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

Detecting multilinear monomials with algebraic fingerprints

Bachelor's Thesis

May 10, 2023

Onni Miettinen

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

ABSTRACT OF
BACHELOR'S THESIS

Author:	Onni Miettinen
Title of thesis:	Detecting multilinear monomials with algebraic fingerprints
Date:	May 10, 2023
Pages:	30
Major:	Computer Science
Code:	SCI3027
Supervisor:	Prof. Eero Hyvönen
Instructor:	D.Sc. Augusto Modanese (Department of Computer Science)

This thesis studies how combinatorial problems can be solved by algebraic means. More specifically, the thesis focuses on the technique of algebraic fingerprinting by Koutis (ICALP 2008) and Williams (ICALP 2009). The thesis overviews the algebra behind the technique and discusses its limits as a general framework for combinatorial problems in parameterized complexity. Finally, the thesis collects some ideas for improving the framework or even surpassing its lower runtime boundary.

Combinatorial problems can be reduced to instances of an algebraic problem of detecting a multilinear monomial in a multivariate polynomial. With algebraic fingerprints, a multilinear monomial can be detected in time $\mathcal{O}(2^k \cdot \mathsf{poly}(n))$, where n is the number of variables in the multivariate k-degree polynomial. This gives a general framework for solving combinatorial problems in parameterized complexity. Currently, this technique underlies the fastest algorithms for many combinatorial problems.

Moreover, $\Omega(2^k \cdot \mathsf{poly}(n))$ is the lower limit of the technique: multilinear monomials cannot be detected faster with the general framework of algebraic fingerprints. However, this 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 may be found for a specific problem. Indeed, e.g. Björklund (FOCS 2010) found an algorithm that finds a Hamiltonian path in time $\mathcal{O}(1.657^n \cdot \mathsf{poly}(n))$ by using techniques from algebraic fingerprinting. Furthermore, it seems that understanding and making use of the ideas behind algebraic fingerprints is the key to improving from the general framework.

Keywords:	multivariate, multilinear, monomial, polynomial, algebraic, fingerprints, fingerprinting, parameterized, combinatorial, FPT
Language:	English

Aalto-yliopisto
Perustieteiden korkeakoulu
Teknistieteellinen kandidaattiohjelma

KANDIDAATINTYÖN
TIIVISTELMÄ

Tekijä:	Onni Miettinen
Työn nimi:	Detecting multilinear monomials with algebraic fingerprints
Päiväys:	10. toukokuuta 2023
Sivumäärä:	30
Pääaine:	Tietotekniikka
Koodi:	SCI3027
Vastuuopettaja:	Prof. Eero Hyvönen
Työn ohjaaja(t):	D.Sc. Augusto Modanese (Tietotekniikan laitos)

Tässä työssä tutkitaan, kuinka kombinatorisia ongelmia voidaan ratkaista algebrallisin keinoin. Kandidaatintyö keskittyy tarkemmin Koutisin (ICALP 2008) ja Williamsin (ICALP 2009) algebrallisia sormenjälkiä hyödyntävään tekniikkaan. Työ tutkii tekniikan taustalla olevaa algebraa ja sen mahdollisia rajoja, kun sitä tarkastellaan yleisenä kehyksenä kombinatoristen ongelmien ratkaisemiseksi parametrisoidussa kompleksisuudessa. Viimeiseksi työssä pohditaan keinoja, joilla tätä kehystä voidaan kehittää tai joilla voidaan ohittaa sen antama raja algoritmien tehokkuudelle.

Kombinatoriset ongelmat voidaan redusoida algebralliseen muotoon, jossa pyritään havaitsemaan multilineaarinen monomi monimuuttujaisesta polynomista. Algebrallisten sormenjälkien avulla multilineaarinen monomi voidaan havaita ajassa $\mathcal{O}(2^k \cdot \mathsf{poly}(n))$, jossa n on k-asteisen polynomin muuttujien lukumäärä. Tämä toimii yleisen kehyksenä, jolla kombinatorisia ongelmia voidaan ratkaista parametrisoidussa kompleksisuudessa. Tällä hetkellä algebrallisten sormenjälkien tekniikka pohjaa nopeimpia algoritmeja monille kombinatorisille ongelmille.

$\Omega(2^k \cdot \mathsf{poly}(n))$ on kuitenkin kehyksen alaraja: tekniikalla ei voida yleisesti havaita multilineaarisia monomeja nopeammin. Toisaalta kehys kohdistuu hyvin yleismuotoiseen multilineaaristen monomien havaitsemisongelmaan; jos hyödynnetään ongelmakohtaisia kombinatorisia ominaisuuksia, voidaan löytää algoritmi, joka ratkaisee ongelman nopeammin. Muun muassa Björklund (FOCS 2010) kehitti algebrallisten sormenjälkien antamien ideoiden avulla algoritmin, joka löytää Hamiltonisen polun ajassa $\mathcal{O}(1.657^n \cdot \mathsf{poly}(n))$. Näyttääkin siltä, että algebrallisten sormenjälkien syvä ymmärrys sekä ongelmakohtainen hyödyntäminen ovat avaimia nopeammille algoritmeille.

Avainsanat:	monimuuttuja, multilineaarinen, monomi, polynomi, algebrallinen, sormenjälki, parametrisoitu, kombinatorinen, FPT
Kieli:	Englanti

Contents

1 Introduction **5**

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

2 Preliminaries **10**

2.1 Algebra 10

2.2 Arithmetic circuits 13

3 General multilinear monomial detection **14**

3.1 Algebraic fingerprinting 14

3.1.1 Specifications for the algebraic structure 15

3.1.2 Using group algebras of $\mathbb{Z}_2^k$ 16

3.1.3 Fingerprints to prevent unwanted cancelation 17

3.2 Polynomial identity testing 19

3.3 Time and space complexity 20

4 Limits of algebraic fingerprinting **21**

4.1 Algebraic optimization 21

4.2 Derandomization 22

4.3 Finding the solution 22

5 Improving from algebraic fingerprinting **23**

5.1 Fingerprinting for cancelation of non-solutions 23

5.2 Other research in detecting multilinear monomials 25

6 Conclusion **26**

References **27**

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 (i.e., a term with multiple variables where each variable appears at most once, see also Section 2.1). This particular problem has proven to be fundamental since many combinatorial problems can be reduced to it. The idea of multilinear monomial detection for combinatorial problems in parameterized complexity was first introduced by Koutis in an early work [Kou05], where a set packing problem is reduced to multilinear monomial detection.

The technique of *algebraic fingeprinting*, introduced by Koutis [Kou08] and further developed by Williams [Wil09], is used for detecting multilinear monomials, and has found great success for many combinatorial problems. For instance with algebraic fingerprinting, the k-path problem that previously could be solved in $\mathcal{O}^*(4^k)$ time[1] 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 [Bjö10] 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 date from 1961 [Bel62; HK61]. This was a significant improvement on a well-studied problem that had seen no progress in nearly fifty years! Soon thereafter, an $\mathcal{O}^*(1.657^k)$ algorithm was found for the parameterized version of the problem, the k-path problem, by using similar ideas as with Hamiltonicity [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 new to this field of study, this thesis overviews a clever use of algebra in theoretical computer science, that is the technique of algebraic fingerprinting.

In the rest of this introduction, we discuss the relation of combinatorics and polynomials, build into the algebrization of combinatorial problems, and define the multilinear monomial detection problem as a prelude for the thesis. Finally in Section 1.4, we overview the area of problems where this technique proves useful.

[1] $\mathcal{O}^*$ may hide factors polynomial to input size.

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 with algebraic means. Section 5 briefly overviews ideas for improvement in the domain of 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 path (i.e., a path where each vertex is visited only once) 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 consists of edges that are joint but share a vertex with at most one other edge.

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 (i.e., an expression with only additions of products) as a set of combinations. For instance, imagine a set of elements $M = \{a, b, c\}$. 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 the polynomial

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

represents all possible combinations if we are to pick twice from M. Indeed, the 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 correspond to non-multilinear terms, and valid solutions to multilinear terms.

Note, however, that detecting a multilinear monomial is not trivial: we are not given the polynomial (the problem input) as a sum of monomials. Instead, we are given an *arithmetic circuit* [2] representing the polynomial, e.g., merely the expression C^2 in the example above. As we see in the next section, this is significant for the detection complexity.

Appropriate definitions for the indeterminates for M, i.e. proper algebrizations, are

[2] see Section 2.2 for information on arithmetic circuits

problem-specific. In what follows, we give an example reduction to multilinear monomial detection; we algebrize the k-3D matching problem which 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 $[\,n\,] = \{1, \ldots, n\}$.

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 the multivariate polynomial $P_k \in \mathbb{Z}[X]$ 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. We confirm by a simple example.

For an 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 1), we have $P_1 = a_1b_1c_1 + a_1b_2c_2 + a_2b_2c_3$. For $k = 2$, we get

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

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 1.

Thus, we have succesfully recuded a combinatorial problem to an algebraic one that is multilinear monomial detection. Note that in the scope of this thesis, we are indeed satisfied by merely *detecting* a solution, i.e., deciding whether the underlying problem has a solution (although we briefly overview actually finding a solution in Section 4.3).

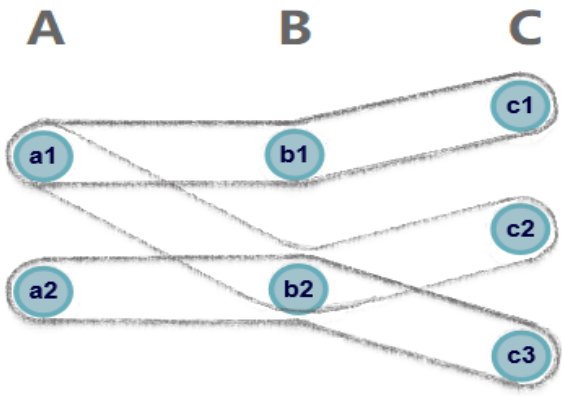

Figure 1: An example instance of k-3D matching, with inputs defined as above. The sets A, B and C are aligned as columns. The triples T are drawn to overlap these columns horizontally.

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? [3]

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 possible monomials [4], 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.

[3] a multilinear monomial with degree k has k indeterminates, see also Section 2.1

[4] e^N is an approximation from $\binom{n+N}{N}$, where n is the number of variables

However, the underlying problems are usually FPT (*fixed-parameter tractable*, i.e., solvable in time polynomial to n if k is fixed). Therefore, 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 $\rho\colon X \to Y$, 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 [5] $P_1 \in \mathbb{Z}[X]$ with every $x \in X$ replaced 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. Thus, for an algorithm to reliably detect a multilinear monomial, it is necessary to use different mappings from X to Y until a multilinear monomial survives the mapping. However, the probability that any given multilinear monomial survives a uniformly chosen random mapping is around $e^{-|Y|}$ [KW16]. 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. Moreover, it yields a general framework for solving combinatorial problems in parameterized complexity since the technique focuses on the abstract k-multilinear monomial detection, to which many combinatorial problems can be reduced to [KW16].

1.4 Related problems

The importance of multilinear monomial detection lies in the fact that many combinatorial problems, as we see in this section, can be reduced to instances of it. Thus, there is great interest in efficiently solving the general k-multilinear monomial detection problem. 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

[5] In algebraic terms, $P_1' = \Phi_\rho(P_1) \in \mathbb{Z}[Y]$ and Φ_ρ is the canonical extension of ρ to a ring homomorphism $\mathbb{Z}[X] \to \mathbb{Z}[Y]$.

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 and e.g. social network analysis. TODO: quickquik example of anomaly

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 algebraic structures and concepts. Section 2.2 gives a brief cover on 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. $\mathsf{poly}(n)$ refers to a polymial function in n. $\mathcal{O}^*$ hides factors polynomial to the input size, e.g., $\mathcal{O}(n^3k^n) = \mathcal{O}^*(k^n)$. Ditto for Ω^*.

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 these elements.

Groups. A *group* $\mathbf{G}$ is a tuple $(G, +)$, where G is a set of elements, $+\colon G \times G \to G$ 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 $\cdot\colon R \times R \to R$ 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: $a \cdot b = ab$. Furthermore, we use coefficients and exponents in $\mathbb{N}$ to denote repeated operations of addition and multiplication, respectively: $a + a = 2a$, $a \cdot a = a^2$.

$\mathbf{R}$ must contain a multiplicative identity $\mathbf{1} \in \mathbf{R}$ such that $\forall a \in \mathbf{R}\colon a \cdot \mathbf{1} = a$. 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.,

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

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

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

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

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

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

Throughout the thesis we may use language such as *cancelation 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: $\forall k \in \mathbb{N}$, $2kf = k(2f) = k\mathbf{0} = \mathbf{0}$

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 $\kappa\colon \mathbf{A} \to \mathbf{B}$ is a bijective mapping that satisfies $\forall u, v \in \mathbf{A}\colon \kappa(uv) = \kappa(u)\kappa(v) \land \kappa(u+v) = \kappa(u)+\kappa(v)$. Thus, if we only care for the relations between the elements, we may say that two isomorphic algebraic structures (i.e., there exists an isomorphism between them) 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. 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 $\forall i \in \{1, \ldots, n\} \colon a_i = 0 \vee a_i = 1$. A k-multilinear monomial has k indeterminates. 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 $c \in \mathbf{K}$:

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

If a polynomial is in such form, we may say that it is in *sum of monomials form*.

Alternatively, a polynomial ring $\mathbf{K}[X]$ can be thought of as a vector space over $\mathbf{K}$ of infinite dimension with the set of all monomials in X 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,

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

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

Evaluation of a polynomial P at Z is denoted $P(Z)$, where Z is a map $X \to \mathbf{R}$ and $\mathbf{R}$ is a ring containing $\mathbf{K}$. Evaluating at Z, i.e., assigning indeterminates X to R, 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 for every mapping $W \colon X \to \mathbf{R}$ $P(W) = \mathbf{0}$.

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 consider *addition* and *multiplication* gates. These gates correspond to the binary operations of the underlying algebraic structure 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, the arithmetic circuit presented in Figure 2 gives an abstract evaluation of a polynomial $P \in \mathbb{Z}[a, b, c, d]$ of the form $P = (ab + cd)^2$. 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. Note that in Section 3.1.3, we augment a circuit with *scalar multiplication*, though we denote them with standard multiplication gates.

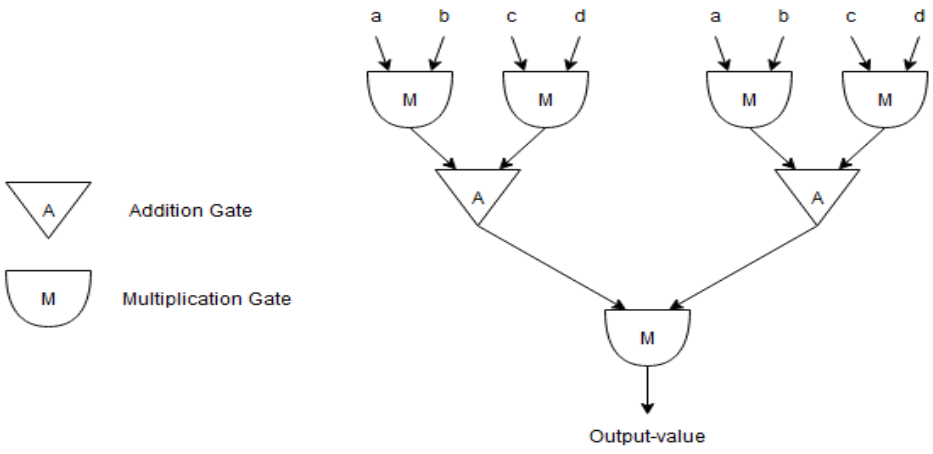

Figure 2: A circuit for evaluating $(ab + cd)^2$. Note that the input terminals have been copied for clarity.

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 the 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; KW09; Wil09] for the k-multilinear monomial detection. In Section 3.1, we dive into the technique of *algebraic fingerprinting*, and see the ideas and implementation of Koutis [Kou08]. Section 3.2 covers how Williams [Wil09] developed this idea in the perspective of polynomial identity testing. In Section 3.3 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.

$$\forall x \in X : x^2 = \mathbf{0} \tag{1}$$

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^2q' for some $x \in X$, and $x^2q' = \mathbf{0}q' = \mathbf{0}$ 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* [KW16]: 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 chosen mapping $Z\colon X \to \mathbf{G}$ where $w \neq \mathbf{0}$,

$$P(Z) = \begin{cases} \mathbf{0} & \text{if no multilinear monomials exist} \\ w & \text{otherwise} \end{cases} \tag{2}$$

In the following subsections, we specify an appropriate algebraic structure such that (2) is met, as well as some requirements for efficiency. Then, we discuss the works of Koutis [Kou08], and see how these specifications were implemented.

This thesis refers to the general framework provided by this algebraic technique for parameterized problems [KW09; KW16] 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 (1) and thus the first equality in (2). We specify for commutative algebraic structures such as commutative rings for multilinear monomial detection; $\mathbf{G}$ should have commutative multiplication in order for (1) to be effective. The ordering of indeterminates in monomials is generally unknown since we have abstracted away the specific algebrization into multilinear monomial detection.

For the second equality in (2), 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 $\gamma\colon X \to \mathbf{G}$ with a constant positive 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 $\gamma\colon X \to \mathbf{G}$. However, if we specify for a constant probability of survival under random γ, we can reliably decide whether a multilinear monomial exists by evaluating $P(X)$ in $\mathbf{G}$ over a constant number of randomized assignments $X \to \mathbf{G}$. 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:

- $\forall g \in \mathbf{G}\colon g^2 = \mathbf{0}$
- 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 $\gamma\colon X \to \mathbf{G}$ with constant positive probability.

In the work that introduced this algebraic fingerprinting technique [Kou08], Koutis used the group algebra $\mathbb{Z}_2[\mathbb{Z}_2^k]$. Williams [Wil09] developed the technique further, generalizing the algebra to $GF(2^l)[\mathbb{Z}_2^k]$. Next, we look at these group algebras of $\mathbb{Z}_2^k$ for $\mathbf{G}$, although the ideas of Williams are discussed only later in Section 3.2.

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:

$$\forall z \in \mathbb{Z}_2^k\colon z^2 = \mathbf{0}. \tag{3}$$

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 mapping as γ, and the resulting element from the polynomial evaluation as $P(\gamma) \in \mathbf{F}[\mathbb{Z}_2^k]$. Koutis observed that due to (3), for all $v \in \mathbb{Z}_2^k$ and $(v_0 + v) \in \mathbf{F}[\mathbb{Z}_2^k]$,

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

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

However, the second equation of (2) does not necessarily hold, and it may be that $w = \mathbf{0}$. A multilinear monomial may evaluate to zero if two variables are assigned the same value from $\mathbf{F}[\mathbb{Z}_2^k]$. This itself is not too concerning here: a monomial $X_1 \cdots X_k$ assigned to be the product $(v_0 + v_1) \cdots (v_0 + v_k)$ survives this uniform random assignment, i.e. evaluates to non-zero, with probability at least $1/4$ [Kou08].

Since $\mathbf{F}$ has characteristic 2 however, multilinear monomials may still cancel each other out in $\mathbf{F}[Z_2^k]$ if they have even leading coefficients in $P(X)$. In such case, multilinear monomials do not survive under *any* assignment $X \to \mathbf{F}[\mathbb{Z}_2^k]$. 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 monomial in P_2 has an even coefficient, and thus would cancel out in $\mathbf{F}[Z_2^k]$ due to characteristic.

$$\begin{aligned} P_2 &= (a_1 b_1 c_1 + a_1 b_2 c_2 + a_2 b_2 c_3)^2 \\ &= a_1^2 b_1^2 c_1^2 + a_1^2 b_2^2 c_2^2 + a_2^2 b_2^2 c_3^2 + 2a_1^2 b_1 b_2 c_1 c_2 + 2a_1 a_2 b_1 b_2 c_1 c_3 + 2a_1 a_2 b_2^2 c_2 c_3. \end{aligned}$$

In general, when k is even, monomials in P_k will have even coefficients.

To tackle this issue, one idea is to add auxiliary indeterminates, called *fingerprints* [KW16], to the monomials in order to make them unique. For example, let $S = \{s_1, \ldots, s_6\}$ be the

set of an appropriate number of fingerprints and augment them to P_2 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 $x \in X \cup S$ to $\mathbf{F}[\mathbb{Z}_2^k]$. 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 monomials. Therefore, it increases the probability that indeterminates in a multilinear monomial are assigned the same value from $\mathbf{F}[Z_2^k]$, which results in the multilinear monomial evaluating to zero with higher probability. This can be countered by increasing the number of elements to assign to, i.e., increasing the number of distinct elements of form $(v_0 + v_i) \in \mathbf{F}[Z_2^k]$. 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 3.3).

Koutis approached this problem by assigning fingerprints to elements from a different algebraic structure: he used $S \to \mathbb{Z}_2$ 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 $S \to \mathbb{Z}_2$, 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 3): 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 (scalar) 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.

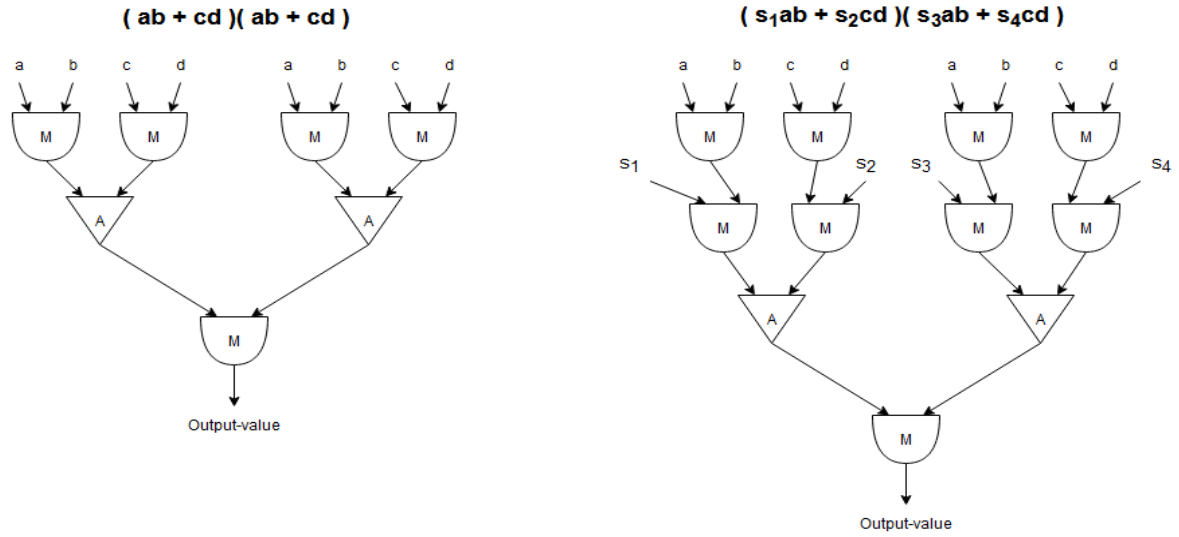

Figure 3: A fingerprinted circuit for evaluating $(ab + cd)^2$. Note the added terminals on the right circuit.

3.2 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 $\overline{X} \colon X \to \mathbf{F}[\mathbb{Z}_2^k]$ 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 not identical to $\mathbf{0}$, i.e., with Φ representing the family of mappings $S \to \mathbf{F}$,

$$\exists \phi \in \Phi \colon P'(\overline{X}, \phi) \neq \mathbf{0}.$$

Therefore, we formulate algebraic fingerprinting as polynomial identity testing, i.e., we test whether $P'(\overline{X}, S) \in \mathbf{K}[S]$ 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. We note the family of assignments $S \to \mathbf{F}$ as Φ. A multilinear monomial is detected if $\exists f \in \Phi \colon Q(f) \neq \mathbf{0}_F \in \mathbf{F}$, where $\mathbf{F}$ is a field of characteristic 2. Polynomial identity testing is a well-studied problem, and can be efficiently solved with random test evaluations by the *Schwartz-Zippel Lemma* [Sax09]. Essentially, the lemma gives an upper bound for the probability of false-negatives and, subsequently, the number of necessary random test evaluations for a satisfactory result on polynomial identity.

Schwartz-Zippel Lemma. *Let $P \in F[X]$ be a non-zero polynomial of degree d over a field F. Let Z be a uniformly chosen random mapping $Z \colon X \to F$. Then,*

$$\Pr[P(Z) = \mathbf{0}] \leq \frac{d}{|F|}.$$

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 $S \to \mathbb{Z}_2$ [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 Φ is much larger than the number of assignments $\phi \in \Phi$ that have $Q(\phi) = \mathbf{0}_F$, $Q(\delta) \neq \mathbf{0}_F$ with a high probability over an arbitrary $\delta \in \Phi$ [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, $Q(S)$ evaluates to $\mathbf{0}_F$ over a random assignment $S \to GF(2^l)$ with probability at most $k/2^l$. With the k-path problem, for instance, Williams used $l = 3 + \log_2 k$, which sets the probability of a false-negative evaluation to at most $1/2^3$. Note that unlike with $\mathbb{Z}_2$, we do not need the additive identity here, and we can assign fingerprints to values of $GF(2^l) \setminus \{\mathbf{0}\}$.

For k-path, 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. Furthermore, Williams generalized the technique by introducing the parameter l for the Galois field.

3.3 Time and space complexity

In the previous sections, we discussed the specifications for an algebraic stucture to implement algebraic fingerprinting. For an appropriate algebraic structure, 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.

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 $\kappa\colon \mathbb{Z}[\mathbb{Z}_2^k] \to \mathcal{M}$, where $\mathcal{M} = \mathbf{M}^{2^k \times 2^k}$ is an algebra of special permutation matrices with binary entries. To translate $\mathbb{Z}[\mathbb{Z}_2^k]$ to $\mathbb{Z}_2[\mathbb{Z}_2^k]$, it is enough to take the modulo 2 of the coefficients a_v in the elements of $\mathbb{Z}[\mathbb{Z}_2^k]$:

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

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 (2) is valid through κ.

Computing on these matrices, Koutis [Kou08] detected k-multilinear monomials 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. 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 $\mathcal{O}^*(2^k)$ time and $\mathcal{O}(\mathsf{poly}(n,k))$ space algorithm for detecting k-degree multilinear monomials in $P(X)$, where $n = |X|$ [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 algebrization of said parameterized problem. For instance with 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 3.3, it was established that k-multilinear monomials in $P(X)$ are detected with algebraic fingerprinting in $\mathcal{O}^*(2^k)$ time. This time was achieved by evaluating the circuit over matrix representations of the group 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

algebraic structure $\mathcal{G}$ 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 and the runtime of $\mathcal{O}^*(2^k)$ are in some sense optimal, and cannot be improved upon. 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 in the near future.

Thus, finding a deterministic algorithm probably requires a different approach from polynomial identity testing. Although this is mostly outside the scope of this thesis, in Section 5.2 we briefly return to this discussion.

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. In algebraic fingerprinting, however, 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, while constructing such a decision algorithm is generally hard for many combinatorial problems (e.g. problems in Section 1.4), finding the solution using such an algorithm (i.e., constructing a *search* algorithm given an oracle that decides the existence of a solution) is often easy. For example, it is widely known that decision and search algorithms for NP-complete problems are essentially equivalent in complexities [BG94]. Moreover, since most of the problems in Section 1.4 have search problems in FPT, it is

known that a solution can be found in polynomial time when its existence can be detected efficiently.

For example in [Kou08], Koutis 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, where n is the number of vertices and m the number of edges in the input graph. 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.

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, though, have potential for exploiting these properties.

Indeed, faster algorithms for specific combinatorial problems have been found by designing new algebrizations utilizing techniques similar to algebraic fingerprinting. In Section 5.1, we see how Björklund [Bjö10] designed a clever algebrization that exploits the cancelation due to characteristic to cancel non-solution monomials. For combinatorial 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 [Bjö10], 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 $a_1 \in C$ ending in v and one arc $a_2 \in C$ starting from v. One can think of a cycle cover as a set of disjoint cycles that cover the entire graph (see Figure 4). A *Hamiltonian cycle cover* consists of a single cycle that covers every vertex. Note that a Hamiltonian cycle cover trivially implies a Hamiltonian path.

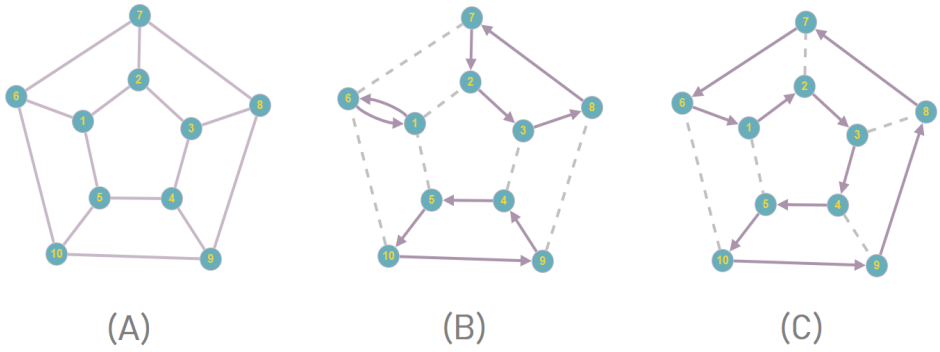

Figure 4: (A) is an undirected graph G. (B) showcases an example cycle cover for G. Note that an undirected graph can be trivially transformed into a directed graph. (C) is a Hamiltonian cycle cover.

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 3.2. 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 [Bjö10]. (In simple terms, the matrix permanent is the matrix determinant where every subtraction is replaced with addition.) While the permanent is generally very hard to compute, and the determinant is significantly easier,

they are identical in characteristic 2: there is no distinction between subtraction and addition. This essentially allowed Björklund to merely compute a matrix determinant to detect a Hamiltonian path, and surpass the $\mathcal{O}^*(2^n)$ bound that is given by the general framework.

5.2 Other research in detecting multilinear monomials

Since the algebraic fingerprinting technique is optimal for the *general* multilinear monomial detection, much research has been done on specific settings. There is a series of papers [CF10; CF11; Che+11; Che+13] studying complexities for different types of multivariate polynomials and detecting different types of monomials, e.g. *q-monomials*. Detecting q-monomials, i.e. d-degree monomials where every exponent is smaller than d, was also further studied by Chen, Chen, and Yang [CCY14].

As mentioned in Section 4.2, algebraic fingerprinting relies on randomized methods which are hard to derandomize. Much effort [Bra19; Bra22; BP21; Che13; CC13; Fom+17; Pra19] 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 in $\mathcal{O}^*(3.84^k)$ time [Fom+17], 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. Specifically, one line of research that has been particularly fruitful is *constrained multilinear monomial detection*. With constrained multilinear monomial detection [BKK16; Kou12], we detect only multilinear monomials that satisfy an additional constraint, e.g., a *proper coloring*. In coloring, we essentially tie colors to variables. A multilinear monomial then satisfies proper coloring if the number of occurrences of each color is below some maximum. This can be extended to detect solutions with a specific cost [BKK16] and/or temporal properties (i.e., we can solve constrained problems that exist in graphs with weights and timestamped vertices) [TGL20].

The evaluation of the polynomial is done sequentially in the algebraic fingerprinting framework. However, the matrix representations of the group algebras offer possibilities for parallelization. Ekanayake et al. [Eka+19] observed this and optimized distributed algorithms for multilinear monomial detection. Using these algorithms, the authors showed applications that outperform the previously fastest parallel algorithms in e.g. graph scan statistics (which is a methodology that detects anomalies in graphs, see Section 1.4).

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 combinatorial problems in parameterized complexity. 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 these 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 $X \to GF(2^l)[Z_2^k]$ of form $x_i \to (v_0 + v_i)$, augmented with scalar multiplications by elements randomly chosen from $GF(2^l) \setminus \{\mathbf{0}\}$. This yields a randomized algorithm for k-multilinear monomial detection that runs in $\mathcal{O}^*(2^k)$ time.

As mentioned 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 possibilites 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, the work of Björklund, which came right after the novel algebraic technique of [Kou08], ended a stagnation of nearly fifty years for the well-studied Hamiltonian path problem. 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 or proving that no improvement is possible 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 (1995), pp. 844–856. DOI: 10.1145/210332.210337.

[BG94] Mihir Bellare and Shafi Goldwasser. "The Complexity of Decision Versus Search". In: *SIAM J. Comput.* 23.1 (1994), pp. 97–119. DOI: 10.1137/S0097539792228289.

[Bel62] Richard Bellman. "Dynamic Programming Treatment of the Travelling Salesman Problem". In: *J. ACM* 9.1 (1962), pp. 61–63. DOI: 10.1145/321105.321111.

[Bjö10] Andreas Björklund. "Determinant Sums for Undirected Hamiltonicity". In: *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*. IEEE Computer Society, 2010, pp. 173–182. DOI: 10.1109/FOCS.2010.24.

[Bjö+17] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. "Narrow sieves for parameterized paths and packings". In: *J. Comput. Syst. Sci.* 87 (2017), pp. 119–139. DOI: 10.1016/j.jcss.2017.03.003.

[BKK16] Andreas Björklund, Petteri Kaski, and Lukasz Kowalik. "Constrained Multilinear Detection and Generalized Graph Motifs". In: *Algorithmica* 74.2 (2016), pp. 947–967. DOI: 10.1007/s00453-015-9981-1.

[Bra19] Cornelius Brand. "Patching Colors with Tensors". In: *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany*. Ed. by Michael A. Bender, Ola Svensson, and Grzegorz Herman. Vol. 144. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, pp. 25:1–25:16. DOI: 10.4230/LIPIcs.ESA.2019.25.

[Bra22] Cornelius Brand. "A note on algebraic techniques for subgraph detection". In: *Inf. Process. Lett.* 176 (2022). DOI: 10.1016/j.ipl.2021.106242.

[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, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*. Ed. by Nikhil Bansal, Emanuela Merelli, and James Worrell. Vol. 198. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, pp. 38.1–38.19. DOI: 10.4230/LIPIcs.ICALP.2021.38.

[CEV17] Jose Cadena, Saliya Ekanayake, and Anil Vullikanti. "Fast graph scan statistics optimization using algebraic fingerprints". In: *2017 IEEE International Conference on Big Data (IEEE BigData 2017), Boston, MA, USA, December 11-14, 2017*. Ed. by Jian-Yun Nie, Zoran Obradovic, Toyotaro Suzumura, Rumi Ghosh, Raghunath Nambiar, Chonggang Wang, Hui Zang, Ricardo Baeza-Yates, Xiaohua Hu, Jeremy Kepner, Alfredo Cuzzocrea, Jian Tang, and Masashi Toyoda. IEEE Computer Society, 2017, pp. 905–910. DOI: 10.1109/BigData.2017.8258007.

[Che+07] Jianer Chen, Songjian Lu, Sing-Hoi Sze, and Fenghui Zhang. "Improved algorithms for path, matching, and packing problems". In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*. Ed. by Nikhil Bansal, Kirk Pruhs, and Clifford Stein. SIAM, 2007, pp. 298–307. URL: http://dl.acm.org/citation.cfm?id=1283383.1283415.

[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*. Ed. by Michael R. Fellows, Xuehou Tan, and Binhai Zhu. Vol. 7924. Lecture Notes in Computer Science. Springer, 2013, pp. 106–117. DOI: 10.1007/978-3-642-38756-2_13.

[CCY14] Shenshi Chen, Yaqing Chen, and Quanhai Yang. "Toward Randomized Testing of Q-Monomials in multivariate polynomials". In: *Discret. Math. Algorithms Appl.* 6.2 (2014). DOI: 10.1142/S1793830914500165.

[CC13] Shenshi Chen and Zhixiang Chen. "Faster Deterministic Algorithms for Packing, Matching and t-Dominating Set Problems". In: *CoRR* abs/1306.3602 (2013). arXiv: 1306.3602.

[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*. Ed. by Weili Wu and Ovidiu Daescu. Vol. 6508. Lecture Notes in Computer Science. Springer, 2010, pp. 309–323. DOI: 10.1007/978-3-642-17458-2_26.

[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*. Ed. by Weifan Wang, Xuding Zhu, and Ding-Zhu Du.

Vol. 6831. Lecture Notes in Computer Science. Springer, 2011, pp. 1–15. DOI: 10.1007/978-3-642-22616-8_1.

[Che+11] Zhixiang Chen, Bin Fu, Yang Liu, and Robert T. Schweller. "Algorithms for Testing Monomials in Multivariate Polynomials". In: *Combinatorial Optimization and Applications - 5th International Conference, COCOA 2011, Zhangjiajie, China, August 4-6, 2011. Proceedings*. Ed. by Weifan Wang, Xuding Zhu, and Ding-Zhu Du. Vol. 6831. Lecture Notes in Computer Science. Springer, 2011, pp. 16–30. DOI: 10.1007/978-3-642-22616-8_2.

[Che+13] Zhixiang Chen, Bin Fu, Yang Liu, and Robert T. Schweller. "On testing monomials in multivariate polynomials". In: *Theor. Comput. Sci.* 497 (2013), pp. 39–54. DOI: 10.1016/j.tcs.2012.03.038.

[Eka+19] Saliya Ekanayake, Jose Cadena, Udayanga Wickramasinghe, and Anil Vullikanti. "MIDAS: Multilinear detection at scale". In: *J. Parallel Distributed Comput.* 132 (2019), pp. 363–382. DOI: 10.1016/j.jpdc.2019.04.006.

[Fom+17] Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. "Representative Families of Product Families". In: *ACM Trans. Algorithms* 13.3 (2017), pp. 36:1–36:29. DOI: 10.1145/3039243.

[HK61] Michael Held and Richard M. Karp. "A dynamic programming approach to sequencing problems". In: *Proceedings of the 16th ACM national meeting, ACM 1961, USA*. Ed. by Thomas C. Rowan. ACM, 1961, pp. 71:201–71:204. DOI: 10.1145/800029.808532.

[KI03] Valentine Kabanets and Russell Impagliazzo. "Derandomizing polynomial identity tests means proving circuit lower bounds". In: *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*. Ed. by Lawrence L. Larmore and Michel X. Goemans. ACM, 2003, pp. 355–364. DOI: 10.1145/780542.780595.

[Kou05] Ioannis Koutis. "A faster parameterized algorithm for set packing". In: *Inf. Process. Lett.* 94.1 (2005), pp. 7–9. DOI: 10.1016/j.ipl.2004.12.005.

[Kou08] 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*. Ed. by Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz. Vol. 5125. Lecture Notes in Computer Science. Springer, 2008, pp. 575–586. DOI: 10.1007/978-3-540-70575-8_47.

[Kou12] Ioannis Koutis. "Constrained multilinear detection for faster functional motif discovery". In: *Inf. Process. Lett.* 112.22 (2012), pp. 889–892. DOI: 10.1016/j.ipl.2012.08.008.

[KW09] Ioannis Koutis and Ryan Williams. "Limits and Applications of Group Algebras for Parameterized Problems". In: *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*. Ed. by Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas. Vol. 5555. Lecture Notes in Computer Science. Springer, 2009, pp. 653–664. DOI: 10.1007/978-3-642-02927-1_54.

[KW16] Ioannis Koutis and Ryan Williams. "Algebraic fingerprints for faster algorithms". In: *Commun. ACM* 59.1 (2016), pp. 98–105. DOI: 10.1145/2742544.

[Pra19] Kevin Pratt. "Waring Rank, Parameterized and Exact Algorithms". In: *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*. Ed. by David Zuckerman. IEEE Computer Society, 2019, pp. 806–823. DOI: 10.1109/FOCS.2019.00053.

[Sax09] Nitin Saxena. "Progress on Polynomial Identity Testing". In: *Bull. EATCS* 99 (2009), pp. 49–79.

[Ter99] Audrey Terras. *Fourier analysis on finite groups and applications*. London Mathematical Society Student Texts 43. Cambridge, UK: 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. DOI: 10.1089/big.2020.0078.

[Val92] Leslie G. Valiant. "Why is Boolean Complexity Theory so Difficult?" In: *Boolean Function Complexity*. Ed. by M. S.Editor Paterson. London Mathematical Society Lecture Note Series 169. Cambridge University Press, 1992, pp. 84–94. DOI: 10.1017/CBO9780511526633.008.

[Wil09] Ryan Williams. "Finding paths of length k in $O^*(2^k)$ time". In: *Inf. Process. Lett.* 109.6 (2009), pp. 315–318. DOI: 10.1016/j.ipl.2008.11.004.