf{R} = (R, +, \cdot)$  and  $u \in R$ . Moreover, multiplication may be denoted by juxtaposition for brevity:  $a \cdot b = ab$ .

$\mathbf{R}$  must contain a multiplicative identity  $\mathbf{1} \in \mathbf{R}$  such that  $\forall a \in \mathbf{R}: a \cdot \mathbf{1} = a$ . We notate the additive identity  $e$  as  $\mathbf{0}$  from here on. Observe that for any  $R \neq \{\mathbf{0}\}$ ,  $\mathbf{1} \neq \mathbf{0}$ . Left and right distributive laws hold for rings, i.e.,

$$\forall a, b, c \in \mathbf{R}: a \cdot (b + c) = (ab) + (ac) \wedge (b + c) \cdot a = (ba) + (ca).$$

$u \in \mathbf{R}$  is called *unit* if it holds that  $\exists v \in \mathbf{R}: uv = vu = \mathbf{1}$ , i.e., it has a multiplicative

inverse  $v \in \mathbf{R}$ .

**Fields.** A *field*  $\mathbf{F} = (F, +, \cdot)$  is defined with the following conditions:

- $(F, +)$  is an Abelian group
- $(F \setminus \{\mathbf{0}\}, \cdot)$  is an Abelian group
- Left and right distributive laws hold for  $\mathbf{F}$ .

Equivalently, a ring is a field if every non-zero element is unit,  $\mathbf{1} \neq \mathbf{0}$ , and multiplication is commutative. The *characteristic* of a field  $\mathbf{F}$  is defined as follows:

$$\text{char}(\mathbf{F}) = \begin{cases} \min\{n \in \mathbb{N} : n \cdot \mathbf{1} = \mathbf{0}\} \\ 0 \end{cases} \quad \text{if such } n \text{ does not exist} \quad (1)$$

Note, that a field  $\mathbf{F}$  with characteristic 2 satisfies the following:

$$\forall u \in \mathbf{F} : u + u = u \cdot (\mathbf{1} + \mathbf{1}) = u \cdot \mathbf{0} = \mathbf{0}$$

Throughout the thesis we may use language such as *cancellation due to characteristic* to refer to the fact that an element  $f \in \mathbf{F}$  may cancel itself out in an expression if  $\mathbf{F}$  has non-zero characteristic. We mainly use characteristic 2 in this thesis, and thus, we may say that  $f$  cancels out due to characteristic when it has an even coefficient:  $f + f + f + f = 4f = 2f + 2f = \mathbf{0} + \mathbf{0} = \mathbf{0}$ . **AM:** For every  $k \geq 0$ ,  $2kf = k(2f) = k \cdot \mathbf{0} = \mathbf{0}$

Characteristics are prevalent in *finite fields*, where the set of elements is finite. A simple example of a finite field is  $\mathbb{Z}_2 = (\{\mathbf{0}, \mathbf{1}\}, +, \cdot)$ , where the operations are standard addition modulo 2, and standard multiplication.

Finite fields can be noted as  $GF(p)$ , called *Galois fields*, where the field is unique up to *isomorphism* if  $p$  is a prime or a power of some prime. An isomorphism  $\kappa : \mathbf{A} \rightarrow \mathbf{B}$  is a bijective mapping that satisfies  $\forall u, v \in \mathbf{A} : \kappa(uv) = \kappa(u)\kappa(v) \wedge \kappa(u + v) = \kappa(u) + \kappa(v)$ . Thus, if we only care for the relations between the elements, we may say that algebraic structures that isomorphic are equivalent for us. Note that if  $p$  is prime and  $k \in \mathbb{N}$ , the characteristic of  $GF(p^k)$  is  $p$ .

**Polynomial rings and multilinearity.** **AM:** For a (finite) set  $X$ , A *polynomial ring* in  $X$ , noted as  $\mathbf{K}[X]$ , is a ring of polynomials with indeterminates (or variables) from  $X$  and coefficients from the commutative ring  $\mathbf{K}$ . In this thesis, we only handle *multivariate* polynomials, i.e., polynomials that have  $|X| = n > 1$ .

A *monomial* in  $X$  is of the form  $X_1^{a_1} X_2^{a_2} \cdots X_n^{a_n}$ , where  $X_i \in X$  and  $a_i \in \mathbb{N}$  marks the exponent. A monomial is *multilinear* if  $\forall i \in \{1, \dots, n\} : a_i = 0 \vee a_i = 1$ . The *degree* of

a monomial  $m$  is the sum of the exponents:  $\deg(m) = \sum_{i=1}^n a_i$ . Any element  $P$  of such polynomial ring is a finite linear combination of these monomials with coefficients  $k \in \mathbf{K}$ :

$$P = \sum_p k_p (X_1^{a_{1,p}} \cdots X_n^{a_{n,p}})$$

AM:

$$P = \sum_{p_1, \dots, p_n \geq 0} k_{p_1, \dots, p_n} X_1^{a_{p_1}} \cdots X_n^{a_{p_n}}$$

Alternatively, a polynomial ring  $\mathbf{K}[X]$  can be thought of as a vector space **AM: over  $\mathbf{K}$  of infinite dimension** with the **AM: set of** all the monomials as a basis. Polynomial rings have three operations: addition, multiplication and scalar multiplication. In terms of vectors, addition and scalar multiplication follow the standard addition and scalar multiplication for vectors. Note that the scalar value must be in  $\mathbf{K}$ . The multiplication is defined as follows: let  $M(X)$  denote the set of all monomials in  $X$ , and  $P_d$  denote a polynomial in  $\mathbf{K}[X]$  of the form  $P_d = \sum_{m \in M(X)} d_m m$ , where  $d_m \in \mathbf{K}$ . Then,

$$\forall P_a, P_b \in \mathbf{K}[X]: P_a \cdot P_b = \left( \sum_{m \in M(X)} a_m m \right) \cdot \left( \sum_{n \in M(X)} b_n n \right) = \sum_{m, n \in M(X)} (a_m b_n) (mn)$$

Note that the definition for the coefficient  $k_m$  essentially decides whether a monomial  $g \in M(X)$  appears in a polynomial. If a polynomial  $P$  does not contain the monomial  $g$ , then it must be that  $k_g = \mathbf{0} \in \mathbf{K}$ .

*Evaluation of a polynomial  $P$  at  $Z$  is denoted  $P(Z)$ , where  $Z \subset \mathbf{R}$  is a set of elements from a ring  $\mathbf{R}$  containing  $\mathbf{K}$ .* Evaluating at  $Z$ , i.e., assigning indeterminates  $X$  to  $Z$ , defines an element from the ring  $\mathbf{R}$ , as  $P(Z) \in \mathbf{R}$ . We may note the unevaluated polynomial as  $P(X)$  to highlight the set of indeterminates. A polynomial is identical to  $\mathbf{0}$ , or *identically evaluates* to  $\mathbf{0}$ , if  $\forall W \in \mathbf{R}: P(W) = \mathbf{0}$ .

AM: I think  $Z$  must be a map  $X \rightarrow \mathbf{R}$

In this thesis, we mainly use elements of *group algebras* for evaluation, i.e.,  $\mathbf{R} = \mathbf{K}[\mathbf{G}]$ , where  $\mathbf{G}$  is a multiplicative group.

**Group algebras.** A group algebra is an algebraic structure very similar to a polynomial ring, and is also noted as  $\mathbf{R}[\mathbf{G}]$ , where  $\mathbf{R}$  is a commutative ring and  $\mathbf{G}$  is a multiplicative group. A group algebra may be thought of as a polynomial ring  $\mathbf{R}[X]$ , where the indeterminates  $X$  form a multiplicative group  $\mathbf{G}$ . The elements and operations in a group algebra have the same form as those in a polynomial ring, with the exception that instead of summing over monomials in  $X$ , we sum over the elements of  $\mathbf{G}$ .

Note that  $\mathbf{0} \in \mathbf{R}[\mathbf{G}]$  corresponds to a polynomial  $P \in \mathbf{R}[\mathbf{G}]$  with zero-coefficients, and  $\mathbf{1} \in \mathbf{R}[\mathbf{G}]$  corresponds to the identity element of  $\mathbf{G}$ .

## 2.2 Arithmetic circuits

Computing arithmetic expressions such as polynomial evaluations can be represented by *arithmetic circuits*, that is, connected acyclic directed graphs, where terminal vertices correspond to initial and output values, and other vertices correspond to *arithmetic gates*, i.e., arithmetic operations.

In this thesis, we only consider *addition* and *multiplication* gates. These gates correspond to the binary operations of the underlying algebra which the arithmetic expression is evaluated over and given initial values from. Thus, the addition and multiplication gates may have *fan-in two* and *fan-out one*, i.e., each gate-vertex has three edges connected to it: two as an input, and one as an output. If the underlying algebra is commutative, we may call the arithmetic circuit a *commutative circuit*.

For example, an abstract evaluation of a polynomial  $P \in \mathbb{Z}[\{a, b, c, d\}]$  of the form  $P = (ab + cd)^2$  is given by the arithmetic circuit presented in Figure 1. Note that we use the same symbols for addition and multiplication gates as seen in the figure in all arithmetic circuit diagrams in this thesis.

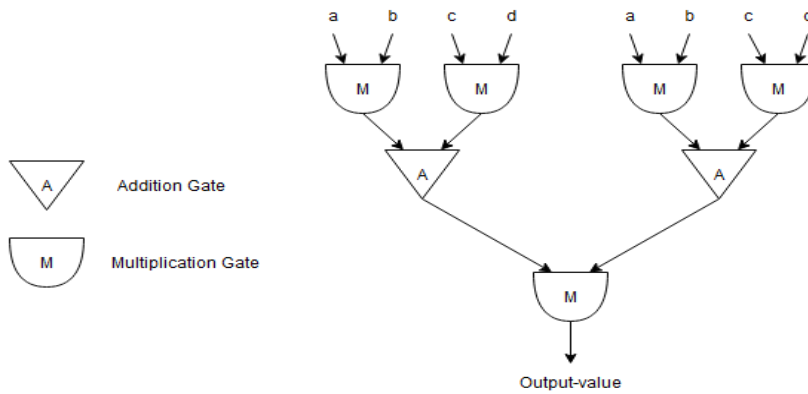


Figure 1: A circuit for evaluating  $(ab + cd)^2$ . Note that the input terminals have been copied for clarity.

## 2.3 Notation and other terminology

The following table holds useful notation and their definitions. The terms and notation are given on the left, and their definitions on the right.

$k$ -multilinear monomial	a multilinear monomial that has a degree of $k$
sum of monomials form	an expression for a polynomial $P(X)$ where there are no polynomial multiplications present, i.e., the expression consists of only additions between monomials
$\mathcal{O}^*$	hides factors polynomial to the input size, e.g., $\mathcal{O}(n^3k^n) = \mathcal{O}^*(k^n)$
FPT	A class of <i>fixed parameter tractable</i> problems, i.e., parameterized problems that can be solved in polynomial time if the parameter is fixed
$\text{poly}(n)$	a polynomial function in $n$ , e.g., $an^4+bn+c$
Deterministic algorithm	Given a fixed input, a <i>deterministic</i> algorithm always gives the same output
Randomized algorithm	A <i>randomized</i> algorithm relies on pseudorandom functions, and may not produce the same outputs for a fixed input
Schwartz-Zippel lemma	TODO: give here or in the context?
term	def

Schwartz-Zippel ref:

J. T. Schwartz, “Fast probabilistic algorithms for verification of polynomial identities,” Journal of the ACM, vol. 27, no. 4, pp. 701–717, 1980.

AM: Drop example for polynomial in  $n$   
AM: No need to define deterministic / randomized algorithm  
AM: S-Z lemma in text seems better  
AM:  
 $k$ -multilinear monomial is better defined before in context

### 3 General multilinear monomial detection

The detection of multilinear monomials in a multivariate polynomial is a fundamental problem, since many important problems can be reduced to it, as seen in Section 1.4. Therefore, any progress in general multilinear monomial detection implies faster algorithms for the problems mentioned in Section 1.4.

In this section, we discuss the algebraic ideas by Koutis and Williams [Kou08; Wil09; KW09] for the  $k$ -multilinear monomial detection. In Section 3.1, we dive into the technique of *algebraic fingerprinting*, and see the ideas and implementations of Koutis and Williams. In Section 3.2 we briefly discuss the time and space complexity of the general algebraic framework given by algebraic fingerprinting. Finally, Section 4 discusses the limits of algebraic fingerprinting.

As a prelude for the following sections, recall the relevant problem definition:

$k$ -MULTILINEAR MONOMIAL DETECTION

**Input:** A commutative arithmetic circuit  $A$  representing a polynomial  $P(X)$ .

**Question:** Does the polynomial  $P(X)$  extended as a sum of monomials contain a multilinear monomial of degree  $k$ ?

#### 3.1 Algebraic fingerprinting

In Section 1.3, we discussed non-algebraic methods for approaching  $k$ -multilinear monomial detection. In particular, we mention an interesting idea of discarding squared variables when dynamically computing the circuit. From this, we build towards algebraic fingerprinting.

This idea of discarding squared variables as soon as they are formed in  $A$  can be expressed mathematically: any squared variable should be identical to zero.

$$\forall x \in X : x^2 = \mathbf{0} \tag{2}$$

This implies that any non-multilinear monomial in  $X$  will evaluate to zero in  $P(X)$ , since any non-multilinear monomial  $q$ , assuming commutativity, can be written as  $x^2 q'$  for some  $x \in X$ , and  $x^2 q' = \mathbf{0} q' = \mathbf{0}$  over any algebra. Therefore,  $P(X)$  will identically evaluate to zero if there are no multilinear monomials, i.e., there exist no solutions to the original problem.

This is the basis of algebraic multilinear monomial detection introduced by Koutis [Kou08], or later referred to as *algebraic fingerprinting* [KW15]: we evaluate  $P(X)$  over some algebraic structure  $\mathbf{G}$ , and detect a multilinear monomial from the value returned by this

evaluation. Ideally, with any evaluation of  $P(X)$  at any  $Z \subset \mathbf{G}$ ,  $|Z| = |X|$  where  $w \neq \mathbf{0}$ ,

$$P(Z) = \begin{cases} \mathbf{0} & \text{if no multilinear monomials exist} \\ w & \text{otherwise} \end{cases} \quad (3)$$

In the following subsections, we specify an appropriate algebraic structure such that (3) is met, as well as some requirements for efficiency. Then, we discuss the works of Koutis and Williams [Kou08; Wil09], and see how these specifications were implemented.

This thesis refers to the general framework provided by this algebraic technique for parameterized problems [KW09; KW15] as algebraic fingerprinting. However, algebraic fingerprinting can also be used to refer specifically to the idea behind solving a problem with multilinear monomials canceling out due to characteristic, which is discussed in Section 3.1.3.

### 3.1.1 Specifications for the algebraic structure

We have arrived at an important task for multilinear monomial detection: find an appropriate algebraic structure  $\mathbf{G}$  for the evaluation of  $P$  over  $\mathbf{G}$  to meet (2) and thus the first equality in (3). We specify for commutative algebraic structures such as commutative rings for multilinear monomial detection;  $\mathbf{G}$  should have commutative multiplication in order for (2) to be effective. The ordering of indeterminates in monomials is generally unknown, since the specific algebrization into multilinear monomial detection is abstracted away.

For the second equality in (3), it is necessary that multilinear monomials in  $P(X)$  are not identical to  $\mathbf{0}$  over  $\mathbf{G}$ . More specifically,  $w$  should not be identical to  $\mathbf{0}$ . Thus, it must be that  $k \leq |\mathbf{G}|$ , where  $k$  is the degree of the monomials. Although this is not enough to ensure that  $w \neq \mathbf{0}$  (as seen in the following section), we leave the issue for now.

These are the necessary specifications for  $\mathbf{G}$  for algebraic multilinear monomial detection. However, this algebraic detection must be efficient for it to be useful. Therefore, further requirements are necessary: (a) the binary operations of  $\mathbf{G}$  must be efficiently computable for a fast evaluation of  $P(X)$ , and (b) multilinear monomials in  $P(X)$  must survive the evaluation of  $P$  under an assignment  $\gamma: X \rightarrow \mathbf{G}$  with a constant probability. We reason the sudden appearance of *survivability* and probabilities soon.

Recall from Section 1.3 that with color coding, multilinear monomials can be detected with a randomized algorithm in  $\mathcal{O}^*((2e)^k)$  time. Thus for (a), we may specify for an evaluation of  $P(X)$  to take  $\mathcal{O}^*(2^k)$  time. The polynomial  $P(X)$  will have at most  $2^k$  multilinear monomials. Therefore, it is necessary that the binary operations of  $\mathbf{G}$  take polynomial time.



With (b), recall that multilinear monomials survive color coding with probability  $e^{-k}$ . With algebraic fingerprinting, although not identical to zero, a multilinear monomial in  $P(X)$  can still evaluate to  $\mathbf{0} \in \mathbf{G}$  under some assignment  $\gamma: X \rightarrow \mathbf{G}$ . However, if we specify for a constant probability of survival under random  $\gamma$ , we can reliably decide whether a multilinear monomial exists by evaluating  $P(X)$  in  $\mathbf{G}$  over a constant number of randomized assignments  $X \rightarrow \mathbf{G}$ . Relating to color coding, this would essentially remove the  $e^k$  factor in  $\mathcal{O}^*((2e)^k)$ .

To restate, we now have to find such a commutative ring  $\mathbf{G}$  that meets the following specifications:

- $\forall g \in \mathbf{G}: g^2 = \mathbf{0}$
- Operating over  $\mathbf{G}$  should be fast, i.e., evaluating  $P(X)$  should take  $\mathcal{O}^*(2^k)$  time.
- Multilinear monomials should evaluate to non-zero through a random assignment  $\gamma: X \rightarrow \mathbf{G}$  with a reasonable constant probability.

In the work that introduced this algebraic fingerprinting technique [Kou08], Koutis used the group algebra  $\mathbb{Z}_2[\mathbb{Z}_2^k]$ . Williams developed the technique further, generalizing the algebra to  $GF(2^l)[\mathbb{Z}_2^k]$  [Wil09]. Next, we look at these group algebras of  $\mathbb{Z}_2^k$  for  $\mathbf{G}$ .

### 3.1.2 Using group algebras of $\mathbb{Z}_2^k$

The multiplicative group  $\mathbb{Z}_2^k$  consists of  $k$ -dimensional  $\{0,1\}$ -vectors with the binary operation defined as component-wise addition modulo 2. For example with  $k = 3$ ,

$$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \mathbf{0} \in \mathbb{Z}_2^3.$$

Observe that in general, every element in  $\mathbb{Z}_2^k$  is its own inverse:

$$\forall z \in \mathbb{Z}_2^k: z^2 = \mathbf{0}. \quad (4)$$

Recall that the elements of a group algebra  $\mathbf{F}[\mathbb{Z}_2^k]$  are linear combinations of the form

$$\sum_{v \in \mathbb{Z}_2^k} a_v v,$$

where  $a_v \in \mathbf{F}$ . From here on, we note the identity of  $\mathbb{Z}_2^k$  as  $v_0$ , additive and multiplicative identities of  $\mathbf{F}$  as  $\mathbf{0}_F$  and  $\mathbf{1}_F$ , respectively, and use  $\mathbf{0}$  and  $\mathbf{1}$  for the identities of  $\mathbf{F}[\mathbb{Z}_2^k]$ . Recall that  $\mathbf{1} = v_0$ , and  $\mathbf{0}$  corresponds to the element  $\sum_{v \in \mathbb{Z}_2^k} a_v v$ , where  $a_v = \mathbf{0}_F$ . Note

that  $+$  represents addition over  $\mathbf{F}[\mathbb{Z}_2^k]$  and not addition over  $\mathbb{Z}_2^k$  which is a multiplicative group.

In his introductory work for this technique [Kou08], Koutis assigned  $X$  to elements of the form  $(v_0 + v_i) \in \mathbf{F}[\mathbb{Z}_2^k]$ , such that for every  $x_i \in X$ , a random  $v_i \in \mathbb{Z}_2^k$  is independently and uniformly picked for the assignment  $\gamma(x_i) = v_0 + v_i$ . We note the assigned values as  $\bar{X}$ , and the resulting element from the polynomial evaluation as  $P(\bar{X}) \in \mathbf{F}[\mathbb{Z}_2^k]$ . Koutis observed that due to (4), for all  $v \in \mathbb{Z}_2^k$  and  $(v_0 + v) \in \mathbf{F}[\mathbb{Z}_2^k]$ ,

$$(v_0 + v)^2 = v_0^2 + v_0v + vv_0 + v^2 = v_0 + v + v + v_0 = 2v_0 + 2v.$$

This implies that if we pick a field with characteristic 2 for  $\mathbf{F}$ ,  $\forall v_i \in \mathbb{Z}_2^k: (v_0 + v_i)^2 = \mathbf{0}$ . Thus, non-multilinear monomials in  $P(X)$  vanish in  $P(\bar{X})$ , and the first equation in (3) will hold.

However, if  $\mathbf{F}$  has characteristic 2, the second equation of (3) does not necessarily hold, and it may be that  $w = \mathbf{0}$ . Multilinear monomials may cancel each other out in  $\mathbf{F}[\mathbb{Z}_2^k]$ , since they may have even leading coefficients in  $P(X)$ . Next, we see how Koutis approached this problem.

### 3.1.3 Fingerprints to prevent unwanted cancelation

Indeed, if we take the  $k$ -3D matching instance from Section 1.2, we notice that the multilinear monomials in  $P_2$  have even coefficients, and thus would cancel out in  $\mathbf{F}[\mathbb{Z}_2^k]$  due to characteristic.

$$\begin{aligned} P_2 &= (a_1b_1c_1 + a_1b_2c_2 + a_2b_2c_3)^2 \\ &= a_1^2b_1^2c_1^2 + a_1^2b_2^2c_2^2 + a_2^2b_2^2c_3^2 + 2a_1^2b_1b_2c_1c_2 + 2a_1a_2b_1b_2c_1c_3 + 2a_1a_2b_2^2c_2c_3. \end{aligned}$$

In general, when  $k$  is even, multilinear monomials will have even coefficients.

To tackle this issue, one idea is to add auxiliary indeterminates, called *fingerprints* [KW15], to the monomials in order to make them unique. For example, let  $S = \{s_1, s_2, \dots\}$  be the set of an appropriate number of fingerprints and augment them to  $P_k$  as follows:

$$\begin{aligned} P'_2 &= (s_1a_1b_1c_1 + s_2a_1b_2c_2 + s_3a_2b_2c_3)(s_4a_1b_1c_1 + s_5a_1b_2c_2 + s_6a_2b_2c_3) \\ &= s_1s_4a_1^2b_1^2c_1^2 + s_2s_5a_1^2b_2^2c_2^2 + s_3s_6a_2^2b_2^2c_3^2 + \\ &\quad s_1s_5a_1^2b_1b_2c_1c_2 + s_2s_4a_1^2b_1b_2c_1c_2 + s_1s_6a_1a_2b_1b_2c_1c_3 + \\ &\quad s_3s_4a_1a_2b_1b_2c_1c_3 + s_2s_6a_1a_2b_2^2c_2c_3 + s_3s_5a_1a_2b_2^2c_2c_3. \end{aligned}$$

Then, the multilinear monomial detection algorithm could assign every  $a \in X \cup S$  to  $\mathbf{F}[\mathbb{Z}_2^k]$ .

AM: Now the multilinear monomial  $a_1a_2b_1b_2c_1c_3$  only cancels itself out in  $\rho(P'_2)$  ( $P'_2(Z)$ )?

AM:  
Actually  $S = \{s_1, \dots, s_6\}$ ?  
AM: Use  
`\align`

unsure about notation) if  $\rho(s_1)\rho(s_6) = \rho(s_3)\rho(s_4)$  under the assignment  $\rho \dots$ . We can control depending on how the  $\rho(s_i)$  are chosen... With this, non-multilinear monomials will still vanish, but multilinear monomials will not identically cancel each other out when  $k$  is even.

However, introducing new indeterminates raises the degree of multilinear monomials. Therefore, it increases the probability that indeterminates in a multilinear monomial are assigned the same value from  $\mathbf{F}[\mathbb{Z}_2^k]$ , which results in the multilinear monomial evaluating to zero with higher probability. Counteractively raising the dimension of  $\mathbb{Z}_2^k$ , however, would exponentially slow down matrix multiplications which prove to be important for efficiency in algebraic fingerprinting (see Section 3.2).

AM: ??

Koutis approached this problem by assigning fingerprints to elements from a different algebraic structure: he used  $S \rightarrow \mathbb{Z}_2$  and set  $\mathbf{F} = \mathbb{Z}_2$  [Kou08]. In this sense, fingerprints are defined to be auxiliary coefficients instead of auxiliary indeterminates. Note that  $\mathbb{Z}_2$  has characteristic 2. With this, Koutis essentially assigns multilinear monomials a coefficient randomly from  $\{\mathbf{0}_F, \mathbf{1}_F\}$ . The idea is that a multilinear monomial, assigned with the fingerprint  $\mathbf{1}_F$ , survives the cancelation due to characteristic if the canceling pair is assigned  $\mathbf{0}_F$ . With randomized assignments  $S \rightarrow \mathbb{Z}_2$ , there is a constant positive probability that a multilinear monomial will have an odd coefficient, and thus survive the assignment [Kou08].

In practice, fingerprints can be implemented into the algebrization as follows (see also Figure 2): for every multiplication gate  $G_i$  feeding into an addition gate in the arithmetic circuit  $A$  for  $P(X)$ , pick a unique  $s_i \in S$ . Insert a new multiplication gate  $\overline{G}_i$  that takes as input  $s_i$  and the output of  $G_i$ . The output of  $\overline{G}_i$  feeds to the gates that read the output of  $G_i$ . We note the new polynomial represented by this circuit as  $P'(X, S)$ . Note that here Koutis picked random elements from  $\mathbb{Z}_2$  for  $s_i$ .

### 3.1.4 Polynomial identity testing

From another perspective, we may look at algebraic fingerprinting as *polynomial identity testing*. That is, we compute  $P'(X, S)$  over assignments  $X \rightarrow \mathbf{F}[\mathbb{Z}_2^k]$  into  $P'(\overline{X}, S)$ , i.e., compute until the gates  $\overline{G}_i$ . Imagine we stop here in the circuit before assigning the fingerprints  $S$  and continuing with the multiplication. Currently, we may see  $P'(\overline{X}, S) \in \mathbf{K}[S]$  as a polynomial with indeterminates from  $S$  and coefficients from  $\mathbf{K} = \mathbf{F}[\mathbb{Z}_2^k]$ . In a sense, we switch the roles of coefficients and indeterminates in  $P'(X, S)$  to find a new polynomial  $Q(S) \in \mathbf{K}[S]$ .

Now, deciding whether a multilinear monomial exists in  $P(X)$  is essentially given by whether the polynomial  $P'(\overline{X}, S) \in \mathbf{K}[S]$  is identical to  $\mathbf{0}$ , i.e., with  $\Phi$  representing the

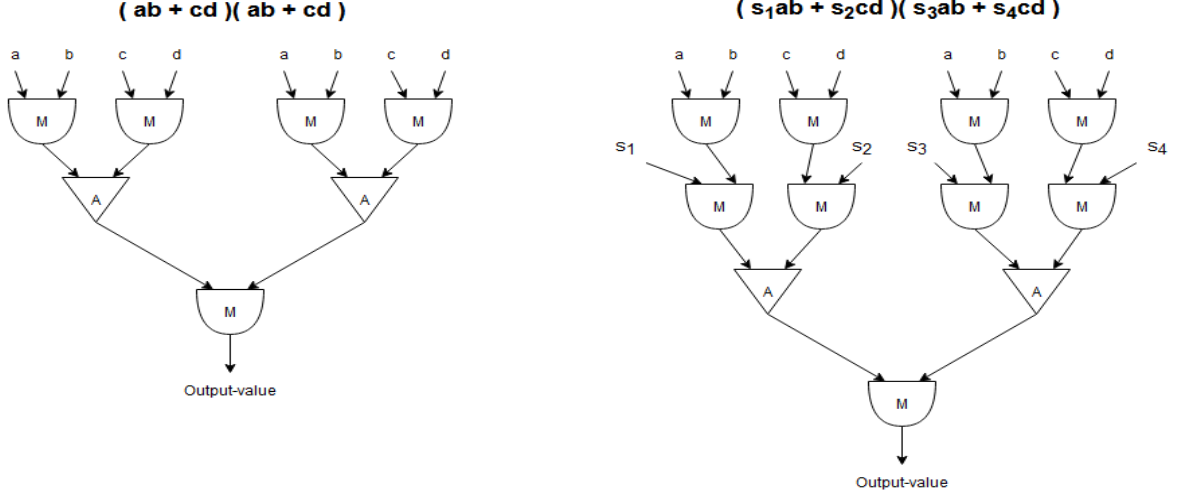


Figure 2: A fingerprinted circuit for evaluating  $(ab + cd)^2$ . Note the added terminals on the right circuit.

family of mappings  $S \rightarrow \mathbf{F}$  and  $\phi(S)$  representing the assigned values of  $S$  with  $\phi$ ,

$$\forall \phi \in \Phi: P'(\overline{X}, \phi(S)) = \mathbf{0}.$$

Therefore, we formulate algebraic fingerprinting as polynomial identity testing, i.e., we test wheter  $P'(\overline{X}, S) \in \mathbf{K}[S]$  is identical to  $\mathbf{0}$ .

#### POLYNOMIAL IDENTITY TESTING

**Input:** An arithmetic circuit  $C$  that computes the polynomial  $Q(S)$ .

**Question:** Is  $Q(S)$  identical to the zero polynomial?

We frame  $P'(\overline{X}, S)$  as  $Q(S) \in \mathbf{K}[S]$ . Here, it is necessary to test whether  $Q(S)$  is identical to *zero modulo 2*, since we used characteristic 2 to eliminate the underlying non-multilinear monomials in  $P'(\overline{X}, S)$ .

Thus, multilinear monomial detection is reduced via algebraic fingerprinting to polynomial identity testing. A multilinear monomial is detected if  $Q(f) \neq \mathbf{0}_F \in \mathbf{F}$ , where  $Q(f)$  represents evaluation under an assignment  $S \rightarrow \mathbf{F}$ , with any field  $\mathbf{F}$  of characteristic 2. We note the family of assignments  $S \rightarrow \mathbf{F}$  as  $\Phi$ .

Assume multilinear monomials exist in  $P(X)$ . Koutis achieved to detect a multilinear monomial with a probability  $(1/4 + 1/4k)$  by testing whether  $Q(S) = \mathbf{0}$  over the field  $\mathbf{F} = \mathbb{Z}_2$  with randomized assignments  $S \rightarrow \mathbb{Z}_2$  [Kou08]. Williams developed the technique of algebraic fingerprinting further by observing that due to the Schwartz-Zippel lemma, if we raise the order of the field  $\mathbf{F}$  such that the number of different assignments in  $\Phi$  is much larger than the number of assignments  $\phi \in \Phi$  that have  $Q(\phi(S)) = \mathbf{0}_F$ ,  $Q(\delta(S)) \neq \mathbf{0}_F$  with a high probability over some  $\delta \in \Phi$  [Wil09].

Of course,  $\mathbf{F}$  must have characteristic 2. For this, Williams used the field  $GF(2^l)$  with  $l \in \mathbb{N}$ . [Wil09]. By Schwartz-Zippel lemma,  $Q(S)$  evaluates to  $\mathbf{0}_F$  over a random assignment  $S \rightarrow GF(2^l)$  with probability at most  $k/2^l$  [Wil09]. For example with  $k$ -path, Williams used  $l = 3 + \log_2 k$ , which sets the probability of a false-negative evaluation to at most  $1/2^3$ . For the  $k$ -path problem, Koutis gave a randomized  $\mathcal{O}^*(2^{3k/2})$  time algorithm [Kou08], and Williams developed this into a randomized  $\mathcal{O}^*(2^k)$  time algorithm [Wil09] with the ideas discussed here.

In the following section, we briefly touch on the complexities of the algebras given here.

### 3.2 Time and space complexity

In the previous section, we discussed the specifications for an algebraic structure to implement algebraic fingerprinting. For an appropriate algebra, we found  $GF(2^l)[Z_2^k]$ . However, we did not discuss the costs related to this algebra. In this section, we briefly discuss the time and space complexity of algebraic fingerprinting.

Define the evaluation of  $P'(X, S)$  at any  $X$  and  $S$  to take  $g(n)$  number of arithmetic operations, where  $n = |X|$ . In general, detecting  $k$ -degree multilinear monomials in  $P(X)$  takes  $\mathcal{O}^*(2^k g)$  time and  $\mathcal{O}(\text{poly}(n, k))$  space [Wil09].

Actually, the complexities given by Koutis [Kou08] and Williams [Wil09] did not directly come from  $\mathbb{Z}_2[Z_2^k]$  or  $GF(2^l)[Z_2^k]$ . To make the complexity claims in [Kou08], Koutis used a well-known [Ter99] isomorphism  $\kappa: \mathbb{Z}[Z_2^k] \rightarrow \mathcal{M}$ , where  $\mathcal{M} = \mathbf{M}^{2^k \times 2^k}$  is an algebra of special permutation matrices with binary entries. To translate  $\mathbb{Z}[Z_2^k]$  to  $\mathbb{Z}_2[Z_2^k]$ , it is enough to take the modulo 2 of the coefficients  $a_v$  in the elements of  $\mathbb{Z}[Z_2^k]$ :

$$\sum_{v \in \mathbb{Z}_2^k} a_v v \in \mathbb{Z}[Z_2^k] \implies \sum_{v \in \mathbb{Z}_2^k} (a_v \bmod 2) v \in \mathbb{Z}_2[Z_2^k]$$

AM: No need  
for  
 $\displaystyle$

AM: Use  
 $\backslash \bmod$

Since Koutis used an isomorphism, all the ideas given in Section 3.1 still hold for detecting multilinear monomials. That is, the given implementation for (3) is valid through  $\kappa$ .

Using these matrices, Koutis [Kou08] detected  $k$ -multilinear monomials by computing the trace of the matrix resulting from the evaluation of  $P_S(X)$  over  $\mathcal{M}$ . The trace can be computed in time  $\mathcal{O}(2^k(nk + t))$  and space  $\mathcal{O}(nk + s)$ , where  $t$  and  $s$  are the time and space complexity of the  $g(n)$  arithmetic operations, respectively.

OM: Maybe  
don't go  
further than  
this  
AM: I would  
rather  
expand in  
Sect. 5 than  
here  
OM: add to  
prelims?

For  $GF(2^l)[Z_2^k]$ , the same ideas are used, though we omit further discussion into this for the sake of brevity. Thus, we have a general  $\mathcal{O}^*(2^k g)$  time and  $\mathcal{O}(\text{poly}(n, k))$  space algorithm for detecting  $k$ -degree multilinear monomials in  $P(X)$ , where  $g$  is the number of gates in the arithmetic circuit for  $P(X)$  and  $n = |X|$  [KW09].

It is important to note here that this complexity is for  $k$ -degree multilinear monomial

detection. Whether a parameterized problem with a parameter  $k$  receives an  $\mathcal{O}^*(2^k)$  time algorithm, is determined by the algebraization of said parameterized problem. For example for  $k$ -path, algebraic fingerprinting gives an  $\mathcal{O}^*(2^k)$  time algorithm [Wil09]. For  $k$ -3D-matching however, as given in Section 1.2, algebraic fingerprinting gives an  $\mathcal{O}^*(2^{3k})$  algorithm, since  $k$ -3D-matching reduces into  $3k$ -degree multilinear monomial detection.

## 4 Limits of algebraic fingerprinting

In this section, we discuss some limiting factors to the general algebraic framework given by algebraic fingerprinting for parameterized problems. First, we discuss the lower bound for the runtime. Then, we discuss the difficulties for derandomization. Finally, we touch on the issue of finding a solution for the underlying combinatorial problem whereas algebraic fingerprinting only detects that one exists.

### 4.1 Algebraic optimization

In Section 3.2, it was established that  $k$ -multilinear monomials in  $P(X)$  are detected with algebraic fingerprinting in  $\mathcal{O}^*(2^k g)$  time, where  $g$  is the number of gates in the arithmetic circuit for  $P(X)$  [Wil09]. This time was achieved by evaluating the circuit over matrix representations of the algebra  $GF(2^l)[\mathbb{Z}_2^k]$ .

Koutis and Williams showed that this particular algebra is nearly optimal for time complexity [KW09]: there is no significantly faster algebra that is appropriate for the general  $k$ -multilinear monomial detection. The authors proved that if any commutative algebra  $\mathcal{G}$  is used for detecting  $k$ -multilinear monomials, the lengths of elements in  $\mathcal{G}$  must be  $\Omega(2^k/k)$ .

Thus for the general  $k$ -multilinear monomial detection, we may say that algebraic fingerprinting is in some sense optimized. However, ideas from algebraic fingerprints can be utilized for faster algorithms for specific  $k$ -multilinear monomial detection instances (see Section 5).

### 4.2 Derandomization

Algebraic fingerprinting gives a randomized algorithm for  $k$ -multilinear monomial detection. The derandomization of the general framework seems hard, since algebraic fingerprinting uses the fact that polynomial identity testing can be solved in polynomial time with a randomized algorithm [Wil09].

If algebraic fingerprinting can be derandomized in polynomial time, it implies that

polynomial identity testing can be solved with a polynomial time deterministic algorithm. However, this implies proving superpolynomial lower bounds for arithmetic circuit sizes, which is notoriously hard [KI03]. Therefore, with respect to the several decades of sustained effort directed at circuit lower bounds [KI03], it seems unlikely that someone discovers a derandomization of polynomial identity testing.

### 4.3 Finding the solution

In many instances of multilinear monomial detection, a solution to the underlying problem can be directly found from the multilinear monomials by reverse-engineering the algebrization. However in algebraic fingerprinting, we do not have this information on the monomials: when we assign the variables values and evaluate the polynomial, we may only detect the presence of a solution. As such, algebraic fingerprinting applies to decision algorithms.

However, finding a solution when its presence is detected *seems easy*. Moreover, since most of the problems in Section 1.4 have search problems in FPT, a solution is found in polynomial time when its existence is detected.

AM: ??

For example, in [Kou08], [Kou08] gives an algorithm that detects a  $k$ -path, and shows that a  $k$ -path can be found by  $\mathcal{O}^*(n + \min(k^2, m))$  applications of the given algorithm. In general for subgraph problems, we may find a solution by calling the decision algorithm a polynomial number of times for different subgraphs of the original input graph.

AM: What are  $n$  and  $m$  here? (I can imagine, but some other reader probably won't)

## 5 Improving from algebraic fingerprinting

As mentioned in Section 4.1, algebraic fingerprinting as a framework for  $k$ -multilinear monomial detection has a lower bound of  $\Omega(2^k)$ . However, this framework is very generic since it approaches the abstracted multilinear monomial detection without utilizing the combinatorial properties specific to the underlying problem. Creative algebrizations of problems, though, have potential for these properties.

Indeed, faster algorithms for specific combinatorial problems have been found by designing new algebrizations with techniques similar to algebraic fingerprinting. In Section 5.1, we see how Björklund [Bjö14] exploited the cancelation due to characteristic to cancel non-solution monomials by clever design of algebrization.

For parameterized problems in general, Section 5.2 briefly overviews other ideas and improvements in the domain of  $k$ -multilinear monomial detection.

## 5.1 Fingerprinting for cancelation of non-solutions

In the algebraic framework by Koutis and Williams, characteristic 2 is essentially a by-product of implementing the idea of discarding squared variables, and fingerprints are merely introduced to prevent valid solutions from canceling out.

Björklund [Bjö14], however, approached the elimination of non-solutions from a different angle: instead of focusing on squared elements, Björklund focused on the coefficients, and used characteristic 2 to eliminate non-solutions by designing the algebrization such that non-solution monomials pair up in the polynomial. For Hamiltonicity, this approach allowed Björklund to utilize fast algorithms for computing matrix determinants, which resulted in an  $\mathcal{O}^*(1.657^n)$  time randomized algorithm.

Before we briefly overview the idea behind this algorithm, we define the relevant problem:

**HAMILTONIAN PATH (HAMILTONICITY)**

**Input:** An undirected graph  $G = (V, E)$ .

**Question:** Does  $G$  contain a simple path that visits every vertex?

Björklund approached this problem by considering *cycle covers*. A cycle cover in a directed graph  $D = (V, A)$  is a subset  $C \subseteq A$  such that for every  $v \in V$  there is exactly one arc  $a_1 \in A$  ending in  $v$  and one arc  $a_2 \in A$  starting from  $v$ . One can think of a cycle cover as a set of disjoint cycles that cover the entire graph (see Figure 3). A *Hamiltonian cycle cover* consists of a single cycle that covers every vertex. Note that a Hamiltonian cycle cover trivially implies a Hamiltonian path.

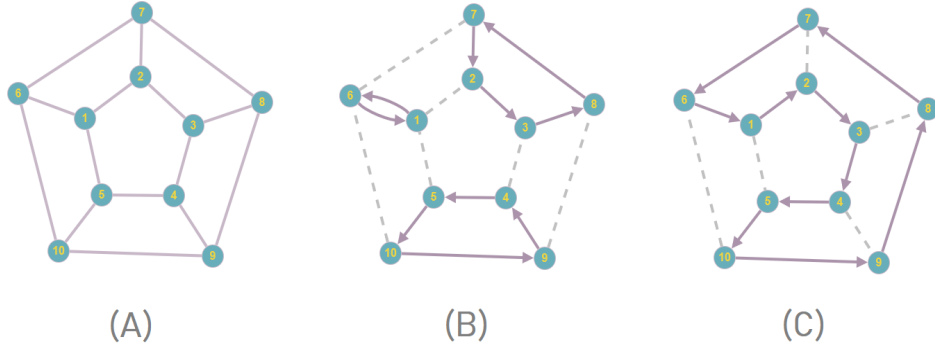


Figure 3: (A) is an undirected graph  $G$ . (B) showcases an example cycle cover for  $G$ . Note that an undirected graph can be trivially transformed into a directed graph. (C) is a Hamiltonian cycle cover.

In terms of algebrization, Björklund essentially formed a multivariate polynomial  $P$ , where the monomials are *labels* corresponding to cycle covers of the input graph. Evaluating  $P$  over a suitable ring, non-Hamiltonian cycle covers cancel out, and we detect only Hamiltonian cycle covers, i.e., valid solutions to the underlying problem.



Björklund achieved this by labeling cycle covers such that all non-Hamiltonian cycle covers receive a label identical to another non-Hamiltonian cycle cover, and Hamiltonian cycle covers receive unique labels. Thus, every non-solution monomial has a pair and therefore cancels out in characteristic 2, while valid solution monomials survive. The design of such labels requires, however, that  $G$  be undirected. Therefore, this approach can only solve Hamiltonicity for undirected graphs.

For detecting whether  $P$  is identical to zero (no solutions exist), Björklund employed the same idea of polynomial identity testing as in Section 3.1.4. For achieving the time complexity of  $\mathcal{O}^*(1.657^n)$ , Björklund utilized the natural interpretation of a sum of labeled cycle covers as a matrix permanent [Bjö14]. Moreover, the matrix permanent is identical to the determinant of the matrix over a ring of characteristic 2 [Bjö14], which essentially allowed Björklund to merely compute a matrix determinant to detect a Hamiltonian path.

## 5.2 Other research in detecting multilinear monomials

Since the runtime of algebraic fingerprinting is arguably optimized for the general multilinear monomial detection, much research has been done on specific settings. In their series of papers [CF11; Che+11; CF10; Che+13], Chen et al. studied complexities for different types of multivariate polynomials, and detecting different types of monomials, i.e.,  $q$ -monomials<sup>9</sup> which were further studied by Chen, Chen, and Yang [CCY14].

Much effort [Che13; CC13; Fom+14; Pra19; Bra19; BP21; Bra22] has been directed into deterministic algorithms for instances of multilinear monomial detection, and closing the gap between the deterministic and the randomized algorithms. The general deterministic  $k$ -multilinear monomial detection is solved in  $\mathcal{O}^*(3.84^k)$  time [Fom+14], though methods that result in much faster algorithms for specific problems have been found. For example, the state-of-the-art deterministic algorithm for  $k$ -path runs in  $\mathcal{O}^*(2.55^k)$  time by computing partial differentials of multivariate polynomials [BP21].

On another front, there has been research in extending the application domain of algebraic fingerprinting. In specific, research on how more information can be taken into the algebrization has found success. With *constrained multilinear monomial detection* [Kou12; BKK16], we detect only multilinear monomials that satisfy an additional constraint, e.g., a proper coloring<sup>10</sup>. This can be extended to detect solutions with a specific cost [BKK16] and/or temporal properties<sup>11</sup> [TGL20].

The evaluation of the polynomial in the algebraic fingerprinting framework is done

---

<sup>9</sup>a  $d$ -degree monomial where every exponent is smaller than  $d$

<sup>10</sup>the number of occurrences of each color tied to a variable in a multilinear monomial is below some maximum

<sup>11</sup>i.e., we can solve constrained problems that exist in graphs with timestamped and colored vertices

sequentially. However, the matrix representations of the group algebras used offer possibilities for parallelization. Ekanayake et al. [Eka+19] observed this and designed an optimized distributed algorithm for multilinear monomial detection. The authors showed applications of this method that outperform previously fastest algorithms for several network scan statistics problems.

## 6 Conclusion

From Section 1.4, it can be seen that a fast algorithm for  $k$ -multilinear monomial detection gives a fast algorithm for many parameterized combinatorial problems. We found that algebraic fingerprinting solves this problem by evaluating the given multivariate polynomial over a specific algebraic structure. Thus, we may say that algebraic fingerprinting gives a general framework for solving parameterized combinatorial problems.

In Section 3.1, we discussed the idea behind algebraic fingerprinting: generate a polynomial  $P(X)$  from the algebrization of a combinatorial problem, and evaluate  $P(X)$  over  $GF(2^l)[Z_2^k]$  with randomized assignments  $X \rightarrow GF(2^l)[Z_2^k]$  of form  $x_i \rightarrow (v_0 + v_i)$ , augmented with scalar multiplications by elements randomly chosen from  $GF(2^l) \setminus \{0\}$ . This gives a randomized algorithm for  $k$ -multilinear monomial detection that runs in  $\mathcal{O}^*(2^k)$  time.

As discussed in Section 5.2, algebraic fingerprints have been studied for deterministic algorithms as well; closing the gap between the runtimes of deterministic and randomized algorithms has seen great effort, and the gaps for several problems are still bound for improvements. Furthermore, Section 5.2 showcases possibilities for extending the application domain of algebraic fingerprinting. Applying the general algebraic framework to different problem areas is a significant field of interest.

As seen in Section 5.1, the technique of algebraic fingerprints can be utilized for even faster randomized algorithms for specific problems. In particular, a near fifty-year stagnation in improvements for the well-studied Hamiltonian path problem saw a significant development by Björklund right after the novel algebraic technique [Kou08] saw publication. This sparked developments for several other problems as well [Bjö+17], all of which were greatly inspired by the general technique of algebraic fingerprints and follow similar utilizations of fingerprints as in Hamiltonicity.

However, most of the problems in Section 1.4 are still solved most efficiently by the general algebraic framework. Solving these specific problems faster is still an open problem. It may be that finding faster algorithms that beat the general algebraic framework requires one to design different algebrizations (as in Hamiltonicity) or come up with completely new ideas.

## References

- [AYZ95] Noga Alon, Raphael Yuster, and Uri Zwick. "Color-Coding". In: *J. ACM* 42.4 (July 1995), pp. 844–856. ISSN: 0004-5411. DOI: 10.1145/210332.210337. URL: <https://doi.org/10.1145/210332.210337>.
- [Bel62] Richard Bellman. "Dynamic programming treatment of the travelling salesman problem". In: *Journal of the ACM (JACM)* 9.1 (1962), pp. 61–63.
- [Bjö14] Andreas Björklund. "Determinant Sums for Undirected Hamiltonicity". In: *SIAM Journal on Computing* 43.1 (2014), pp. 280–299. DOI: 10.1137/110839229. eprint: <https://doi.org/10.1137/110839229>. URL: <https://doi.org/10.1137/110839229>.
- [BKK16] Andreas Björklund, Petteri Kaski, and Lukasz Kowalik. "Constrained multilinear detection and generalized graph motifs". In: *Algorithmica* 74 (2016), pp. 947–967.
- [Bjö+17] Andreas Björklund et al. "Narrow sieves for parameterized paths and packings". English. In: *Journal of Computer and System Sciences* 87.C (2017), pp. 119–139. DOI: 10.1016/j.jcss.2017.03.003.
- [Bra19] Cornelius Brand. "Patching colors with tensors". In: *27th Annual European Symposium on Algorithms (ESA 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. 2019.
- [Bra22] Cornelius Brand. "A note on algebraic techniques for subgraph detection". In: *Information Processing Letters* 176 (2022), p. 106242.
- [BP21] Cornelius Brand and Kevin Pratt. "Parameterized Applications of Symbolic Differentiation of (Totally) Multilinear Polynomials". In: *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*. Ed. by Nikhil Bansal, Emanuela Merelli, and James Worrell. Vol. 198. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, 38:1–38:19. ISBN: 978-3-95977-195-5. DOI: 10.4230/LIPIcs.ICALP.2021.38. URL: <https://drops.dagstuhl.de/opus/volltexte/2021/14107>.
- [CEV17] Jose Cadena, Saliya Ekanayake, and Anil Vullikanti. "Fast graph scan statistics optimization using algebraic fingerprints". In: *2017 IEEE International Conference on Big Data (Big Data)*. IEEE. 2017, pp. 905–910.

- [Che+07] Jianer Chen et al. "Improved Algorithms for Path, Matching, and Packing Problems". In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '07. New Orleans, Louisiana: Society for Industrial and Applied Mathematics, 2007, pp. 298–307. ISBN: 9780898716245.
- [Che13] Shenshi Chen. "Monomial testing and applications". In: *Frontiers in Algorithmics and Algorithmic Aspects in Information and Management: Third Joint International Conference, FAW-AAIM 2013, Dalian, China, June 26-28, 2013. Proceedings*. Springer. 2013, pp. 106–117.
- [CCY14] Shenshi Chen, Yaqing Chen, and Quanhai Yang. "TOWARD RANDOMIZED TESTING OF  $q$ -MONOMIALS IN MULTIVARIATE POLYNOMIALS". In: *Discrete Mathematics, Algorithms and Applications* 6.02 (2014), p. 1450016.
- [CC13] Shenshi Chen and Zhixiang Chen. "Faster Deterministic Algorithms for Packing, Matching and  $t$ -Dominating Set Problems". In: *arXiv preprint arXiv:1306.3602* (2013).
- [CF10] Zhixiang Chen and Bin Fu. "Approximating multilinear monomial coefficients and maximum multilinear monomials in multivariate polynomials". In: *Combinatorial Optimization and Applications: 4th International Conference, COCOA 2010, Kailua-Kona, HI, USA, December 18-20, 2010, Proceedings, Part I* 4. Springer. 2010, pp. 309–323.
- [CF11] Zhixiang Chen and Bin Fu. "The complexity of testing monomials in multivariate polynomials". In: *Combinatorial Optimization and Applications: 5th International Conference, COCOA 2011, Zhangjiajie, China, August 4-6, 2011. Proceedings* 5. Springer. 2011, pp. 1–15.
- [Che+11] Zhixiang Chen et al. "Algorithms for testing monomials in multivariate polynomials". In: *Combinatorial Optimization and Applications: 5th International Conference, COCOA 2011, Zhangjiajie, China, August 4-6, 2011. Proceedings* 5. Springer. 2011, pp. 16–30.
- [Che+13] Zhixiang Chen et al. "On testing monomials in multivariate polynomials". In: *Theoretical Computer Science* 497 (2013), pp. 39–54.
- [Eka+19] Saliya Ekanayake et al. "MIDAS: Multilinear detection at scale". In: *J. Parallel Distributed Comput.* 132 (2019), pp. 363–382. DOI: 10.1016/j.jpdc.2019.04.006. URL: <https://doi.org/10.1016/j.jpdc.2019.04.006>.
- [Fom+14] Fedor V Fomin et al. "Representative sets of product families". In: *Algorithms-ESA 2014: 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings* 21. Springer. 2014, pp. 443–454.

- [HK62] Michael Held and Richard M. Karp. "A Dynamic Programming Approach to Sequencing Problems". In: *Journal of the Society for Industrial and Applied Mathematics* 10.1 (1962), pp. 196–210. DOI: 10.1137/0110015. eprint: <https://doi.org/10.1137/0110015>. URL: <https://doi.org/10.1137/0110015>.
- [KI03] Valentine Kabanets and Russell Impagliazzo. "Derandomizing polynomial identity tests means proving circuit lower bounds". In: *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*. 2003, pp. 355–364.
- [Kou05] Ioannis Koutis. "A faster parameterized algorithm for set packing". In: *Information Processing Letters* 94.1 (2005), pp. 7–9. ISSN: 0020-0190. DOI: <https://doi.org/10.1016/j.ipl.2004.12.005>. URL: <https://www.sciencedirect.com/science/article/pii/S0020019004003655>.
- [Kou08] Ioannis Koutis. "Faster Algebraic Algorithms for Path and Packing Problems". In: *Automata, Languages and Programming*. Ed. by Luca Aceto et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 575–586. ISBN: 978-3-540-70575-8.
- [Kou12] Ioannis Koutis. "Constrained multilinear detection for faster functional motif discovery". In: *Information Processing Letters* 112.22 (2012), pp. 889–892.
- [KW09] Ioannis Koutis and Ryan Williams. "Limits and Applications of Group Algebras for Parameterized Problems". In: *Automata, Languages and Programming*. Ed. by Susanne Albers et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 653–664. ISBN: 978-3-642-02927-1.
- [KW15] Ioannis Koutis and Ryan Williams. "Algebraic Fingerprints for Faster Algorithms". In: *Commun. ACM* 59.1 (Dec. 2015), pp. 98–105. ISSN: 0001-0782. DOI: 10.1145/2742544. URL: <https://doi.org/10.1145/2742544>.
- [Pra19] Kevin Pratt. "Waring rank, parameterized and exact algorithms". In: *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2019, pp. 806–823.
- [Ter99] Audrey Terras. *Fourier analysis on finite groups and applications*. 43. Cambridge University Press, 1999.
- [TGL20] Suhas Thejaswi, Aristides Gionis, and Juho Lauri. "Finding path motifs in large temporal graphs using algebraic fingerprints". In: *Big Data* 8.5 (2020), pp. 335–362.
- [Val92] Leslie G Valiant. "Why is Boolean complexity theory difficult". In: *Boolean Function Complexity* 169.84-94 (1992), p. 4.

- [Wil09] Ryan Williams. "Finding paths of length  $k$  in  $O(2^k)$  time". In: *Information Processing Letters* 109.6 (2009), pp. 315–318. ISSN: 0020-0190. DOI: <https://doi.org/10.1016/j.ipl.2008.11.004>. URL: <https://www.sciencedirect.com/science/article/pii/S0020019008003396>.