Paths and Walks, Forests and Planes

Arcadian Algorithms and Complexity

A dissertation submitted towards the degree Doctor of Natural Sciences (Dr. rer. nat.) of the Faculty of Mathematics and Computer Science of Saarland University

by

Cornelius Brand

Saarbrücken
2019

Colloquium Information
Date: September 26, 2019
Dean: Prof. Sebastian Hack
Examination Board: Prof. Wolfang J. Paul (Chair),
Balagopal Komarath (Assistant),
Profes. Holger Dell, Markus Bläser, Petteri Kaski

Dem Andenken meiner Großmutter Elisabeth gewidmet.

Abstract

This dissertation is concerned with new results in the area of parameterized algorithms and complexity.

We develop a new technique for hard graph problems that generalizes and unifies established methods such as Color-Coding, representative families, labelled walks and algebraic fingerprinting. At the heart of the approach lies an algebraic formulation of the problems, which is effected by means of a suitable exterior algebra.

This allows us to estimate the number of simple paths of given length in directed graphs faster than before. Additionally, we give fast deterministic algorithms for finding paths of given length if the input graph contains only few of such paths. Moreover, we develop faster deterministic algorithms to find spanning trees with few leaves. We also consider the algebraic foundations of our new method.

Additionally, we investigate the fine-grained complexity of determining the precise number of forests with a given number of edges in a given undirected graph. To wit, this happens in two ways. Firstly, we complete the complexity classification of the Tutte plane, assuming the exponential time hypothesis. Secondly, we prove that counting forests with a given number of edges is at least as hard as counting cliques of a given size.

Abriß

Diese Dissertation befasst sich mit neuen Ergebnissen auf dem Gebiet parametrisierter Algorithmen und Komplexitätstheorie.

Wir entwickeln eine neue Technik für schwere Graphprobleme, die etablierte Methoden wie *Color-Coding*, *representative families*, *labelled walks* oder *algebraic fingerprinting* verallgemeinert und vereinheitlicht. Kern der Herangehensweise ist eine algebraische Formulierung der Probleme, die vermittels passender Graßmannalgebren geschieht.

Das erlaubt uns, die Anzahl einfacher Pfade gegebener Länge in gerichteten Graphen schneller als bisher zu schätzen. Außerdem geben wir schnelle deterministische Verfahren an, Pfade gegebener Länge zu finden, falls der Eingabegraph nur wenige solche Pfade enthält. Übrigens entwickeln wir schnellere deterministische Algorithmen, um Spannbäume mit wenigen Blättern zu finden. Wir studieren außerdem die algebraischen Grundlagen unserer neuen Methode.

Weiters untersuchen wir die *fine-grained*-Komplexität davon, die genaue Anzahl von Wäldern einer gegebenen Kantenzahl in einem gegebenen ungerichteten Graphen zu bestimmen. Und zwar erfolgt das auf zwei verschiedene Arten. Erstens vervollständigen wir die Komplexitätsklassifizierung der Tutte-Ebene unter Annahme der Exponentialzeithypothese. Zweitens beweisen wir, dass Wälder mit gegebener Kantenzahl zu zählen, wenigstens so schwer ist, wie Cliquen gegebener Größe zu zählen.

Acknowledgements

It goes without saying that I am above all indebted to my *Doktorvater*, Holger Dell. He allowed me to pursue all of my academic interests over the course of the last three years, with freedom when I wanted it, and guidance when I needed it. The amount of support I received from him, from the first day onwards, is second to none. I also thank Markus Bläser for making me acquainted with the theory group in Saarbrücken, and for providing me with support and advice whenever I asked for it. I wish to express my sincere gratitude also to Franz J. Brandenburg at the Universität Passau, whose copy of Papadimitriou's *Computational Complexity* made me encounter this wonderful field for the first time, and whose support and trust in my abilities well made up for my own lack thereof at times. I am grateful to Thore Husfeldt for the ideas, academic and non-academic alike, he contributed, and for hosting me in Copenhagen. Thanks also to Petteri Kaski for serving as external referee for this thesis.

On a more personal note, I thank my mother and father for being my parents (and everything that goes with it), which sounds much more trivial than it is, and my brother, Christopher. He has been a role model for me since childhood. Speaking of childhood, his and his wife Marina's rather recent gift to the world, Antonius, I thank for helping me distinguish between what is meaningful and what is not. Of course, I want to thank *Vanessa*—only partially in line with what Jonathan Swift had in mind with this name—for allowing me to take my mind off mathematics, and instead enjoy the few even more pleasant things in life. I also thank Raphael for the homely exclave we established here together, as well as the rest of the lot, who—for their own sake—shall not be named: C.B., S.B., J.E., A.L., H.S.

Lastly, I want to thank my late grandmother Elisabeth, to whom this thesis is dedicated, for providing me with the genetic material necessary to pursue an academic career in theoretical computer science, and for everything else.

I mention honorably:

1. The German taxpayer, who—with his generous, continued and involuntary financial support—enabled me to pursue my research interests as a surrogate activity.
2. The city of Saarbrücken for its[1] tireless efforts to provide me with ample motivation to take no more time than necessary.

[1] Cities are often endowed with female traits by their admirers.

Contents

Preface

Related publications. This thesis compiles the results of several articles that I authored or coauthored: The first part contains all results from a joint article with Holger Dell and Thore Husfeldt [BDH18], to which I made central technical and conceptual contributions, as well as some results that will appear in [Bra19]. The second part consists of expository material of known, but scattered results from the literature, as well as unpublished, new results. These will also appear in [Bra19]. The third part contains those parts of the joint articles with Marc Roth [BR17] and Holger Dell and Marc Roth [BDR16; BDR19] to which I have contributed significantly. Note that [BDR19] is the journal version of [BDR16]. My joint work with Michael Sagraloff [BS16] is not included in this thesis.

From now on, *we* will use the grammatical first person in the plural form for referring to the one author of this thesis.

Organization. The three parts of the thesis are not entirely independent of each other, but after reading the preliminaries, each one of them should at least be understandable without having read the others.

The first part of the thesis is concerned with developing faster algorithms for selected problems on graphs. In particular, we study the longest path problem, and provide new and sometimes improved algorithms for different settings in which the problem can be considered. Other improved algorithms are given for the problem of finding spanning trees with few inner nodes, and finding colorful matchings.

The second part is about questions in computational algebra that arise from the results of the first part. In particular, we give an account of the state-of-the-art for computing with Clifford algebras, as well as the exterior algebra. This is non-trivial because these results are

scattered across the literature in sometimes quite inaccessible ways. Additionally, we present new structural results about a related algebraic object.

The third and final part of the thesis is concerned with questions in fine-grained counting complexity, both in the exponential-time and the parameterized setting. We consider the complexity of counting the number[2] of forests in a graph, and show that this is a hard problem, under two different notions of hardness.

[2]It has been remarked frequently that one can either *compute the number of* objects, or *count* them, but we use this variant deliberately.

1. Preliminary Material

> *We won't be easy in our minds until everything has been said once and for all, then we'll fall silent and we'll no longer be afraid of keeping still. That will be the day.*
>
> Louis-Ferdinand Céline, *Journey to the End of the Night*

In this short chapter, we will agree on notation and standard assumptions in all basic matters that do not belong specifically to any one part of the thesis.

We introduce the following notions not necessarily to make the reader acquainted with them, but to remove any ambiguity in those cases where the exact properties of the defined objects—such as whether a ring always has a unity or not (it does), or whether graphs are simple by default (they are)—are generally not agreed upon in the literature.

1.1. Algorithms and Complexity

Landau Symbols. The symbols $o(\cdot), \omega(\cdot), O(\cdot), \Omega(\cdot)$ and $\Theta(\cdot)$ we use as usual, with a small exception: To emphasize the presence of the respective underlying partial orders on functions that these symbols induce (save for the case of Θ), we will use the symbols "$<$" and "$\leq$" as a compromise between the widespread, but blatantly false "$=$" and the correct, but blatantly pedantic "$\in$." For instance, we shall come across expressions such as $\omega(1) < n \leq O(n^2)$ or $\Omega(1) \leq 561 < o(n)$.

We will extend the O by using an asterisk in lieu of the phrase "up to polynomial factors in the input size," much like the original symbols might replace an "up to constant factors." For example, $O^*(1)$ is the set of all polynomially bounded functions taking naturals to naturals.

Note the importance of the effect of the asterisk being in reference to the input size, and *not* the whole argument of the O. To indicate the latter, we use the notation $\text{poly}(\cdot)$, which denotes the class of functions that are bounded by a polynomial in the entire argument, entailing $\text{poly}(1)$ being the same as $O(1)$, but $\text{poly}(n)$ being the same as $O^*(1)$, where n is the size of the input.

Let k be some computable function that assigns to every instance I of some computational problem a natural number $k = k(I)$. We call k the *parameter* of the instance I. An algorithm for the problem is called *fixed-parameter tractable (fpt)* if it solves the problem on instances of size n with parameter k in time $f(k) \cdot \text{poly}(n)$, where f is some computable function. This will made be more formal when needed.

Problems and Complexity Classes. Complexity classes will be typeset in *sans serif* fonts ($\mathsf{P}, \mathsf{NP}, \mathsf{EXP}$ etc.) Problems will, most of the time, be referred to in prose ("the longest path problem"). When there are so many different problems that are relevant in the current context as to make this practice cumbersome, we will give a proper name to them, and typeset this name in small capitals (Longest Path).

1.2. Combinatorics and Probability

Sets. Whenever X is a finite set and k is a natural number, we write $\binom{X}{k}$ for the set of all subsets of X of size k, and we denote with 2^X the *power set* of X, *i.e.*, , the set of all subsets of X. Note that these sets conveniently contain $\binom{|X|}{k}$ and $2^{|X|}$ elements, respectively. For any logical predicate P, we use the *Iverson bracket* $[P]$, which evaluates to one if P is true, and zero otherwise. For instance, $\sum_{i=0}^{2n} i \cdot [i \text{ is odd}] = n^2$.

For a random variable X, we write $\mathbb{E}(X)$ for its expectation. For another random variable Y, $\text{Cov}(X, Y) = \mathbb{E}(XY) - \mathbb{E}(X)\,\mathbb{E}(Y)$ is the *covariance* of X and Y, and the *variance* of X is $\text{Var}(X) = \text{Cov}(X, X)$. Observe that Cov is a bilinear form.

Graphs. A *graph* is a pair (V, E), consisting of a set V of *vertices* or *nodes*, and a set E of pairs of nodes, called *edges*. If the pairs in E are unordered pairs of the form $\{u, v\}$ with $u, v \in V$ (*i.e.*, sets of size at most two), we call G *undirected*. Hence, if E contains ordered pairs of the form (u, v) with $u, v \in V$, we say that G is *directed* (or that G is a *digraph*), in which case we may refer to the elements of E not as edges, but as *arcs*. For any two vertices u and v of a graph, we mean the edge $\{u, v\}$ or (u, v) between u and v whenever we juxtapose u and v as uv, regardless of whether G is directed or not, which will be clear from the context. In this case, we call the vertices u and v the *endpoints* of the edge uv, and for the case of directed graphs, we call u its *head* and v its *tail*. An edge uu is called a *loop*, and a graph that does not contain edges of this form is called *loopless*. Given a graph G, we may write $V(G)$ and $E(G)$ for its set of vertices and edges, respectively. Graphs will generally have n vertices and m edges, and in all problems about graphs, n and m will consistently refer to the number of vertices and edges in the input graph, respectively.

A *subgraph* of a graph (V, E) is a graph (V', E') such that all the edges in E' have their endpoints in V'. A *walk* in a (V, E) is a sequence of t vertices $v_1, \ldots, v_t$ such that $v_i v_{i+1} \in E$ holds for all $1 \leq i < t$. If additionally, the sequence does not contain a vertex repeatedly, it is called a *path*. Here, the number t of vertices (not edges) in the sequence is the *length* of the walk or path. A walk or path of length t is also called a t-walk or t-path, respectively. The set of paths or walks of a graph G we denote as $\mathscr{P}(G)$ and $\mathscr{W}(G)$, respectively. If the first and the last vertex of the walk agree, we call the walk a *circuit*, and if a circuit without its last vertex is a path, we call it a *cycle*. Unless otherwise noted, all graphs are undirected, simple and loopless.

1.3. Algebra

Algebraic structures. A *semigroup* is a set S together with an associative binary operation $\circ$, and a semigroup is called a *monoid* if it contains an identity element. A *group* is a monoid in which every

element is invertible. Groups in which any two elements commute are called *abelian*. The group of permutations on n elements is written $\mathfrak{S}_n$.

A *ring* is a set R together with two associative binary operations, $+$ and $\circ$, such that R is a monoid with $\circ$, and an abelian group together with $+$. Additionally, $\circ$ and $+$ have to be compatible in the usual sense. We call a ring *commutative* if its multiplication is.

For a field F, an F-algebra A is a ring that is simultaneously a vector space over F, and the multiplication of the field is compatible with multiplication in the ring in the usual sense. In particular, $1 \cdot a = a$ should hold for all $a \in A$, as well as $(\lambda a)b = \lambda(ab) = a(\lambda b)$ for all $a, b \in A, \lambda \in F$.

An algebra is called *graded* if it admits a direct sum composition $A = \bigoplus_{i \in \mathbb{N}} A_i$ as a vector space, and in such a way that $A_i A_j \subseteq A_{i+j}$. An element is said to be of *degree* i if it is contained in A_i. Consequently, 0 is of degree i for all i.

Computation. We will perform calculations in various algebraic structures. Unless the context requires further clarifications, additions and multiplications will be written using the ordinary symbols "$+$" and "$\cdot$", and we may even drop the "$\cdot$" and just indicate a multiplication through a juxtaposition of elements.

In various places of the thesis, we will employ operations over finite fields of characteristic two. We record here once some important facts about the algorithms used in this respect, and shall later on use them rather implicitly, referring to them, correctly, as standard results. Let F be a finite field of characteristic 2 and of size $q = 2^\ell$, and let $p \in \mathbb{Z}_2[x]$ be an irreducible polynomial of degree ℓ. Since p is irreducible, the ideal $\mathfrak{p}$ generated by p is maximal and $\mathbb{Z}_2[x]/\mathfrak{p}$ is a field containing 2^ℓ elements. Arithmetic in F can then be implemented in time $\mathrm{poly}(\log(q)) = \mathrm{poly}(\ell)$ by using naive algorithms for polynomial multiplication and modular reduction, *e.g.* Euclid's algorithm for the latter, modulo $\mathfrak{p}$. Constructing an irreducible polynomial of degree ℓ (which is necessary in order to reduce modulo this polynomial later) can as well be performed in time $\mathrm{poly}(\ell)$: It suffices to choose a random polynomial of degree ℓ, and

check it for irreducibility using one of the many algorithms for this task (cf. the discussion in [Sho94]). To get rid of randomization in fields of small characteristic (such as two), we can use Shoup's deterministic construction [Sho88].

The constant $\omega \geq 2$ is the exponent of matrix multiplication, *i.e.*,

$$\omega = \inf\{\alpha \mid \text{two } n \times n \text{ matrices can be multiplied in } n^\alpha \text{ time}\}.$$

The best known upper bound currently is $\omega \leq 2.374$ [Sto10; Wil12; Gal14].

Important Sets. The naturals, integers, rationals, reals and the complex numbers are written $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$ and $\mathbb{C}$, respectively. Given a commutative ring R and some set $X = \{x_1, \ldots, x_n\}$ of indeterminates, the polynomial ring over the indeterminates X is written $R[x_1, \ldots, x_n]$, or $R[X]$.

Part I.

Faster Algorithms for Hard Graph Problems

2. Introduction

The realistic nature of many graph problems makes it tempting to motivate their study through humorous or historical examples. The famous *Seven Bridges of Königsberg* and their analysis by Leonhard Euler are often called upon as a specimen of the latter category, and indeed, this serves as a good justification for the classic concepts of an[1] *Euler cycle*, or *Eulerian graphs*. On a more entertaining, or at least silly, note, a *Hamiltonian path* can be exemplified in the realm of student life as a pub crawl, and generalizing this to finding paths of given length k might even be seen as a concession to the severe practical impediments entailed by the requirement of visiting *all* nodes during one pub crawl.

More seriously, it is true that graph algorithms are among the most classic of topics in algorithms research, and a vast body of work has been dedicated to their study. Indeed, any general textbook on algorithms will in large part be concerned with these algorithms (cf. the standard references [Cor+09; KT06; MS08; DPV08]). The textbook examples of graph algorithms typically solve the respective problem—Eulerian cycles, shortest paths, minimum spanning trees, and so forth—*fast*, in time polynomial, preferrably (quasi)linear, in the input size. By contrast, in this part, we will be concerned with *hard* graph problems, which means that the considered problems are all NP-hard, ruling out polynomial-time solutions unless P equals NP. The fact that these problems are formally NP-hard doesn't mean that we cannot solve them quickly on *some* instances, and it is the case that many hard problems are routinely solved in adequate time on real input instances. It is therefore of vital interest to give better algorithms even for those problems where this will never culminate in polynomial-time solvability.

[1] We ask the non-German reader to resist the urge to pronounce Euler's name with anlaut as in *you*, and instead pretend the name were spelled *Oyler*.

In Chapt. 3, we give a comprehensive overview of some of the most prevalent techniques in this area, which will become relevant later, when we compare the new method we develop in this part to existing ones.

A prototypical example for such hard problems is the aforementioned one of finding Hamiltonian paths. Indeed, we will be concerned with the generalization that was alluded to, namely finding paths that need not traverse all vertices of a graph, but only at least some number k of vertices, without hitting any one vertex twice. This is the content of Chapt. 4, and we refer the reader to this chapter for detailed background on the problem and a statement of the new results. In short, we give a faster method to estimate the number of such paths in a given graph quicker than was previously known. Furthermore, we give new and fast deterministic algorithms for the case where the input graph is promised to contain only few paths of a given length. To do so, we introduce a new method called *Extensor-Coding*, based on algebraic objects called *extensors,* which live in the *exterior algebra*. We also show how this technique can be seen as a strict generalization of the approaches from Chapt. 3.

We furthermore apply this new method to the problem of finding spanning trees that contain many internal nodes (and hence, few leaves), which is NP-hard as well, and some other hard graph problems. We produce new, faster, and elegant deterministic algorithms for the problems. This is done in Chapt. 5.

3. Review: Algorithmic Techniques

The past casts its shadows into the future.

Hallodrian proverb

In this chapter, we will give a gentle overview over various algorithmic techniques that are employable to solve a kind of subgraph problem. A *subgraph problem* is, informally speaking, the task of detecting the presence or absence of a certain subgraph, which is typically fixed beforehand, or even computing the number of occurrences of such a subgraph, in a given input graph.

We shall mainly be concerned with applying these techniques to a flagship subgraph problem, for the simplicity with which it can be described and the research efforts that have gone into its algorithms. This problem is the *longest path problem*, and it asks: Given a *directed* graph, what is the longest path it contains? Or, in its guise as a decision problem: Given a directed graph and some number k, is there a path in the graph of length at least k? This problem is obviously NP-hard, since it generalizes the problem of detecting a Hamiltonian path in a graph, *i.e.*, $k = n$. Since any subpath of a path is again a path, it does not matter whether we ask for a path of length at least k, or one of length exactly k, and we will prefer the latter in what follows. When formulated in this way, this problem is also referred to as the *k-path problem*. It is, however, important to note that the techniques laid out here are by no means limited to this problem, and later on, we will encounter other applications, such as the problem of finding directed spanning trees with few inner nodes. All these problems will have in

common that they are NP-hard, so we will search for solutions in the realm of fixed-parameter tractable algorithms.

Of course, other relevant techniques have been proposed in the past to solve this kind of problems. Our selection is motivated by the fact that these are, in our opinion, the most well-understood and well-studied ones. The reason why we devote ample space to present well-known techniques at all is that, later in the thesis, we will generalize and unify them using our new method of Extensor-Coding. This should allow readers without close acquaintance with the original works, where the techniques were introduced, to appreciate the results, and makes this work more self-contained.

3.1. Color-Coding

Color-Coding was introduced by Alon, Yuster and Zwick in 1994 [AYZ94; AYZ95]. It is generally considered a breakthrough result, resolving to the affirmative a conjecture of Papadimitriou and Yannakakis [PY93; PY96]. Namely, Color-Coding makes it possible to find paths of length logarithmic in the number of vertices in a given input graph in time polynomial in the number of vertices. More precisely, it made it possible to decide the presence of a path of length k in time $\exp(O(k)) \cdot \mathrm{poly}(n)$. We proceed now with a description of the technique, which turns out to be delightfully simple. Speaking somewhat hyperbolically, the essential idea is *not* to solve the problem at all, but instead, solve a much simpler problem *a lot of times*, and then extract from the answers to this simpler problem an answer to the original question, at least with high probability. The much simpler problem mentioned above is a *colored* variant of the k-path problem, which explains the origin of the term *Color-Coding*.

3.1.1. The Colorful k-Path Problem

The *colorful k-path problem* asks: Given an integer k and a graph G with its vertices colored arbitrarily using t colors, is there a path of length k in the graph that does not contain any of the colors twice?

We call such a path *colorful*. A coloring of a graph is here an arbitrary assignment of colors[1] to vertices. In particular, it does not have to suffice any additional conditions of being a proper coloring or the like.

As for solving this problem algorithmically, this is done in a straightforward way using dynamic programming.

Lemma 3.1.1. *There is an algorithm that decides the existence of a colorful k-path in a given directed graph with n nodes and m edges that is colored using t colors, in time $O(2^t \cdot (n+m))$.*

Proof. We record for every subset C of the t colors and every vertex $v \in V(G)$ whether there is a colorful path of length $|C|$ in G that ends in v and uses *only* the colors from the subset C. Let us write the truth value of this condition down in a table $P[C, v]$, for each subset C of colors, and $v \in V$. Note that the number of colors in a set is equal to the length of the path under consideration, since any color may appear at most once. The base cases are the singleton sets of colors, and for each node v, there is a colorful path ending in v with one vertex if and only if v has the color that the singleton is comprised of. Formally, this is

$$P[\{c\}, v] = [v \text{ has color } c], \text{ for all } v \in V(G)\,.$$

Now, if we want to decide for a node v and some set C of colors with $|C| > 1$ whether there is a colorful path ending in v that uses only the colors from C, we proceed as follows: Let c_v be the color of a vertex v. Consider the set $N^-(v)$ of in-neighbors of v, *i.e.*, the set of vertices u in $V(G)$ such that uv is an edge in G. There is now a colorful path ending in v using colors only from C if and only if there is some colorful path ending in one of the in-neighbors of v, which, in addition to using colors only from C, does not use the color c_v. We formally have therefore

$$P[C, v] = \bigvee_{u \in N^-(v)} P[C - \{c_v\}, u]\,. \tag{3.1}$$

The answer to the question of whether there is a colorful path of length

[1] Despite the colorful language used in this respect, vertex colors are, formally, just numbers between 1 and t.

k in G is now the logical or over all sets of colors of size k and all possible end vertices.

Since every edge is considered exactly once per color subset, and the same for every vertex, and there are less than 2^t subsets of C of size k, this procedure can be implemented in time $O(2^t \cdot (n+m))$. □

It is not immediately clear what we have gained by solving this superficially similar problem. Of course, if there is a colorful path of length k in G, then certainly, there is also a simple path of length k in G. However, the absence of a colorful k-path does not allow us to conclude that there does not exist a simple k-path: The graph might just be colored in such a way that all the simple paths received color assignments that make them non-colorful. As hinted at before, we will get rid of this by performing a large, but not too large, number of randomized trials.

3.1.2. Random Colorings

We now take care of the problem that all simple k-paths might become non-colorful in our chosen coloring of the graph. We can ask ourselves how probable this unfortunate situation is under a *random* coloring. As it turns out, the probability of this *not* happening, that is, us being lucky, is only exponentially small *in* k. Of course, if G does not have a k-path in the first place, there is nothing we could accidentally make disappear, so we don't have to worry about this case.

Lemma 3.1.2. *Consider a path of length k, and a random coloring of its vertices with $t \geq k$ colors. Then, the probability that the resulting colored path is colorful is at least $e^{-k+o(1)} \cdot \left(\frac{t}{t-k}\right)^{t-k+1/2}$.*

Proof. The probability of the path becoming colorful under a random coloring we can now compute as follows: There are t^k colorings of the k vertices with t colors in total. The colorings that make the k-path colorful correspond to the injections $\{1,\dots,k\} \to \{1,\dots,t\}$, of which there are $\frac{t!}{(t-k)!}$, provided $t \geq k$. If $t > k$, the quotient $\frac{t!}{(t-k)!t^k}$ can now be estimated using Stirling's formula. We do this explicitly using a

variant due to Robbins [Rob55], who showed that for all naturals $s > 0$,

$$s! = \sqrt{2\pi} s^{s+1/2} e^{-s} \cdot e^{r_s} ,$$

where r_s satisfies the inequalities

$$\frac{1}{12s+1} < r_s < \frac{1}{12s} .$$

Therefore,

$$\frac{t!}{(t-k)!} = \frac{t^{t+1/2} e^{-t} \cdot e^{r_t}}{(t-k)^{t-k+1/2} e^{k-t} \cdot e^{r_{t-k}}} = e^{r_t - r_{t-k} - k} \cdot \left(\frac{t}{t-k}\right)^{t+1/2} \cdot (t-k)^k$$

and

$$\frac{t!}{(t-k)! t^k} = e^{r_t - r_{t-k} - k} \cdot \left(\frac{t}{t-k}\right)^{t-k+1/2} .$$

Using the bounds on r_t and r_{t-k}, this is bounded from below by

$$e^{-k+1/(12t+1)-1/(12t-12k)} \cdot \left(\frac{t}{t-k}\right)^{t-k+1/2} ,$$

proving the case of $t > k$.

If $t = k$, plugging in the estimates for r_k gives again the inequality from the statement. □

Therefore, coloring a path of length k at random using k colors yields a colorful path with probability at least e^{-k}.

3.1.3. The Algorithm

The algorithm is now very simple: We pick a random coloring of the graph using $t = k$ colors, and decide using dynamic programming in time $O(2^k(n+m))$ if there is a colorful k-path in the resulting colored graph. Repeating this procedure e^k times yields constant one-sided error probability. The running time of this algorithm in total is therefore $O((2e)^k \cdot (n+m))$, and the procedure is captured in Algorithm CC.

Algorithm CC (*Detect k-path using Color-Coding.*) *Given directed*

graph G and integer k, this algorithm determines if G contains a k-path.

CC1 (Initialize.) Set $j = 1$.

CC2 (Set up jth trial.) Color each vertex uniformly at random with one of the t colors, and obtain a vertex-colored graph G'.

CC3 (Solve colored instance.) Use the algorithm from Lemma 3.1.1 to solve the resulting instance of the colorful k-path problem on G'. If the algorithm detects such a path, output 'yes.'

CC4 (Repeat e^k times.) If $j < e^k$ then increment j and go to CC2.

CC5 (All trials failed.) Output 'no.' □

Theorem 3.1.3 ([AYZ95])**.** *Algorithm CC decides the existence of a k-path in a given directed graph with n nodes and m edges in time $O((2e)^k \cdot (n+m))$. The algorithm has a constant one-sided error probability of claiming the absence of such a path in a graph that actually does have a k-path.*

Proof. Correctness and the claim about the error probability follows from Lemmas 3.1.1 and 3.1.2, and we already discussed the algorithm's running time. □

Hüffner's t

While it is already quite satisfying that we can now detect logarithmically long paths in polynomial time, the degree of this polynomial is of course dependent on the basis of the exponential appearing in the running time. Lowering this basis is therefore of significant interest. In particular, choosing the number of colors t equal to the length of the sought path k is rather uninspired. Of course, using fewer than k colors is nonsensical, but using more than k, say $t = \alpha \cdot k$ for some constant factor $\alpha > 1$, might enable a tradeoff between the running time of the dynamic program, which becomes slower as t grows, and the probability of obtaining a colorful path, which increases as t grows. Indeed, as observed by Hüffner *et al.* [HWZ08], this is the case.

Theorem 3.1.4 ([HWZ08])**.** *There is an algorithm that decides the existence of a k-path in a given directed graph with n nodes and m edges in time $O(4.312^k \cdot (n+m))$. The algorithm has a constant one-sided error probability of claiming the absence of such a path in a graph that actually does have a k-path.*

Proof. Let us consider the term $\left(\frac{t}{t-k}\right)^{t-k}$, which for $t = \alpha \cdot k$ becomes $\left(\frac{\alpha}{\alpha-1}\right)^{(\alpha-1)\cdot k}$, which is clearly monotone in k for $\alpha > 1$. The function $2^\alpha \cdot \frac{\alpha}{\alpha-1}^{1-\alpha}$, which is the exponential part of the running time of the algorithm, has a unique minimizer α^* on $(1, \infty)$, which satisfies $1.30201 \leq \alpha^* \leq 1.30202$, leading to a running time bounded by $O(4.312^k \cdot (n+m))$. We skip the analytical details of this argument, which can be found in [Gut+18]. Using αk colors therefore yields an algorithm of the desired running time, and correctness carries over directly from the case of $t = k$. □

Approximate Counting

Let us quickly sketch how to employ the method of this section to approximately count the paths of length k. A slower variant of this result, using $t = k$, is presented in [Alo+08], and our presentation of the result is informed by theirs. Throughout this subsection, let N denote the quantity we seek to estimate, *i.e.*, the number of k-paths in the input graph.

First, observe that Eq. (3.1) can be used to count solutions of the colored problem, just by replacing the logical or with a proper sum. All that remains is then to estimate the number of k-paths in the original, uncolored instance, from the number of colorful k-paths, averaged over all trials. This is accomplished by the following analysis: Let c be a random coloring of the input graph as produced in step CC2 in Algorithm CC. Let P be any k-path in the input graph. Denote with $\chi_P(c)$ the random variable, depending on the coloring c, that is defined as

$$\chi_P(c) = \begin{cases} 1, & \text{if } c \text{ makes } P \text{ into a colorful path} \\ 0, & \text{otherwise.} \end{cases}$$

Of course, over the various choices of P, the χ_P are identically (but certainly not independently) distributed Bernoulli variables with parameter

$$p = \frac{t!}{(t-k)!}\,.$$

which is therefore the expected value of χ_P, as argued at some length before.

Let furthermore

$$\chi = \sum_P \chi_P$$

be the sum of the χ_P, where P varies over all distinct k-paths in the input graph. By linearity of expectation, the expected value of χ equals

$$\mathbb{E}\,(\chi) = N \cdot p\,,$$

and thus χ/p is an unbiased estimator of the quantity N.

We can sample values of $\chi(c)/p$ by choosing colorings c uniformly at random, computing the number of colorful k-paths in the graph colored using c and the sum-version of Eq. (3.1), and averaging over the number of trials.

We are left to prove concentration of χ/p around N, *i.e.*, bound $\mathrm{Var}\,(\chi/p)$. But this is standard: One readily observes that the expectation of $\chi_P\chi_{P'}$ (*i.e.*, both P and P' are colorful under the same coloring) for two paths P, P' is at most p. Therefore, the covariance

$$\mathrm{Cov}\,(\chi_P, \chi_{P'}) = \mathbb{E}\,(\chi_P \cdot \chi_{P'}) - \mathbb{E}\,(\chi_P) \cdot \mathbb{E}\,(\chi_{P'}) \leq \mathbb{E}\,(\chi_P \cdot \chi_{P'})$$

is bounded by p as well.

Using the bilinearity of the covariance (and the fact that variance is the covariance of a variable with itself), we can thus bound the variance of χ with $N^2 \cdot p$. Chebychev's inequality then allows to bound the probability that χ exceeds or falls below its expectation by more than (an additive term of) $\varepsilon \cdot N \cdot p$ by a constant, and using repeated trials and a median argument then allows to decrease this probability to an arbitrarily small constant $\delta > 0$ using $O(\log(1/\delta))$ repititions.

We have the following theorem.

Theorem 3.1.5 ([Alo+08]). *There is a randomized algorithm that, upon input a directed graph G, a natural number k as well as an error bound*[2] *$\varepsilon > 0$ produces an estimate $\tilde{N}$ of the number N of k-paths in G, such that with 99% probability,*

$$(1-\varepsilon)N \leq \tilde{N} \leq (1+\varepsilon)N\,,$$

and this algorithm runs in time $O((2e)^k \cdot (n+m)\varepsilon^{-2})$.

3.2. Representative Sets

The idea of *representative sets* (or *families*) can be traced back at least to work of Lovász in 1977 [Lov77], and was algorithmically applied first by Monien in 1985 [Mon85]. We will lay out the concept of these families and prove some of their basic properties, and also sketch an algorithm of how to compute them relatively fast. However, we will not cover the extremely intricate state-of-the-art as established by Fomin *et al.* [Fom+16], and the account given here should provide a good intuition for the approach. Additionally, to keep our presentation crisp, we will restrict our attention to uniform matroids.

As in Color-Coding, we will iteratively build up longer and longer paths, starting with a single vertex. Now, instead of remembering each and every partial solution, *i.e.*, path of length less than k, that we already constructed (of which there might be $n^{\Omega(k)}$ many), we will retain only a much smaller subset of them. Namely, our aim is to keep a subset of partial solutions that is equivalent to the full set with respect to being augmentable in such a way as to produce a full solution, *i.e.*, a path of length k. This intuition guides the following definition.

Definition 3.2.1. Let $\mathcal{F} \subseteq 2^U$ be a family of subsets of a universe U with n elements, and let $q \leq n$ be a natural. Then, a subfamily $\mathcal{F}' \subseteq \mathcal{F}$

[2] We avoid needless pedantry in assuming bounds of this kind real. Of course, in an actual implementation, one might choose to represent them as the reciprocal of some natural.

is said to *q-represent* (or is a *q-representant of*) $\mathcal{F}$ if the following condition holds for every subset A of U of size at most q: If there is some set $B \in \mathcal{F}$ disjoint from A, then there is a set $B' \in \mathcal{F}'$ disjoint from A.

As common sense suggests, we say $\mathcal{F}'$ *represents* (or is a *representant of*) $\mathcal{F}$ if $\mathcal{F}'$ q-represents $\mathcal{F}$ for some q.

3.2.1. First Properties

Let us note some simple and important consequences of the foregoing definition.

Proposition 3.2.2. *1. q-representation is reflexive: Every family q-represents itself, for all q.*

2. q-representation is transitive: If $\mathcal{F}''$ q-represents $\mathcal{F}'$, which in turn q-represents $\mathcal{F}$, then $\mathcal{F}''$ q-represents $\mathcal{F}$.

3. q-representation is definite: The empty family 0-represents only the empty family, and the empty family is represented only by the empty family.

4. Representation and union commute: If $\mathcal{E}', \mathcal{F}'$ q-represent $\mathcal{E}, \mathcal{F}$, respectively, then $\mathcal{E}' \cup \mathcal{F}'$ q-represents $\mathcal{E} \cup \mathcal{F}$, and this extends to arbitrary unions.

Proof. We only demonstrate the proof of the third item: The second claim is trivial by the condition that $\mathcal{F}' \subseteq \mathcal{F}$ whenever $\mathcal{F}'$ q-represents $\mathcal{F}$ for any q.

For the other claim, note:[3] Let $\mathcal{F} \subseteq 2^U$ be some family of subsets of U, and let $\mathcal{F}'$ 0-represent $\mathcal{F}$. Let $A = \emptyset$ be the only subset of U of size zero. If $\mathcal{F} \neq \emptyset$, pick $B \in \mathcal{F}$. Then A is in particular disjoint from B. In order to 0-represent $\mathcal{F}$, $\mathcal{F}'$ has to contain some B' (and being disjoint from $A = \emptyset$ is automatic for B'). Therefore, $\mathcal{F}'$ is not empty. □

[3] We spell this out in painstaking detail. The astute reader, to which the statement of the proposition appears as a blatant triviality, is spot-on.

For our application to finding long paths in a graph G, unsurprisingly, the universe U is the just the vertex set of the graph. Now, for $i \leq k$ a natural number, and v some vertex of G, let $\mathcal{P}_i^v$ be the following family:

$$\mathcal{P}_i^v = \{V(P) \mid P \text{ is a path of length } i \text{ in } G \text{ ending in } v\}.$$

Keep in mind that we mean with the length of a path not the number of its edges, but of its vertices. Put this way, G has a k-path if and only if $\mathcal{P}_k^v \neq \emptyset$ for some vertex v of G. From Proposition 3.2.2, we might as well take any family $\mathcal{P}'$ that 0-represents $\mathcal{P}_k^v$, and $\mathcal{P}' \neq \emptyset$ is equivalent to $\mathcal{P}_k^v \neq \emptyset$. We will now describe how to successively compute such a 0-representant of $\mathcal{P}_k^v$ for some fixed v, and then let v vary over all vertices.

First for some notation: For a set family $\mathcal{F}$ and a set S, $\mathcal{F} \dot\cup S$ is the set of all (non-empty) *disjoint* unions of the form $F \cup S$, *i.e.*, such that $F \in \mathcal{F}$ and $F \cap S = \emptyset$.

We first observe the following:

Proposition 3.2.3. *Let $\mathcal{F} \subseteq 2^U$, and let $S \subseteq U$ be of size $s \leq q$. If $\mathcal{F}'$ q-represents $\mathcal{F}$, then $\mathcal{F}' \dot\cup S$ $(q-s)$-represents $\mathcal{F} \dot\cup S$.*

Proof. First, note that $\mathcal{F}' \dot\cup S \subseteq \mathcal{F} \dot\cup S$ clearly holds. Let then A be of size at most $q - s$. Let $B \in \mathcal{F} \dot\cup S$ be disjoint from A (if there is no such B, there is nothing to show). We want to prove existence of $B' \in \mathcal{F}' \dot\cup S$ disjoint from A. By definition of $\dot\cup$, B is a union of the form $F \cup S$ for some $F \in \mathcal{F}$ disjoint from S. The existence of such B in particular implies that both F and S are disjoint from A, and moreover that F is disjoint from $A \cup S$, a set of size q. Because $\mathcal{F}'$ q-represents $\mathcal{F}$ by assumption, there is some $B' \in \mathcal{F}'$ that is disjoint from $A \cup S$. In particular, since B' is disjoint from S, $B' \cup S$ is an element of $\mathcal{F}' \dot\cup S$. And since B' is also disjoint from A, and (as argued before) A is disjoint from S, $B' \cup S \in \mathcal{F}' \cup S$ is disjoint from A. □

3.2.2. Representing Paths

The following insight is akin to Eq. (3.1), for $i > 1$.

$$\mathcal{P}_i^v = \bigcup_{u \in N^-(v)} \left(\mathcal{P}_{i-1}^u \dot\cup \{v\}\right) \tag{3.2}$$

The truth of this equation is evident from the very definition of $\dot\cup$. With the aid of Proposition 3.2.3 and the other foregoing observations, we can lift this equation to representative sets: Assume that $\mathcal{P}'^u_{i-1}$ q-represents $\mathcal{P}^u_{i-1}$ for some naturals i, q and vertex u. Then, $\mathcal{P}'^u_{i-1} \dot\cup \{v\}$ $(q-1)$-represents $\mathcal{P}^u_{i-1} \dot\cup \{v\}$. Let

$$\mathcal{P}'^v_i = \bigcup_{u \in N^-(v)} \left(\mathcal{P}'^u_{i-1} \dot\cup \{v\}\right) . \tag{3.3}$$

Since unions and representation commute (4 in Proposition 3.2.2), $\mathcal{P}'^v_i$ actually $(q-1)$-represents $\mathcal{P}^v_i$. Unfortunately, computing $\mathcal{P}'^v_i$ directly with increasing i, as suggested by its definition, will break down after a few iterations, as the $\mathcal{P}'^v_i$ might grow roughly with the maximum degree of G in each iteration. Instead, we seek to compute in every step a smaller subset $\mathcal{P}^{*v}_i \subseteq \mathcal{P}'^v_i$ which inhibits this blowup. This raises the question of how big representative families can actually become. There is a surprisingly clear-cut answer to this, whose very elegant proof we defer to much later in Sect. 4.4.5, when we have introduced the exterior algebra.

Proposition 3.2.4 ([Lov77])**.** *Let $\mathcal{F} \subseteq \binom{U}{p}$,* i.e.*, a set family containing only sets of size p. Then $\mathcal{F}$ has a q-representant of size $\binom{p+q}{p}$.*

For now, suffice it to say that there is a natural way to interpret sets of size p as elements of a vector space of dimension $d = \binom{p+q}{p}$ for any choice of q. We can do this in such a way that the notion of $\mathcal{F}'$ being a q-representant of some set family $\mathcal{F}$ translates to the vectors corresponding to $\mathcal{F}'$ spanning the same space as the vectors corresponding to $\mathcal{F}$. Then it is elementary linear algebra that every set family containing more than d sets, *i.e.*, vectors, has a basis of size at most d as a subset. What remains to do is to exhibit an algorithm that

computes small representative families.

3.2.3. The Algorithm

The following is an easy (and of course inefficient) algorithmic variant of the proof of the preceding proposition, and we will encounter it again, later on.

Proposition 3.2.5. *Let p, q be naturals, and let $k = p + q$. There is a deterministic algorithm that, given a set family $\mathcal{F}$ containing sets of size p over a universe of size n, computes a q-representant $\mathcal{F}'$ of size $\binom{k}{p}$ of $\mathcal{F}$, and takes time $\exp(O(k)) \cdot \mathrm{poly}(|\mathcal{F}|, n))$.*

Proof sketch. As insinuated above, we can interpret sets of size p as vectors of dimension $\binom{p+q}{p} = \binom{k}{p} \leq 2^k$. Naturally, the set family $\mathcal{F}$ then becomes a set of vectors, say column vectors, which are most conveniently arranged as a matrix of size $|\mathcal{F}| \times \binom{k}{p}$. The entries of these vectors will have bit lengths bounded polynomially in n, and we can construct the matrix from the input family in the required time bound. It is then a standard matter to compute a column basis of such a matrix in time polynomial in its dimensions, which implies the statement of the proposition. □

The state-of-the-art for accomplishing this task is due to Fomin *et al.* [Fom+16]. Note, however, that there is an additional parameter that controls the running time as well as the size of the solution, and optimizing this parameter is typically an important step in applying their result.

Proposition 3.2.6 ([Fom+16, Theorem 4.15])**.** *Let $0 < x < 1$, and p, q be naturals with $k = p + q$. There is a deterministic algorithm that, given a set family $\mathcal{F}$ containing sets of size p, computes a q-representant $\mathcal{F}'$ of size $x^{-p}(1-x)^{-q} \cdot 2^{o(k)}$, and takes time $O(|\mathcal{F}| \cdot \log|\mathcal{F} + (1-x)^{-q} \cdot 2^{o(k)} \cdot \log n)$.*

Let us now spell out how to use representative sets to actually solve the longest path problem. To this end, consider Algorithm RS.

Algorithm RS (*Detect k-path using representative sets.*) *Given directed graph G and integer k, this algorithm determines if G contains a k-path.*

RS1 (Set up $\mathcal{P}^*$.) Let ${\mathcal{P}^*}_1^v = \{\{v\}\}$ for all $v \in V(G)$. Let $i = 2$.

RS2 (Compute unions.) Compute ${\mathcal{P}'}_i^v$ for each $v \in V(G)$ as in Eq. (3.3).

RS3 (Reduce representants.) Use the algorithm asserted by either of Props. 3.2.5 or 3.2.6 to compute a $(k-i)$-representant ${\mathcal{P}^*}_i^v$ of ${\mathcal{P}'}_i^v$, for all $v \in V(G)$.

RS4 (Repeat k times.) If $i \leq k$, then increment i and go to RS2.

RS5 (Decide.) If ${\mathcal{P}^*}_0^v$ is non-empty for some v, then return 'yes.' Otherwise, return 'no.' □

Using Proposition 3.2.5 in Algorithm RS allows to immediately deduce the following:

Theorem 3.2.7. *There is a deterministic algorithm that, given a directed graph G and an integer k, decides whether k has a k-path in time* $\exp(O(k)) \cdot \mathrm{poly}(n)$.

It requires a lot more care to put Proposition 3.2.6 to efficient use. A somewhat intricate analysis and a clever choice of the values of x across the different iterations then allows to prove:

Theorem 3.2.8 ([Fom+16, Sect. 5.5.1])**.** *There is a deterministic algorithm that, given a directed graph G and an integer k, decides whether G has a k-path in time $O(2.619^k \cdot m \log n)$.*

Minimum Weight Paths

The above approach can relatively easily be modified to yield not only paths of a certain length, but—if the input graph is weighted—to produce a path of a certain length of minimum (or maximum) weight. This necessitates extending the notion of a representant to so-called *min-representants*, and an algorithm for computing such min-representant. Otherwise, everything can remain the same; in particular, there are no malusses in running time.

Of course, the approach of representative sets was not designed for handling approximate counting of solutions.

3.3. Labeled Walks

While the previous two approaches, Color-Coding and representative sets, are of purely combinatorial flavour, we now turn to the first technique that might be described, at least partially, as algebraic, in that it encodes combinatorial objects as polynomials, *i.e.*, the epitomes of algebraic objects. Whenever we speak of a polynomial in this subsection, we mean a polynomial with coefficients from a field F of characteristic two and of size to be determined.

The main goal of the *labeled walk* approach of Björklund *et al.* [Bjö+17a] was to give an algorithm for the *undirected* case running in time $1.66^k \cdot \text{poly}(n)$. This is achieved by a method called *narrow sieves*, which involves reducing the number of so-called labels used on the graph. The underlying walk labeling idea itself, however, remains valid also on directed graphs and when keeping *all* labels, and then reproduces the randomized $2^k \cdot \text{poly}(n)$ running time bound of Williams [Wil09]. This is nicely laid out in the textbook by Cygan *et al.* [Cyg+15b, Section 10.4], and the following presentation is guided by theirs.

3.3.1. Algebraic Encoding of Paths

First, with each directed edge $e \in E(G)$, we associate a symbolic variable y_e, and let a vector of variables $(x_i^{(1)}, \ldots, x_i^{(k)})^T$ be associated with each vertex $v_i \in V(G)$. The superscript index is referred to as the *label* of a vertex in a walk. Consider the following polynomial in the y_e and $x_i^{(j)}$:

$$P(x,y) = \sum_{w_1 \cdots w_k \in \mathscr{W}(G)} \sum_{\ell \in \mathfrak{S}_k} \prod_{i=1}^{k} y_{w_i w_{i+1}} \prod_{i=1}^{k} x_i^{\ell(i)}. \tag{3.4}$$

The crucial insight is that over characteristic 2, the sum can be restricted to paths instead of walks (although the proof of this is again rather

combinatorial):

Lemma 3.3.1. *Let $P(x, y)$ be the polynomial defined above, with coefficients in characteristic two. Then,*

$$P(x,y) = \sum_{w_1 \cdots w_k \in \mathscr{P}(G)} \sum_{\ell \in \mathfrak{S}_k} \prod_{i=1}^{k-1} y_{w_i w_{i+1}} \prod_{i=1}^{k} x_i^{\ell(i)} \,. \tag{3.5}$$

Proof. We will show that for every walk, an even number of copies of the monomial corresponding to this walk appears in the sum over all permutations of its labels. To this end, consider a non-simple walk $w_1, \ldots, w_k \in \mathscr{W}(G)$, and say that $w_i = w_j$ for some $i \neq j$. Fix now some even permutation ℓ. Then, ℓ composed with the transposition (ij) is odd and produces the same monomial. It is also clear that this is a bijection between even and odd permutations, and it follows that all summands corresponding to non-simple walks vanish, which is precisely what we claimed. □

In this form, the statement is true only over characteristic two. However, even over characteristic 0, a similar statement can be made when taking into account the sign of the permutation ℓ.

With the foregoing Lemma, it is easy to see that the property of the sum indices being paths, and not walks, allows to reconstruct the permutation ℓ as well as the path $w_1, \ldots, w_k$ from a monomial of the form $\prod_{i=1}^{k-1} y_{w_i w_{i+1}} \prod_{i=1}^{k} x_i^{\ell(i)}$. Since these monomials are trivially linearly independent for different paths, we immediately obtain:

Proposition 3.3.2. *The polynomial P is non-zero (as a polynomial) if and only if G contains a k-path.*

3.3.2. Testing for Zero

Let us state, for completeness, the well-known DeMillo–Lipton–Schwartz–Zippel Lemma.

Lemma 3.3.3 ([DL78; Sch80; Zip79])**.** *Let f be a non-zero, n-variate polynomial of total degree at most d, and let $\xi_1, \ldots, \xi_n \in S$ be chosen*

uniformly at random. Then, the probability that $f(\xi_1, \ldots, \xi_n) = 0$ *is at most* $d/|S|$.

Therefore, if we choose our field F to have at least $4k$ elements, then the polynomial P of degree $2k-1$—provided it is non-zero—evaluates to a non-zero element of F when evaluated at random points from F with probability at least $1/2$.

3.3.3. Evaluating the Polynomial

The only thing left to do is now to describe this evaluation. We again employ a dynamic programming table, somewhat similar in spirit to Eqs. 3.1 and 3.2. To this end, let $v \in V(G)$ be a vertex and $X \subseteq \{1, \ldots, k\}$ the subset of allowed labels. Define

$$T[X, v] = \sum_{\substack{w_1, \ldots, w_{|X|} \in \mathscr{W}(G), \\ w_{|X|} = v}} \sum_{\substack{\ell : \{1, \ldots, |X|\} \to X \\ \text{bijective}}} \prod_{i=1}^{k-1} y_{w_i w_{i+1}} \prod_{j=1}^{|X|} x^{\ell}_{w_j}(j)\,. \quad (3.6)$$

Clearly,

$$P(x, y) = \sum_{v \in V(G)} T[\{1, \ldots, k\}, v]\,.$$

Let $\mathbf{a} = \{a_v^j\}_{j \in \{1,\ldots,k\}, v \in V(G)}, \mathbf{b} = \{b_e\}_{e \in E(G)} \in F$ be arbitrary evaluation points for $P(x, y)$. Since evaluation commutes with summation and addition of polynomials, $P(\mathbf{a}, \mathbf{b}) = \sum_{v \in V(G)} T[\{1, \ldots, k\}, v](\mathbf{a}, \mathbf{b})$. We will drop reference to the evaluation points in what follows. Per definition, $T[\emptyset, v] = 0$. Furthermore, for $\emptyset \neq X \subseteq \{1, \ldots, k\}$ we have

$$T[X, v] = \sum_{u \in N^-(v)} b_{uv} \cdot \left(\sum_{l \in X} T[X - \{l\}, u] \cdot a_v^l \right). \quad (3.7)$$

3.3.4. The Algorithm

Taking all this together culminates in the following algorithm.

Algorithm LW (*Detect k-path using labeled walks.*) *Given directed graph* G *and integer* k*, this algorithm determines if* G *contains a* k*-path.*

LW1 (Initialize.) Construct the finite field F having more than $4k$ elements. Let $j = 0$ and $T[\emptyset, v] = 0$ for all $v \in V(G)$.

LW2 (Set up trial.) Pick $\{a_v^l\}_{l\in\{1,\ldots,k\},v\in V(G)}$ and $\{b_e\}_{e\in E(G)}$ uniformly at random from F.

LW3 (Build table T.) Use Eq. (3.7) for each $v \in V(G)$ and $X \subseteq \{1,\ldots,k\}$ in ascending cardinality of X to compute all entries of $T[X, v]$.

LW4 (Decide.) If $\sum_{v\in V(G)} T[\{1,\ldots,k\}, v] = 0$, output 'no.' Otherwise, output 'yes.' □

Theorem 3.3.4 (Implicit in [Bjö+17a])**.** *Algorithm LW decides the existence of a k-path in a given directed graph with n nodes and m edges in time $2^k \cdot \mathrm{poly}(k) \cdot m$. The algorithm has a constant one-sided error probability of claiming the absence of such a path in a graph that actually does have a k-path.*

Proof. Correctness follows from Proposition 3.3.2, the recursive equation (3.7) and Lemma 3.3.3.

As for the running time, step LW3 can be implemented as follows: Given the values of T at sets of size $|X| - 1$, filling $T[X, \cdot]$ for fixed X and all vertices can be done in time $O(km)$. In total, LW3 can be performed in time $O(2^k km)$. The final summation in LW4 takes time $O(n)$. Finally, construction of (in step LW1) and arithmetic in F (implicit in the algorithm) is possible by standard methods in time $O(\mathrm{poly}(k))$. □

Using Fewer Labels. The running time of Algorithm LW is obviously dominated by filling up the table T, which in turn contains $n \cdot 2^t$ entries, where t is the number of labels. Therefore, if there were some way to use fewer than k labels, the running time would improve by an exponential factor. In the general directed case, it is not clear how to do this. However, on *bipartite undirected graphs*, it is sufficient to use $k/2$ labels, in combination with an intricate exchange argument that allows to pair up non-simple walks, having them cancel out over

characteristic two. This reduces the running time for the bipartite case to $O^*(2^{k/2})$. Also note that while Eq. (3.4) can be extended to characteristic zero by introducing an appropriate sign factor to the sum over the permutations, it seems much harder to do so for the bipartite case.

In their ingenious work, Björklund *et al.* [Bjö+17a] managed to set up a similar argument for the non-bipartite undirected case, using a number of labels that leaves us with a running time of $O^*(1.66^k)$.

3.4. Algebraic Fingerprinting

The method of *algebraic fingerprinting*, studied in the works of Koutis and Williams [Wil09; Kou08; KW16; KW15] is the first purely algebraic technique when it comes to solving subgraph problems. We will introduce *ad hoc* several objects that we will encounter again later, and with more rigor. This was the first method proposed to solve the problem in randomized time $O^*(2^k)$.

3.4.1. Another Algebraic Encoding of Walks

The first of these objects is the multivariate generating function of walks in a graph, that is somewhat similar (and, as we will see later, this is far from a coincidence) to the polynomial $P(x, y)$ studied when using labeled walks. Namely, we associate again with every vertex $v \in V(G)$ a variable x_v, and with every edge $e \in E(G)$ a variable y_e. Then, we define the formal polynomial

$$\varphi = \sum_{w_1,\ldots,w_k \in \mathscr{W}(G)} \prod_{i=1}^{k} x_{w_i} \prod_{j=1}^{k-1} y_{w_j w_{j+1}} \,. \tag{3.8}$$

The structural insight here is now the following: If we could somehow make those monomials disappear that contain a factor x_v^2, for some $v \in V(G)$, and simultaneously make sure that the other monomials yield a non-zero contribution, then the result being non-zero is again equivalent to G containing a k-path.

There are now two obstacles to overcome: First, we want to make this work *somehow*, *i.e.*, find things to plug into the variables that satisfy this requirement. Then, we would like to make this work *efficiently*, *i.e.*, finding a way how to actually compute the sum in Eq. (3.8).

3.4.2. Computing the Walk-Sum

Let us first get the easiest part of this out of the way, which is the evaluation of Eq. (3.8) at given algebra elements $r_1, \ldots, r_{n+m}$ with a reasonable number of algebra operations:

Proposition 3.4.1. *For given $r_1, \ldots, r_{n+m}$ from some commutative algebra, the sum $F(r_1, \ldots, r_{n+m})$ can be evaluated using $O(km)$ ring operations.*

Proof sketch. This is a straightforward generalization of the well-known undergraduate exercise of proving that the powers of the adjacency matrix of a graph enumerate the walks in this graph, and that (given a sparse representation of the matrix), the final sum over all walks can be computed using $O(km)$ multiplications and additions. □

Of course, this Lemma says nothing about the *actual* cost of implementing those ring operations using ordinary bit operations. Since this of course depends quite heavily on the algebra from which the evaluation points come, we now turn to the first obstacle described above, namely finding suitable objects to plug in.

3.4.3. Group Algebras

Let F be a field and let M be a monoid with multiplication $*$. We denote with $F[M]$ the *monoid algebra of M over F*. If M is actually a group, we call $F[M]$ the *group algebra of M over F*. That is, $F[M]$ is the set of all finite formal linear combinations of elements from M with coefficients in F. An element of $F[M]$ is thus of the form $\sum_{m \in M} r_m \cdot m$, with only finitely many of the $r_m \in F$ non-zero. Elements from $F[M]$ admit a natural point-wise addition and scalar multiplication. Multiplication in

$F[M]$, written $\bullet$, is defined by the distributive law,

$$\left(\sum_{m\in M} c_m \cdot m\right) \bullet \left(\sum_{m\in M} d_m \cdot m\right) = \left(\sum_{g,h\in G} (c_g \cdot d_h) \cdot (g * h)\right),$$

which is again an element of $F[M]$.

As the name suggests, the monoid algebra $F[M]$ is indeed an F-algebra, and is of dimension $|M|$. Usually, multiplication and addition in the ground field F, the monoid M, and the group algebra $F[M]$ are all denoted by $\cdot$ and $+$.

The Group Algebra of $\mathbb{Z}_2^k$

Denote with $\mathbb{Z}_2^k$ the additive group of k-dimensional vectors modulo two, and write χ_i for the i-th element of the standard basis of $\mathbb{Z}_2^k$. Despite the group being commutative, we stick with *multiplicative* notation when referring to the group operations, *i.e.*, for $x, y \in \mathbb{Z}_2^k$, the product $xy \in \mathbb{Z}_2^k$ refers to their point-wise *addition* modulo two, and 1 is the all-zeroes vector of $\mathbb{Z}_2^k$. In a word, we think of $\mathbb{Z}_2^k$ as $\{-1, 1\}^k$ with point-wise multiplication as operation.

This might admittedly create some confusion right away, but bearing with this pay off later to spare us some even more confusing notation. For a concrete example, if $k = 4$, then the element $(0, 0, 1, 1) \in \mathbb{Z}_2^k$ can be written as the product $\chi_3\chi_4$, and in general, if S is the support of an element $\mathbb{Z}_2^k$, then $\prod_{s\in S} \chi_s$ is the corresponding representation as a product of the generators χ_i, and for any such set S we write $\chi_S \in \mathbb{Z}_2^k$ for the vector having one-entries at indices of S, and zeroes everywhere else. Consequently, for every element $\chi_S \in \mathbb{Z}_2^k$, we find $\chi_S^2 = 1$.

Let again F be a field of characteristic two, with a number of elements that we shall fix later, again with an eye towards applying the DeMillo–Lipton–Schwartz–Zippel Lemma 3.3.3. Consider the group algebra $F[\mathbb{Z}_2^k]$, which is commutative since $\mathbb{Z}_2^k$ is. In $F[\mathbb{Z}_2^k]$, we can consider the elements $\nu_S = 1 + \chi_S$ for all χ_S.[4] Observe that, since F is of

[4] This is now where writing $\mathbb{Z}_2^k$ in a multiplicative manner spares us having to distinguish two entirely different kinds of addition.

characteristic two, we have $\nu_S^2 = 1 + 1 = 0$. Trivially, all the ν_S are linearly independent over F as S varies. This is already half the battle: We have found linearly independent elements of some commutative algebra that square to zero. We might now randomly assign to the x_i in Eq. (3.8) the elements ν_S for random choices of S, and certainly, φ would evaluate to 0 if G doesn't have a k-path, as all those monomials containing a term x_i^2 (and hence ν_S^2 for some S) become zero.

It remains to argue that for a random such assignment of sets to the x_i, any product of k distinct x_i is non-zero with high probability. This argument is formally similar to the reasoning during the analysis of Color-Coding, where we calculated the probability that k distinct vertices receive k distinct colors. This time, however, we have to argue what the probability is not only that the "colors," *i.e.*, elements ν_S, are distinct, but that they have non-zero product. As it turns out, this case is much more benign. But first, let us investigate the exact conditions of such a product becoming zero, following [Kou08].

Lemma 3.4.2. *Let $\mathcal{S}$ be a collection of subsets of $\{1, \ldots, k\}$. If the vectors $\{\chi_S\}_{S \in \mathcal{S}} \in \mathbb{Z}_2^k$ are linearly dependent over $\mathbb{Z}_2$, then*

$$\prod_{S \in \mathcal{S}} \nu_S = 0\,.$$

Before the simple proof, recall that we write addition in $\mathbb{Z}_2^k$ multiplicatively.

Proof. Being linearly independent over $\mathbb{Z}_2$ just that there is some subset $\mathcal{T} \subseteq \mathcal{S}$ such that the vectors corresponding to $\mathcal{T}$ sum to $0 \in \mathbb{Z}_2^k$, which translates to multiplicative notation as

$$\prod_{T \in \mathcal{T}} \chi_T = 1\,.$$

For arbitrary $\mathcal{U} \subseteq \mathcal{T}$, we can multiply this equation with $\prod_{D \in \mathcal{T} \Delta \mathcal{U}} \chi_D$

to obtain

$$\prod_{U\in\mathcal{U}} \chi_S = \prod_{D\in\mathcal{T}\Delta\mathcal{U}} \chi_D \,. \tag{3.9}$$

Consider now the subproduct

$$\prod_{T\in\mathcal{T}} \nu_T = \prod_{T\in\mathcal{T}} (1+\chi_T) = \sum_{\mathcal{U}\subseteq\mathcal{T}} \prod_{U\in\mathcal{U}} \chi_U \,.$$

Using Eq. (3.9), we find that each of the inner products appear twice in the outer sum, and—since we work over characteristic two—cancel out. It follows that the subproduct, and hence the entire product, is zero, as claimed. □

Let us introduce the following shorthand: We write $\mathbb{1}$ to denote

$$\mathbb{1} = \sum_{S\subseteq\{1,\ldots,k\}} \chi_S \,,$$

justified by the fact that $\mathbb{1} \in F[\mathbb{Z}_2^k]$, considering $F[\mathbb{Z}_2^k] \cong F^{2^k}$ as a 2^k-dimensional vector space over F, is the all-ones vector.

Lemma 3.4.3 ([Kou08])**.** *Let $S_1, \ldots, S_k$ be a collection of k subsets of $\{1, \ldots, k\}$. If the vectors $\{\chi_{S_i}\}_{i=1}^k$ are linearly independent over $\mathbb{Z}_2$, then*

$$\prod_{i=1}^{k} \nu_{S_i} = \mathbb{1}.$$

Proof. Clearly,

$$\prod_{i=1}^{k} \nu_{S_i} = \prod_{i=1}^{k} (1+\chi_{S_i}) = \sum_{J\subseteq\{1,\ldots,k\}} \prod_{j=1}^{k} \chi_{S_j} \,.$$

Since the dimension of $\mathbb{Z}_2^k$ over $\mathbb{Z}_2$ is k, any k linearly independent (over $\mathbb{Z}_2$) vectors yield a unique way to write every element of $\mathbb{Z}_2^k$ as a linear combination of them. Over $\mathbb{Z}_2$, a linear combination is just the sum over some subset of them, and every such subset yields a distinct

element of $\mathbb{Z}_2$. Thus,

$$\sum_{J\subseteq\{1,\dots,k\}} \prod_{j=1}^{k} \chi_{S_j} = \sum_{K\subseteq\{1,\dots,k\}} \chi_K = \mathbb{1}\,.$$

□

To summarize:

Proposition 3.4.4 ([Kou08])**.** *Let $v_1,\dots,v_k \in \mathbb{Z}_2^k$ be any k vectors. Then, $\prod_{i=1}^{k} v_i$ zero if and only if $\{v_i\}_{i=1}^{k}$ is linearly dependent, and equal to $\mathbb{1}$ otherwise.*

This raises the question: What's the probability that k *randomly* chosen vectors over $\mathbb{Z}_2^k$ are linearly independent? The answer to this question is neither difficult nor long, and we include it for completeness; Williams [Wil09] gives as reference Lemma 6.3.1 in [BK95].

Lemma 3.4.5. *The probability that k randomly chosen vectors in $\mathbb{Z}_2^k$ are linearly independent is at least $1/4$.*

Proof. Let us choose $v_1,\dots,v_k$ one after another. Having picked i vectors, they span a linear subspace of $\mathbb{Z}_2^k$ of dimension at most i, *i.e.*, it contains at most 2^i elements. Therefore, when picking the $(i+1)$st vector, there are at least $2^k - 2^i$ choices for v_{i+1} for it to not be contained in the subspace spanned by $v_1,\dots,v_i$. In total, the probability to obtain k linearly independent vectors is therefore at least $\prod_{i=1}^{k}\left(1-\frac{1}{2^{k-i}}\right)$, which is bounded from below by a constant strictly greater than $1/4$. This follows from standard results on the q-analogon of the so-called Pochhammer symbol. □

After collecting all these basic properties, let us now come back to our initial object, the walk generating polynomial φ. Let $\{v_w\}_{w\in V(G)}$ be any n vectors in $\mathbb{Z}_2^k$. Consider the evaluation of φ at the $\{v_w\}_w$, while leaving the y_e intact; that is, $F(\{v_w\}_w, \{y_e\}_e)$. By definition and

using Proposition 3.4.4, this is equal to

$$\left(\sum_{\substack{w_1,\ldots,w_k \in \mathscr{P}(G), \\ \{v_{w_i}\}_{i=1}^k \text{ are linearly independent}}} \prod_{i=1}^{k-1} y_{w_i w_{i+1}} \right) \cdot \mathbb{1} \tag{3.10}$$

If G has at least one k-path, then the probability that at least one of the products in Eq. (3.10) is non-zero is at least $1/4$, by Lemma 3.4.5.

Denote the assignment $V \to \mathbb{Z}_k^2, w \mapsto v_w$ as α, *i.e.*, $\alpha(w) = v_w$. For some fixed α, consider now the coefficient C_α of $\mathbb{1}$ in Eq. (3.10),which is an element of $F[\{y_e\}_e]$, *i.e.*,

$$C_\alpha(\{y_e\}_e) = \sum_{\substack{w_1,\ldots,w_k \in \mathscr{P}(G), \\ \{v_{w_i}\}_{i=1}^k \text{ are linearly independent}}} \prod_{i=1}^{k-1} y_{w_i w_{i+1}} \in F[\{y_e\}_e] \,.$$

Of course, since a path in a directed graph is determined by its edges, the monomials in the y_e are linearly independent, so that the polynomial $C_\alpha(\{y_e\}_e)$ is non-zero if and only if there is some path whose vertex variables are mapped to a set of linearly independent vectors. To be precise:

Lemma 3.4.6. *If G has a k-path, let $\alpha = \{v_w\}_{v \in V(G)}$ be such that for at least one of the k-paths of G, say P, its set of vertices $V(P)$ is such that the associated set of vectors $\{v_w\}_{w \in V(P)}$ is linearly independent. Then, $C_\alpha \neq 0$.*

Again with the DeMillo–Lipton–Schwartz–Zippel Lemma 3.3.3, we obtain that if we pick $|F| > 4k$ and $C_\alpha \neq 0$, then for random $\lambda_1, \ldots, \lambda_m \in F$, $C_\alpha(\lambda_1, \ldots, \lambda_m)$ is non-zero with probability at least $1/2$.

Putting all of the above together, we obtain:

Proposition 3.4.7. *If G has a k-path, then for $v_1, \ldots, v_n \in \mathbb{Z}_2^k$ and scalars $\lambda_1, \ldots, \lambda_m$ all chosen uniformly at random, $F(v_1, \ldots, v_n, \lambda_1, \ldots, \lambda, m)$ is non-zero with probability at least $1/8$.*

This completely deals with the correctness of the approach, and we will now have to deal with the question of how to implement the approach in an efficient way.

3.4.4. The Algorithm

We saw in Proposition 3.4.1 that evaluating φ can be done using $O(km)$ operations in the algebra $F[\mathbb{Z}_2^k]$. It remains to prove how to compute these operations quickly. We can represent each element of $F[\mathbb{Z}_2^k]$ using 2^k coefficients, one for each element of the group $\mathbb{Z}_2^k$. This makes addition trivially implementable using 2^k field operations. As for multiplication, it is easy to see (and we will come across this again later) that multiplication in group algebras of $\mathbb{Z}_2^k$ is the same as performing a subset convolution, as studied by Björklund *et al.* [Bjö+07], and can be computed using $O(k^2 2^k)$ field operations. Finally, representing and computing with elements of F can be done in time polylogarithmic in the number of field elements in F by standard methods, adding a $\mathrm{poly}(k)$-factor to the running time. To summarize:

Proposition 3.4.8. *Given $x, y \in F[\mathbb{Z}_2^k]$ as lists of coefficients, their sum and product $x+y, xy \in F[\mathbb{Z}_2^k]$ can each be computed using $O(2^k \operatorname{poly}(k))$ bit operations.*

In particular, combining this with Proposition 3.4.1, we obtain:

Corollary 3.4.9. *Given $v_1, \ldots, v_n \in \mathbb{Z}_2^k$ and $\lambda_1, \ldots, \lambda_m \in F$, φ can be evaluated at these points in time $2^k \cdot \mathrm{poly}(k) \cdot m$.*

Taking all the results from this subsection together now yields the following algorithm.

Algorithm AF (*Detect k-path using algebraic fingerprints.*) *Given directed graph G and integer k, this algorithm determines if G contains a k-path.*

AF1 (Initialize.) Construct the field F with more than $4k$ elements.

AF2 (Set up trial.) Pick $v_1, \ldots, v_n \in \mathbb{Z}_2^k$ and $\lambda_1, \ldots, \lambda_m \in F$ uniformly at random.

AF3 (Evaluate φ.) Use the algorithm from Corollary 3.4.9, evaluate φ at $v_1, \ldots, v_n, \lambda_1, \ldots, \lambda_m$.

AF4 (Decide.) If φ evaluated to 0, output 'no.' Otherwise, output 'yes.' □

Theorem 3.4.10 ([Kou08; Wil09])**.** *Algorithm AF decides the existence of a k-path in a given directed graph with n nodes and m edges in time $2^k \cdot \mathrm{poly}(k) \cdot m$. The algorithm has a constant one-sided error probability of claiming the absence of such a path in a graph that actually does have a k-path.*

Proof. Correctness and the error probability are established by Proposition 3.4.7, the running time bound follows from Corollary 3.4.9. □

Counting and Determinism

We note here two limitations of the algebraic approaches. Since the DeMillo–Lipton–Schwartz–Zippel Lemma plays a central rôle in the algebraic methods, *i.e.*, the group-algebraic approach as well as the labeled walks (which are arguably combinatorial in flavor, but in the end rely on polynomial identity testing nonetheless), their derandomization seems highly non-trivial, and it is by no means clear how to obtain an algebraic algorithm (*i.e.*, one relying on evaluations of polynomials or the like—of course, this term is not sharply defined) that can be used to produce even moderately fast *deterministic* decision algorithms.

Similarly, the methods seem to require that the ground field have characteristic two. The fact that $1+1 = 0$ in this setting makes *counting* become rather futile rather quick, and it is not clear how to circumvent this limitation. We shall see how to do this in the next chapter.

4. Extensor-Coding Longest Paths

Cancel me not—for what then shall remain?

Stanisław Lem, *The Cyberiad*

We will now present our new method, generalizing and unifying all the ones presented so far. Crucially, we will find: A path is just a walk that does not vanish in the exterior algebra. This observation leads us to a new approach for algebraic graph algorithms for the k-path problem. As noted, our approach generalizes and unifies previous techniques in a clean fashion, including the Color-Coding method of Alon, Yuster, and Zwick [AYZ95] and the algebraic-fingerprinting idea of Koutis [Kou08]. Color-Coding yields a randomized algorithm for approximately counting k-paths [Alo+08] that runs in time $(2e)^k \operatorname{poly}(n)$ (cf. Theorem 3.1.5). We improve the running time to $4^k \operatorname{poly}(n)$, addressing an open problem in the survey article of Koutis and Williams [KW15]. Our approach applies not only to paths, but also to other subgraphs of bounded pathwidth.

In hindsight, it is obvious that the exterior algebra enjoys exactly the properties needed for the k-path problem. Thus, it seems strange that this construction has eluded algorithms designers for so long. But as the eminent combinatorialist Gian-Carlo Rota observed in 1997, "[t]he neglect of the exterior algebra is the mathematical tragedy of our century," [Rot97] so we are in good company.

The exterior algebra is also called alternating algebra, extended algebra, or Grassmann algebra after its 19th century discoverer. It is treated extensively in any modern textbook on algebra, and has applications in many fields, from differential geometry and representation theory to

theoretical physics. Conceptually, our contribution is to identify yet another entry in the growing list of applications of the exterior algebra, inviting the subgraph isomorphism problem to proudly take its place between simplicial complexes and supernumbers.

4.1. Results and Related Work

Longest Path. The Longest Path problem is the optimization problem to find a longest (simple) path in a given graph. Clearly, this problem generalizes the NP-hard Hamiltonian path problem [GJ79]. As before, we consider the decision version, the k-path problem, in which we wish to decide existence of a path of length k in a given graph G. It was proved fixed-parameter tractable *avant la lettre* [Mon85], and a sequence of both iterative improvements and conceptual breakthroughs [Bod93; AYZ95; Bjö14; Kne+06; Che+09; Fom+16; Wil09] have lead to the current state-of-the-art for undirected graphs: a randomized algorithm by Björklund *et al.* [Bjö+17a] in time $1.66^k \cdot \mathrm{poly}(n)$. For details on these methods, we refer the reader to Chap. 3. For directed graphs, the fastest known randomized algorithm is by Koutis and Williams [KW16] in time $2^k \cdot \mathrm{poly}(n)$, whereas the fastest deterministic algorithm is due to Zehavi [Zeh15] in time $2.5961^k \cdot \mathrm{poly}(n)$.

Subgraph isomorphism. The subgraph isomorphism problem generalizes the k-path problem and is one of the most fundamental graph problems [Coo71; Ull76]: Given two graphs H and G, decide whether G contains a subgraph isomorphic to H. This problem and its variants have a vast number of applications, covering areas such as statistical physics, probabilistic inference, and network analysis [Mil+02]. For example, such problems arise in the context of discovering *network motifs*, small patterns that occur more often in a network than would be expected if it was random. Thus, one is implicitly interested in the counting version of the subgraph isomorphism problem: to compute the *number* of subgraphs of G that are isomorphic to H. Through network motifs, the problem of counting subgraphs has found applica-

tions in the study of gene transcription networks, neural networks, and social networks [Mil+02]. Consequently, there is a large body of work dedicated to algorithmic discovery of network motifs [GK07; Alo+08; OSM09; Kas+09; SS05; Che+06; Kas+04; Wer06; Sch+15]. For example, Kibriya and Ramon [KR13; Ram+14] use the ideas of Koutis and Williams [KW16] to enumerate all trees that occur frequently.

Counting subgraphs exactly. The complexity of exact counting is often easier to understand than the corresponding decision or approximate counting problems. For instance, the counting version of the famous dichotomy conjecture by Feder and Vardi [FV93; FV98] was resolved by Bulatov [Bul08; Bul13] almost a decade before proofs were announced for the decision version by Bulatov [Bul17] and Zhuk [Zhu17]. A similar phenomenon can be observed for the parameterized complexity of the subgraph isomorphism problem, the counting version of which is much better understood than the decision or approximate counting versions: The problem of counting subgraphs isomorphic to H is fixed-parameter tractable if H has a vertex cover of bounded size [WW13] (also cf. [KLL13; CM14b; CDM17]), and it is #W[1]-hard whenever H is from a class of graphs with unbounded vertex cover number [CM14b; CDM17], and thus it is not believed to be fixed-parameter tractable in the latter case. In particular, this is the case for counting all k-paths in a graph. The fastest known general-purpose algorithm [CDM17] for counting H-subgraphs in an n-vertex graph G runs in time $k^{O(k)}n^{t^*+1}$ where k is the number of vertices of H and t^* is the largest treewidth among all homomorphic images of H. For some special choices of the pattern graph H, the branchwidth-based bounds of Austrin *et al.* [AKK18] improve over the treewidth bounds of Curticapean *et al.* In particular, this is the case for paths of lengths 7, 8 or 9; more generally, whenever the branchwidth of H is bounded by $2(t+1)/\omega$, where t is the treewidth of H.

New Results. For finite directed or undirected graphs H and G, let $\mathrm{Sub}(H, G) \in \mathbb{N}$ be the number of (not necessarily induced) subgraphs of G that are isomorphic to H. The main algorithmic result in this

chapter is a randomized algorithm that computes an approximation to this number.

Theorem 4.1.1 (Approximate subgraph counting)**.** *There is a randomized algorithm that is given two graphs H and G, and a number $\varepsilon > 0$ to compute an integer $\tilde{N}$ such that, with probability* 99%,

$$(1-\varepsilon)\cdot \mathrm{Sub}(H,G) \leq \tilde{N} \leq (1+\varepsilon)\cdot \mathrm{Sub}(H,G)\,. \tag{4.1}$$

This algorithm runs in time $\varepsilon^{-2}\cdot 4^k n^{\mathrm{pw}(H)+1}\cdot \mathrm{poly}(k)$, where H has k vertices and pathwidth $\mathrm{pw}(H)$, and G has n vertices.

Our algorithm works for directed and undirected graphs with the same running time (in fact, undirected graphs are treated as being bi-directed). An algorithm such as the one in Theorem 4.1.1 is called a fixed-parameter tractable randomized approximation scheme (FPT-RAS) for Sub . The notion of an FPT-RAS was defined by Arvind and Raman [AR02], who use a sampling method based on Karp and Luby [KL83] to obtain a version of Theorem 4.1.1 with an algorithm that runs in time $\exp(O(k\log k))\cdot n^{\mathrm{tw}(H)+O(1)}$. For the special cases of paths and cycles, Alon and Gutner [AG09; AG10] are able to combine the Color-Coding technique by Alon, Yuster, and Zwick [AYZ95] with balanced families of hash functions to obtain an algorithm for approximately counting paths or cycles in time $\exp(O(k\log\log k))\cdot n\log n$. Alon *et al.* [Alo+08], in turn, use the Color-Coding technique to obtain the first singly-exponential time version of Theorem 4.1.1, in particular with an algorithm running in time $\varepsilon^{-2}\cdot(2e)^k\cdot n^{\mathrm{tw}(H)+O(1)}$. Theorem 4.1.1 was the fastest known algorithm to approximately count subgraphs of small pathwidth, until very recently, Björklund *et al.* [Bjö+19] improved upon our results, and gave a polynomial-space, deterministic algorithm. It has an almost identical runtime, with the only difference being a slightly worse dependency on ε. Their algorithm extends to graphs of bounded treewidth.

When we are promised that G contains not too many subgraphs isomorphic to H, we obtain the following deterministic algorithm.

Theorem 4.1.2 (Detecting subgraphs when there are few)**.** *There is a deterministic algorithm that is given two graphs H and G to decide whether G has a subgraph isomorphic to H, with the promise that G has at most $C \in \mathbb{N}$ such subgraphs. This algorithm runs in time $O(C^2 2^k n^{\mathrm{pw}(H)+O(1)})$, where the number of vertices of H is k and the number of vertices of G is n.*

Without the promise on the number of subgraphs, Fomin *et al.* [Fom+12b] detect subgraphs in randomized time $\tilde{O}(2^k n^{\mathrm{tw}(H)+1})$ and Fomin *et al.* [Fom+16] do so in deterministic time $2.619^k n^{O(\mathrm{tw}(H))}$. For $C \leq O(1)$, or $C \leq \mathrm{poly}(n, k)$ when ignoring polynomial factors, we thus match the running time of the fastest randomized algorithm, but do so deterministically, and for $C \leq O(1.144^k)$, our algorithm is the fastest deterministic algorithm for this problem. For the interesting special case of paths, the running time of the fastest deterministic algorithm for undirected or directed k-paths (without promise) is $2.5961^k \cdot \mathrm{poly}(n)$ by Zehavi [Zeh15], which we improve upon if $C \leq O(1.139^k)$.

Our method also applies to the problem of detecting whether a multivariate polynomial contains a multilinear term.

Theorem 4.1.3 (Detecting multilinear terms)**.** *Given an algebraic circuit C over $\mathbb{Z}[\zeta_1, \ldots, \zeta_n]$ and a number k, we can detect whether the polynomial $C(\zeta_1, \ldots, \zeta_n)$ has a degree-k multilinear term in randomized time $4.32^k \cdot |C| \cdot \mathrm{poly}(n)$.*

Using algebraic fingerprinting with elements from a group algebra, Koutis and Williams [Kou08; KW16] can do this in randomized $2^k \cdot \mathrm{poly}(n)$ time for monotone algebraic circuits, that is, circuits that do not involve negative values. Working over an algebra whose ground field of characteristic 0, we are able to remove the requirement that the circuit is free of cancellations in Theorem 4.1.3. To the best of our knowledge, this was the first fixed-parameter tractable algorithm for the problem of detecting a k-multilinear term in the polynomial computed by a general algebraic circuit. Using refined algebraic techniques, Arvind *et al.* and Pratt [Arv+18b; Pra18; Pra19] independently improved upon this, with the current record bound now using $4.08^k \cdot \mathrm{poly}(n)$ time, and polynomial space.

Our algorithm uses Color-Coding and performs the computation in the exterior algebra over $\mathbb{Q}^k$. To reduce the running time from $2^k e^k \cdot \mathrm{poly}(n)$ to $4.32^k \cdot \mathrm{poly}(n)$, we use an idea of Hüffner, Wernicke, and Zichner [HWZ08], who improved Color-Coding by using $1.3 \cdot k$ instead of only k different colors.

Related hardness results. Under the exponential-time hypothesis (ETH) by Impagliazzo and Paturi [IP01], the running time of the algorithm in Theorem 4.1.1 is optimal in the following asymptotic sense: The exponent of n cannot be improved since $f(k)n^{o(t)}$ time is impossible even in the case that H is a k-clique [Che+05], where $t = k-1$. Likewise, a running time of the form $\exp(o(k)) \cdot \mathrm{poly}(n)$ is impossible even in the case that $t = 1$, since this would imply an $\exp(o(n))$ time algorithm for the Hamiltonian cycle problem and thereby contradict ETH [IPZ01]. Moreover, the factor ε^{-2} in the running time stems from an application of Chebyshev's inequality and is unlikely to be avoidable.

4.1.1. The Walk-Sum

We wil now discuss in slightly higher generality an object related to Eq. (3.8), where we let $\mathscr{W} = \mathscr{W}(G), \mathscr{P} = \mathscr{P}(G)$, where G is the input graph.

Let R be a ring and consider a mapping $\xi\colon V(G) \cup E(G) \to R$. The *walk-sum* $f(G;\xi)$ of ξ is defined via

$$f(G;\xi) = \sum_{w_1 \dots w_k \in \mathscr{W}} \xi(w_1)\xi(w_1w_2)\xi(w_2) \cdots \xi(w_{k-1})\xi(w_{k-1}w_k)\xi(w_k)\,, \tag{4.2}$$

evaluated in R. As a matter of folklore, the walk-sum can be evaluated with $O(kn^2)$ operations over R using using a well-known connection with powers of the adjacency matrix. This is a reformulation of Proposition 3.4.1. Observe that we do *not* require R be commutative, in contrast

to the previously employed polynomial rings.

$$f(G;\xi) = \Big(1 \dots 1\Big) \cdot A^{k-1} \cdot \begin{pmatrix} \xi(v_1) \\ \vdots \\ \xi(v_n) \end{pmatrix}, \tag{4.3}$$

where A is the $n \times n$ matrix whose vw-entry is given by

$$a_{vw} = \begin{cases} \xi(v)\xi(vw), & \text{if } vw \in E(G); \\ 0, & \text{otherwise.} \end{cases} \tag{4.4}$$

Note that the expression for $f(G;\xi)$ in (4.3) can be evaluated in such a way that every product in R has the form $x \cdot y$ where y belongs to the range of ξ (rather than all of R). Moreover, we assume input graphs to be given as adjacency lists, in which case the expression in (4.3) can be evaluated with $O(k(n+m))$ operations over R, since the product of an m-sparse matrix and a vector can be computed with $O(n+m)$ operations over R. If $\xi\colon V(G) \to R$ is a partial assignment, we silently extend it to a full assignment by setting the remaining variables to $1 \in R$.

4.2. The Exterior Algebra

4.2.1. Concrete Definition

We now give an elementary and very concrete definition of the exterior algebra, and recall the properties of the wedge product. Readers familiar with this material can skip Sect. 4.2.1.

Let F be a field, k be a positive integer, and let $\mathbf{e}_1, \dots, \mathbf{e}_k$ be the canonical basis of the k-dimensional vector space F^k. Every element a of F^k is a linear combination $a_1\mathbf{e}_1 + \cdots + a_k\mathbf{e}_k$ with field elements $a_1, \dots, a_k \in F$. We sometimes write a as the column vector $(a_1, \dots, a_k)^T$. Addition and scalar multiplication are defined in the usual way.

We extend F^k to a much larger, 2^k-dimensional vector space $\Lambda(F^k)$

as follows. Each basis vector $\mathbf{e}_I$ of $\Lambda(F^k)$ is defined by a subset I of indices from $\{1, \ldots, k\}$. The elements of $\Lambda(F^k)$ are called *extensors*. Each element is a linear combination $\sum_{I \subseteq \{1,\ldots,k\}} a_I \mathbf{e}_I$ of basis vectors. We turn $\Lambda(F^k)$ into a vector space by defining addition and scalar multiplication in the natural fashion. For instance, if F is the rationals, typical elements in $\Lambda(F^k)$ with $k = 3$ are $x = 3\mathbf{e}_{\{1,2\}} - 7\mathbf{e}_{\{3\}}$ and $y = \mathbf{e}_{\{1\}} + 2\mathbf{e}_{\{3\}}$ and we have $x + 2y = 3\mathbf{e}_{\{1,3\}} + 2\mathbf{e}_{\{1\}} - 3\mathbf{e}_{\{3\}}$. By confusing $\mathbf{e}_i$ with $\mathbf{e}_{\{i\}}$ for $i \in \{1, \ldots, k\}$, we can view F^k as a subspace of $\Lambda(F^k)$ spanned by the singleton basis vectors. This subspace is sometimes called $\Lambda^1(F^k)$, the set of *vectors*. The element $\mathbf{e}_\emptyset$ is just 1 in the underlying field, so $\Lambda^0(F^k) = F$. In general, $\Lambda^i(F^k)$ is the set of extensors spanned by basis vectors $\mathbf{e}_I$ with $|I| = i$, sometimes called *i-vectors*. Of particular interest is $\Lambda^2(F^k)$, the set of *blades* (also called bivectors).

To turn $\Lambda(F^k)$ into an *algebra*, we define a multiplication $\wedge$ on the elements of $\Lambda(F^k)$. The multiplication operator we define is called the *wedge product* (also called exterior or outer product) and the resulting algebra is called the *exterior algebra*. We require $\wedge$ to be associative

$$(x \wedge y) \wedge z = x \wedge (y \wedge z)$$

and bilinear

$$\begin{aligned} x \wedge (a \cdot y + z) &= a \cdot x \wedge y + x \wedge z\,, \\ (x + a \cdot y) \wedge z &= x \wedge z + a \cdot y \wedge z\,, \end{aligned}$$

for all $a \in F$ and $x, y, z \in \Lambda(F^k)$. Thus, it suffices to define how $\wedge$ behaves on a pair of basis vectors $\mathbf{e}_I$ and $\mathbf{e}_J$. If I and J contain a common element, then we set $\mathbf{e}_I \wedge \mathbf{e}_J = 0$. Otherwise, we set $\mathbf{e}_I \wedge \mathbf{e}_J = \pm\mathbf{e}_{I \cup J}$; it only remains to define the sign, which requires some delicacy. (The intuition is that we want $\wedge$ to be anti-commutative on F^k, that is, $x \wedge y = -y \wedge x$ for $x, y \in F^k$.) Write $I = \{i_1, \ldots, i_r\}$ and $J = \{j_1, \ldots, j_s\}$, both indexed in increasing order. Then we define

$$\mathbf{e}_I \wedge \mathbf{e}_J = \operatorname{sgn}(I, J)\mathbf{e}_{I \cup J}\,,$$

where $\operatorname{sgn}(I, J)$ is the sign of the permutation that brings the sequence $i_1, \ldots, i_r, j_1, \ldots, j_s$ into increasing order.

For instance, if $\max I < \min J$, then there is nothing to permute, so $\mathbf{e}_1 \wedge \mathbf{e}_2 = \mathbf{e}_{\{1,2\}}$. Consequently, we now abandon the set-indexed notation $\mathbf{e}_{\{i_1,\ldots,i_r\}}$ (where $i_1 < \cdots < i_r$) and just write $\mathbf{e}_{i_1} \wedge \cdots \wedge \mathbf{e}_{i_r}$ instead. It is also immediate that $\mathbf{e}_1 \wedge \mathbf{e}_2 = -\mathbf{e}_2 \wedge \mathbf{e}_1$. In general, we can multiply basis vectors using pairwise transpositions and associativity, *e.g.*, $(\mathbf{e}_1 \wedge \mathbf{e}_3 \wedge \mathbf{e}_6) \wedge (\mathbf{e}_2 \wedge \mathbf{e}_4) = -\mathbf{e}_1 \wedge \mathbf{e}_3 \wedge \mathbf{e}_2 \wedge \mathbf{e}_6 \wedge \mathbf{e}_4 = \mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3 \wedge \mathbf{e}_6 \wedge \mathbf{e}_4 = -\mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3 \wedge \mathbf{e}_4 \wedge \mathbf{e}_6$.

4.2.2. Properties

The wedge product on F^k has the following properties:

(W1) *Alternating on vectors.* By its definition, the wedge product enjoys anticommutativity on the basis vectors of F^k, which is to say $\mathbf{e}_i \wedge \mathbf{e}_j = -\mathbf{e}_j \wedge \mathbf{e}_i$. Employing bilinearity, this directly translates to any two vectors $x, y \in F^k$, meaning $x \wedge y = -y \wedge x$ holds, whereby $x \wedge x$ vanishes.

(W2) *Alternating on decomposable extensors.* An extensor $x \in \Lambda(F^k)$ is *decomposable* if there are vectors $v_1, \ldots, v_r \in F^k$ satisfying $x = v_1 \wedge \cdots \wedge v_r$. Every extensor in $\Lambda^i(F^k)$ is decomposable for $i \in \{0, 1, k-1, k\}$, but not all extensors are decomposable: $\mathbf{e}_1 \wedge \mathbf{e}_2 + \mathbf{e}_2 \wedge \mathbf{e}_4 \in \Lambda^2(F^4)$ is an example. The previous property extends to decomposable vectors: If the extensors $x_1, \cdots, x_r$ are decomposable and two of them are equal, then it follows from Property (W1) that their wedge product $x_1 \wedge \cdots \wedge x_r$ vanishes.

(W3) *Determinant on $F^{k\times k}$.* For $k = 2$ write $x, y \in F^2$ as column vectors (x_1, x_2) and (y_1, y_2). Elementary calculations show $x \wedge y = (x_1 y_2 - y_1 x_2) \cdot \mathbf{e}_1 \wedge \mathbf{e}_2$, and we recognize the determinant of the 2×2-matrix whose columns are x and y. This is not a coincidence. Since $\Lambda^k(F^k)$ is linearly isomorphic to F—indeed, $\Lambda^k(F^k) = F \cdot (\mathbf{e}_1 \wedge \cdots \wedge \mathbf{e}_k)$—we can understand the map taking $(x_1, \ldots, x_k)$ to $x_1 \wedge \cdots \wedge x_k \in \Lambda^k(F^k) \cong F$ as a multilinear form,

which by virtue of the previous properties is alternating and sends $(\mathbf{e}_1, \ldots, \mathbf{e}_k)$ to 1. These properties already characterize the determinant among the multilinear forms. With this, we have arrived at a fundamental property of the exterior algebra. Let $x_1, \ldots, x_k \in F^k$ and write

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{k1} \end{pmatrix}, \qquad \ldots, \qquad x_k = \begin{pmatrix} x_{1k} \\ \vdots \\ x_{kk} \end{pmatrix}.$$

The wedge product of $x_1, \ldots, x_k$ exhibits a determinant:

$$x_1 \wedge \cdots \wedge x_k = \det \begin{pmatrix} x_{11} & \cdots & x_{1k} \\ \vdots & \ddots & \vdots \\ x_{k1} & \cdots & x_{kk} \end{pmatrix} \cdot \mathbf{e}_{[k]}\,, \tag{4.5}$$

where we use the shorthand $\mathbf{e}_{[k]}$ for the highest-grade basis extensor $\mathbf{e}_1 \wedge \cdots \wedge \mathbf{e}_k$.

To avoid a misunderstanding: Neither of these properties extends to all of $\Lambda(F^k)$. For instance, if $x = \mathbf{e}_1 \wedge \mathbf{e}_3 + \mathbf{e}_2$ then $x \wedge x = (\mathbf{e}_1 \wedge \mathbf{e}_3 + \mathbf{e}_2) \wedge (\mathbf{e}_1 \wedge \mathbf{e}_3 + \mathbf{e}_2) = \mathbf{e}_1 \wedge \mathbf{e}_3 \wedge \mathbf{e}_1 \wedge \mathbf{e}_3 + \mathbf{e}_1 \wedge \mathbf{e}_3 \wedge \mathbf{e}_2 + \mathbf{e}_2 \wedge \mathbf{e}_1 \wedge \mathbf{e}_3 + \mathbf{e}_2 \wedge \mathbf{e}_2 = 0 - \mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3 - \mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3 + 0 = -2 \cdot \mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3 \neq 0$.

4.2.3. Representation and Computation

We represent an extensor $x \in \Lambda(F^k)$ by its coefficients in the expansion $x = \sum_{I \subseteq \{1,\ldots,k\}} x_I \mathbf{e}_I$, using 2^k elements x_I from F. The sum $z = x + y$ is given by coefficient-wise addition $z_I = x_I + y_I$, requiring 2^k additions in F. The wedge product $z = x \wedge y$ is

$$\left(\sum_{I \subseteq K} x_I \mathbf{e}_I \right) \wedge \left(\sum_{J \subseteq K} y_J \mathbf{e}_J \right) = \sum_{I,J \subseteq K} x_I y_J \cdot \mathbf{e}_I \wedge \mathbf{e}_J\,.$$

When y belongs to $\Lambda^j(F^k)$, we can restrict the summation to subsets J with $|J| = j$. Thus:

Name		$v_i \mapsto$	$e \mapsto$	Algebra	Section
ϕ	Vandermonde	$(i^0, \dots, i^{k-1})^T$	1	$\Lambda(F^k)$	4.3.2, 4.3.3
$\overline{\phi}$	Lifted Vandermonde	$\overline{\phi(v_i)}$	1	$\Lambda(F^{2k})$	4.3.5
$\overline{\beta}$	Lifted Bernoulli	$\overline{(\pm 1, \dots, \pm 1)^T}$	1	$\Lambda(F^{2k})$	4.3.6
η	Edge-variable	$\phi(v_i)$	y_e	$\Lambda(F^k)[Y]$	4.3.7
ρ	Random edge-weight	$\phi(v_i)$	Random $r \in \{1, \dots, 100k\}$	$\Lambda(F^k)$	4.4.1
λ	Labeled walks	$(x_i^{(1)}, \dots, x_i^{(k)})^T$	y_e	$\Lambda(F^k)[X, Y]$	4.4.3
$\overline{\chi}$	Color-coding	$\overline{\mathbf{e}_j}$, random $j \in \{1, \dots, k\}$	1	$Z(F^k) \subset \Lambda(F^{2k})$	4.4.4

Table 4.1.: Extensor-codings of graphs used in this chapter.

Proposition 4.2.1. *For $x \in \Lambda(F^k)$ and $y \in \Lambda^j(F^k)$, $x \wedge y$ can be computed using $2^k \binom{k}{j}$ multiplications in F.*

This is the only wedge product we need for our results, and only for $j \in \{1, 2\}$.

In particular, $\Lambda(F^k)$ is a ring with multiplication $\wedge$. Then, for a mapping $\xi \colon V(G) \to \Lambda^j(F^k)$, we can compute the walk-sum $f(G; \xi)$ from (4.2) using $O(n+m)2^k \binom{k}{j}$ field operations, which is $(n+m)2^k \operatorname{poly}(k)$ for $j = O(1)$.

For completeness, the case where $y \in \Lambda(F^k)$ is a general extensor, can be computed faster than 4^k. By realizing that the coefficient z_I is given by the *alternating subset convolution*

$$z_I = \sum_{J \subseteq I} \operatorname{sgn}(J, I \setminus J) x_J y_{I \setminus J}\,, \tag{4.6}$$

we see that $x \wedge y$ can be computed in 3^k field operations. By following Leopardi [Leo+05] and the subsequent analysis of Włodarczyk [Wło16], this bound can be improved to $O^*(2^{\omega \frac{k}{2}})$. This works by making use of an efficient embedding of a Clifford algebra related to $\Lambda(F^k)$ into a matrix algebra of dimension $2^{k/2} \times 2^{k/2}$, and expressing one product in $\Lambda(F^k)$ as k^2 products in this Clifford algebra. A detailed and streamlined account of the required arguments are given in Chap. 7. (We never need this.)

4.3. Extensor-coding

4.3.1. Walk Extensors

An *extensor-coding* is a mapping $\xi\colon V(G) \to \Lambda(F^k)$ associating an extensor with every vertex of G. If W is a walk $w_1 \dots w_\ell$ of length ℓ in G, then we define the *walk extensor* $\xi(W)$ as

$$\xi(W) = \xi(w_1) \wedge \dots \wedge \xi(w_\ell)\,.$$

Suppose now that ξ always maps to decomposable extensors. We can formulate our main insight:

Lemma 4.3.1. *If $\xi(v)$ is decomposable for all $v \in V(G)$ and W is not a path, then $\xi(W) = 0$.*

Proof. Directly follows from Property (W2). □

In particular, the (easily computed) walk-sum of ξ over the ring R with $R = \Lambda(F^k)$ is a sum over paths:

$$f(G;\xi) = \sum_{W \in \mathscr{W}} \xi(W) = \sum_{P \in \mathscr{P}} \xi(P)\,. \tag{4.7}$$

We can view ξ as the $(k \times n)$ matrix Ξ over F consisting of the columns $\xi(v_1), \dots, \xi(v_n)$. By (4.5), we have

$$\xi(w_1 \dots w_k) = d \cdot \mathbf{e}_{[k]}\,, \tag{4.8}$$

where d is the determinant of the $(k \times k)$-matrix Ξ_P of columns $\xi(w_i), \dots, \xi(w_k)$. This matrix is a square submatrix of Ξ, and vanishes if two columns are the same.

While it is terrific that non-paths vanish, we are faced with the dangerous possibility that $f(G;\xi)$ vanishes as a whole, even though $\mathscr{P}$ is not empty. There are two distinct reasons why this might happen: the extensor $\xi(P)$ might vanish for a path $P \in \mathscr{P}$, or the sum of non-vanishing extensors $\xi(P)$ vanishes due to cancellations in the linear combination.

4.3.2. Vandermonde Vectors

To address the first concern, we consider an extensor-coding ξ in *general position*, that is, such that $\xi(w_1 \dots w_k) \neq 0$ for all k-tuples of distinct vertices $w_1 \dots w_k$. Thus, ξ is in general position if and only if all square submatrices of Ξ are non-singular. Rectangular Vandermonde matrices have this property.

Lemma 4.3.2. *Let the* Vandermonde extensor-coding ϕ of G *be*

$$\phi(v_i) = (1, i^1, i^2, \dots, i^{k-1})^T \text{ for all } i \in \{1, \dots, n\}\,. \tag{4.9}$$

If $i_1, \dots, i_k \in \{1, \dots, n\}$, *then*

$$\phi(v_{i_1} \dots v_{i_k}) = \det \Phi_P \cdot \mathbf{e}_{[k]}\,,$$

where

$$\Phi_P = \begin{pmatrix} 1 & 1 & \dots & 1 \\ i_1 & i_2 & \dots & i_k \\ \vdots & \vdots & \ddots & \vdots \\ i_1^{k-1} & i_2^{k-1} & \dots & i_k^{k-1} \end{pmatrix}. \tag{4.10}$$

In particular,

$$d = \det \Phi_P = \prod_{\substack{i_a, i_b \\ a<b}} (i_a - i_b)\,. \tag{4.11}$$

4.3.3. Baseline Algorithm

Our second concern was that distinct non-vanishing paths might lead to extensors $\phi(P)$ that cancel in the sum in (4.7). Let us consider a case where this never happens by assuming that the graph G has at most one k-path. Then the sum over paths in (4.7) has at most one term and cancellations cannot occur.

This allows us to establish Thm. 4.1.2 for the special case where H is the k-path and the number C of occurrences of H in G is either zero or one.

Algorithm U (*Detect unambiguous k-path.*) *Given directed graph G and*

integer k, such that the number of k-paths in G is 0 or 1, this algorithm determines if G contains a k-path.

U1 (Set up ϕ.) Let $F = \mathbb{Q}$. Let ϕ be the Vandermonde extensor-coding as in (4.9).

U2 (Compute the walk-sum) Compute $f(G;\phi)$ as in (4.4).

U3 (Decide.) If $f(G;\phi)$ is non-zero, then return 'yes.' Otherwise, return 'no.' □

Theorem 4.3.3. *Algorithm U is a deterministic algorithm for the unambiguous k-path problem with running time $2^k(n+m)\,\mathrm{poly}(k)$.*

Proof. Consider the extensor $f(G;\phi)$ computed in Step U2. If G contains no k-path, then $f(G;\phi) = 0$ holds by (4.7). Otherwise, we have $f(G;\phi) = \phi(P)$ for the unambiguous k-path P in G. Let $P = v_{i_1} \dots v_{i_k}$. By our choice of ϕ in U1, Lemma 4.3.2 implies $f(G;\phi) = d \cdot \mathbf{e}_{[k]}$ with $d \neq 0$.

The running time of Algorithm U is clearly dominated by U2. As we discussed in Sec. 4.2.3, the value $f(G,\phi)$ can be computed with $k \cdot O(n+m)$ operations in $\Lambda(F^k)$, each of which can be done with $O(k2^k)$ operations in F. The Vandermonde extensor-coding ϕ uses only integer vectors and the absolute value of $f(G,\phi)$ is bounded by $n^{\mathrm{poly}(k)}$. In the usual word-RAM model of computation with words in $\{-n,\dots,+n\}$, we can thus store each number using $\mathrm{poly}(k)$ words. We conclude that Algorithm U has the claimed running time. □

4.3.4. Blades and Lifts

The reason that cancellations can occur in (4.7) is that the coefficients $d \in F$ in (4.8) may be negative. We will now give a general way to modify an extensor-coding in such a way that these coefficients become d^2 and thus are always positive.

Instead of $\Lambda(F^k)$, we will now work over $\Lambda(F^{2k})$. For an extensor $x = \sum_{i\in\{1,\dots,k\}} a_i \mathbf{e}_i \in F^k \subseteq \Lambda(F^k)$, we define its *lifted* version $\overline{x} \in \Lambda^2(F^{2k})$

as the blade

$$\overline{x} = \Big(\sum_{i \in \{1,\dots,k\}} a_i \mathbf{e}_i \Big) \wedge \Big(\sum_{j \in \{1,\dots,k\}} a_j \mathbf{e}_{j+k} \Big) . \tag{4.12}$$

If we let $0 \in F^k$ denote the zero vector in F^k, we can write this as

$$\overline{x} = \begin{pmatrix} x \\ 0 \end{pmatrix} \wedge \begin{pmatrix} 0 \\ x \end{pmatrix} .$$

Crucially, every $\overline{x}$ is decomposable, so Lemma 4.3.1 applies.

For an extensor-coding $\xi \colon V(G) \to F^k$, we define the lifted extensor-coding $\overline{\xi} \colon V(G) \to \Lambda(F^{2k})$ by setting $\overline{\xi}(v) = \overline{\xi(v)}$ for all $v \in V(G)$. For a path $P \in \mathscr{P}$, with $P = w_1 \cdots w_k$, the correspondence between $\xi(P)$ and $\overline{\xi}(P)$ is as follows. Consider the $k \times k$ matrix Ξ_P of extensors given by

$$\Xi_P = \Big(\xi(w_1) \dots \xi(w_k) \Big) .$$

From Property (W3), we get

$$\xi(P) = (\det \Xi_P) \mathbf{e}_{[k]} ,$$

and

$$\overline{\xi}(P) = \det \begin{pmatrix} \xi(w_1) & 0 & \dots & \xi(w_k) & 0 \\ 0 & \xi(w_1) & \dots & 0 & \xi(w_k) \end{pmatrix} \mathbf{e}_{[2k]} .$$

Using basic properties of the determinant, we can rewrite the coefficient of $\mathbf{e}_{[2k]}$ to

$$(-1)^{\binom{k}{2}} \det \begin{pmatrix} \xi(w_1) & \dots & \xi(w_k) & 0 & \dots & 0 \\ 0 & \dots & 0 & \xi(w_1) & \dots & \xi(w_k) \end{pmatrix} = (-1)^{\binom{k}{2}} (\det \Xi_P) \cdot (\det \Xi_P) = (-1)^{\binom{k}{2}} (\det \Xi_P)^2 .$$

Thus, we have

$$\overline{\xi}(P) = \pm (\det \Xi_P)^2 \mathbf{e}_{[2k]} ,$$

where the sign depends only on k.

We evaluate the walk-sum over $\Lambda(F^{2k})$ at $\overline{\xi}$ to obtain:

$$f(G;\overline{\xi}) = \pm \sum_{P \in \mathscr{P}} (\det \Xi_P)^2 \cdot \mathbf{e}_{[2k]} \,. \tag{4.13}$$

4.3.5. Deterministic Algorithm for Path Detection

As an application of the lifted extensor-coding, let $\phi\colon V(G) \to F^k$ be the Vandermonde extensor-coding from Lemma 4.3.2. We imitate Algorithm U to arrive at a deterministic algorithm for k-path. Our algorithm slightly improves upon the time bound of $4^{k+o(k)} \cdot \mathrm{poly}(n)$ of Chen *et al.* [Che+07; Che+09], but does not come close to the record bound $2.5961^k \cdot \mathrm{poly}(n)$ of Zehavi [Zeh15].

Theorem 4.3.4 (Superseded by [Zeh15])**.** *There is a deterministic algorithm that, given a directed graph G, checks if G has a path of length k in time $4^k(n+m)\,\mathrm{poly}(k)$.*

Proof. The algorithm is just Algorithm U, except that we evaluate the walk-sum over $\Lambda(F^{2k})$ and at $\overline{\phi}$. The correctness of this algorithm follows from (4.13). Each addition $y + z$ in $\Lambda(F^{2k})$ can be carried out using $O(2^{2k})$ addition operations in F, and each multiplication $y \wedge \overline{x}$ with elements of the form $\overline{x}$ for $x \in F^k$ takes at most $O(2^{2k}k^2)$ operations in F, as discussed in Sec. 4.2.3. Overall, this leads to the claimed running time. □

4.3.6. Bernoulli Vectors

We present our algorithm for approximate counting. Now instead of the Vandermonde extensor-coding as in Lemma 4.3.2, we sample an extensor-coding $\beta\colon V(G) \to \{-1, 1\}^k$ uniformly at random.

The approximate counting algorithm is based on the following observation: If B_P is the $k \times k$ matrix corresponding to $\beta(w_1), \ldots, \beta(w_k)$, then all matrices B_P are sampled from the same distribution. Thus, the random variables $\det B_P^2$ have the same mean $\mu > 0$. The expectation of the sum of determinant squares is $\mu \cdot |\mathscr{P}|$, from which we can recover

an estimate for the number of paths. Our technical challenge is to bound the variance of the random variable $\det B_P^2$.

Algorithm C (*Randomized counting of k-path.*) *Given directed graph G and integers k and t, approximately counts the number of k-paths using t trials.*

C1 (Initialize.) Set $j = 1$.

C2 (Set up jth trial.) For each $i \in \{1, \ldots, n\}$, let $\beta(v_i)$ be a column vector of k values chosen from ± 1 independently and uniformly at random.

C3 (Compute scaled approximate mean X_j.) Compute X_j with

$$f(G; \overline{\beta}) = X_j \cdot \mathbf{e}_{[2k]}.$$

C4 (Repeat t times.) If $j < t$ then increment j and go to C2.

C5 (Return normalized average.) Return $(X_1 + \cdots + X_t)/(k!t)$

We are ready for the special case of Theorem 4.1.1, approximating $\mathrm{Sub}(H, G)$ when H is the k-path. In this case, $\mathrm{Sub}(H, G) = |\mathscr{P}|$.

Theorem 4.3.5. *For any $\varepsilon > 0$, Algorithm C produces in time $(4^k/\varepsilon^2) \cdot (n + m) \cdot \mathrm{poly}(k)$ a value X such that with probability at least 99%, we have*

$$(1 - \varepsilon) \cdot |\mathscr{P}| \leq X \leq (1 + \varepsilon) \cdot |\mathscr{P}| .$$

A matrix whose entries are i.i.d. random variables taking the values $+1$ and -1 with equal probability $\frac{1}{2}$ is called *Bernoulli*. We need a result from the literature about the higher moments of the determinant of such a matrix.

Theorem 4.3.6 ([NRR54])**.** *Let B be a $k \times k$ Bernoulli matrix. Then,*

$$\mathbb{E} \det B^2 = k! \tag{4.14}$$

$$\mathbb{E} \det B^4 \leq (k!)^2 \cdot k^3 . \tag{4.15}$$

For completeness, we include a careful proof for a slightly different distribution, later on.

Proof of Theorem 4.3.5. Run algorithm C with $t = 100k^3/\varepsilon^2$. Set $\mu = |\mathscr{P}|$. Recall from (4.13) that X_j can be written as

$$X_j = \pm(\det B_1^2 + \det B_2^2 + \cdots + \det B_\mu^2)\,, \tag{4.16}$$

where for $i \in \{1, \ldots, \mu\}$, each B_i is a submatrix of of the $k \times n$ matrix with columns $\beta(v_1)$, $\beta(v_2)$,$\cdots$, $\beta(v_n)$. The sign can be easily computed and only depends on k; we assume without loss of generality that it is $+1$. By our choice of β in Step C2, each B_i is therefore a Bernoulli matrix, but they are not independent.

By Theorem 4.3.6, we have $\mathbb{E} \det B_i^2 = k!$ for each $i \in \{1, \ldots, \mu\}$, so by linearity of expectation,

$$\mathbb{E}X_j = \mu k!\,.$$

We turn to $\operatorname{Var} X_j$, which requires a bit more attention. For all $i, \ell \in \{1, \ldots, \mu\}$, the matrices B_i and B_ℓ follow the same distribution, so $\operatorname{Var} \det B_i^2 = \operatorname{Var} \det B_\ell^2$. Thus, using Cauchy–Schwartz, we have

$$\operatorname{Cov}(\det B_i^2, \det B_\ell^2) = \sqrt{(\operatorname{Var} \det B_i^2) \cdot (\operatorname{Var} \det B_\ell^2)} = \sqrt{(\operatorname{Var} \det B_i^2)^2} = \operatorname{Var} \det B_i^2 \le \mathbb{E} \det B_i^4 \le (k!)^2 k^3\,,$$

where the last two inequalities uses $\operatorname{Var} Y \le \mathbb{E}Y^2$ with $Y = \det B_i^2$ and (4.15) in Theorem 4.3.6 with $B = B_i$. We obtain

$$\operatorname{Var} X_j = \operatorname{Cov}(X_j, X_j) = \operatorname{Cov}\left(\sum_{i=1}^{\mu} \det B_i^2, \sum_{\ell=1}^{\mu} \det B_\ell^2\right) = \sum_{i,\ell=1}^{\mu} \operatorname{Cov}(\det B_i^2, \det B_\ell^2) \le \mu^2 \cdot (k!)^2 \cdot k^3\,.$$

Now consider the value X returned by the algorithm in Step C5 and observe $X = (X_1 + \ldots + X_t)/(k!t)$. By linearity of expectation, we have $\mathbb{E}X = t\mu k!/(k!t) = \mu$. Recalling that $\operatorname{Var}(a \cdot X) = a^2 \cdot \operatorname{Var}(X)$ for a

random variable X and a scalar a, by independence of the X_j, we have

$$\operatorname{Var} X = \operatorname{Var}\left(\frac{1}{k!t}\sum_{j=1}^{t} X_j\right) = \frac{1}{(k!t)^2}\sum_{j=1}^{t}\operatorname{Var} X_j \leq \frac{1}{(k!t)^2} t\mu^2 (k!)^2 k^3 = \frac{\mu^2 k^3}{t}\,.$$

Now Chebyshev's inequality gives

$$\Pr(|X-\mu| \geq \varepsilon\mu) \leq \frac{\operatorname{Var} X}{\varepsilon^2\mu^2} \leq \frac{\mu^2 k^3}{\varepsilon^2\mu^2 t} = \frac{1}{100}\,,$$

which implies the stated bound.

The claim on the running time follows from the discussion in Sec. 4.2.3 and the representation of the input as adjacency lists. □

4.3.7. Edge-Variables

We extend Algorithm U from the unambiguous case to the case where the number of k-paths is bounded by some integer C. The construction uses a coding with formal variables on the edges. To this end, enumerate E as $\{e_1, \ldots, e_m\}$ and introduce the set Y of formal variables $\{y_1, \ldots, y_m\}$. Our coding maps e_j to y_j.

We then use the following theorem about deterministic polynomial identity testing of sparse polynomials due to Bläser *et al.*:

Theorem 4.3.7 (Theorem 2 in [Blä+09])**.** *Let f be an m-variate polynomial of degree k consisting of C distinct monomials with integer coefficients, with the largest appearing coefficient bounded in absolute value by H. There is a deterministic algorithm which, given an arithmetic circuit of size s representing f, decides whether f is identically zero in time $O((mC\log k)^2 s \log H)$*

To use this result, we need to interpret the walk-sum as a small circuit in the variables Y with integer coefficients. This requires 'hard-wiring' every skew product in the exterior algebra by the corresponding small circuit over the integers. Algorithm F contains a detailed description.

Algorithm F (*Detect few k-paths*) *Given directed graph G and integer k, such that the number of k-paths in G is at most C, this algorithm determines if G contains a k-path.*

F1 [Set up η.] Let $F = \mathbb{Z}$ and define $\eta\colon V(G) \cup E(G) \to \Lambda(F^k)[Y]$ by $\eta(v) = \phi(v)$ and $\eta(e_j) = y_j$.

F2 [Circuit K over $\Lambda(F^k)[Y]$.] Let K be the skew arithmetic circuit from (4.3) for computing $f(G;\eta)$ from its input gates labeled by $\eta(v)$ for $v \in V(G)$ and $\eta(e)$ for $e \in E(G)$.

F3 [Circuit L over $\mathbb{Z}[Y]$.] Create a circuit L with inputs from $\mathbb{Z}$ and Y as follows. Every gate g in K corresponds to 2^k gates g_I for $I \subseteq \{1,\dots,k\}$ such that $g = \sum_I g_I \cdot \mathbf{e}_I$. When g is an input gate of the form $g = \phi(v_i)$ the only nonzero gates in L are $g_{\{j\}} = i^j$, an integer. When g is an input gate of the form $g = y_j$ then the only nonzero gate is the variable $g_\emptyset = y_j$. If $g = g' + g''$ then g_I is the addition gate computing $g'_I + g''_I$. If g is the skew product $g' \cdot g''$, where g'' is an input gate, then g_I is the output gate of a small subcircuit that computes

$$\sum_{\substack{J \subseteq I \\ |J| \leq 1}} \operatorname{sgn}(I \setminus J, J) g'_{I \setminus J} g''_J \,.$$

(This is (4.6), noting $g''_J = 0$ for $|J| > 1$.) If g is the output gate of K then $g_{\{1,\dots,k\}}$ is the output gate of L.

F4 [Decide.] Use the algorithm from the above theorem to determine if L computes the zero polynomial. Return that answer.

We are ready to establish Theorem 4.1.2 for the case where the pattern graph H is a path.

Theorem 4.3.8. *Algorithm F is a deterministic algorithm for the k-path problem when there are at most $C \in \mathbb{N}$ of them, and runs in time $C^2 2^k n^{O(1)}$.*

Proof. Let G be a graph with at most C paths of length k. First, we argue for correctness of Algorithm F. From (4.2), it follows that the

circuit K outputs

$$f(G;\eta) = \sum_{P \in \mathscr{P}} \left(\prod_{e_i \in P} y_i \right) \cdot \det(\Phi_P) \cdot \mathbf{e}_{[k]} \in \Lambda(F^k)[Y]\,,$$

where Φ_P is the Vandermonde matrix associated with the vertices on P from (4.10). By the construction of L, the output gate of L computes the polynomial

$$\sum_{P \in \mathscr{P}} \left(\prod_{e_i \in P} y_i \right) \cdot \det(\Phi_P) \in F[Y]\,,$$

which is just an m-variate, multilinear polynomial over the integers. Note that, by construction, all the appearing determinants are non-zero. Since all our graphs are directed, any path is already uniquely determined by the unordered set of edges that appear on it. It follows that the monomials belonging to the distinct k-paths in a graph, each formed as the product of the edge variables corresponding to the edges on the path, are linearly independent. Therefore, the monomials of the polynomial in Y computed by L are in bijective correspondence with the k-paths in G. Theorem 4.3.7 thus yields the correct answer.

As for the running time, we see that every gate in K is replaced by at most $2^k(k+1)$ new gates to produce L. Since K was of size $O(k(n+m))$, the resulting circuit L is of size $O(2^k(n+m)\,\mathrm{poly}(k))$ and can be constructed in this time. Since, as noted, the monomials in the polynomial computed by L are in bijection with the k-paths in G, there are at most C many. The application of Theorem 4.3.7 is thus within the claimed running time bound. □

4.4. Connection to Previous Work

In this section, we show how our approach using exterior algebras specializes to the group algebra approach of Koutis [Kou08] (cf. Sect. 3.4) when the ground field has characteristic two. We also argue that the combinatorial approach of Björklund *et al.* [Bjö+17a] (cf. Sect. 3.3)

using *labeled walks* can be seen as an evaluation over an exterior algebra. Moreover, we show how Color-Coding [AYZ95] (cf. Sect. 3.1) arises as a special case, and present the recent approach of representative paths due to Fomin *et al.* [Fom+16] (cf. Sect. 3.2) in the language of exterior algebra.

4.4.1. Random Edge-Weights

We begin with a randomized algorithm for detecting a k-path in a directed graph, recovering Koutis's and Williams's result (cf. Theorem 3.4.10).

Theorem 4.4.1 ([Kou08; Wil09])**.** *There is a randomized algorithm for the k-path problem with running time $2^k(n+m)\,\mathrm{poly}(k)$.*

Proof. The algorithm is the baseline Algorithm U, but with the following step replacing U1:

U1′ Enumerate the edges as $E = \{e_1, \ldots, e_m\}$ and choose m integers $r_1, \ldots, r_m \in \{1, \ldots, 100k\}$ uniformly at random. Define the extensor-coding ρ on $V(G) \cup E(G)$ by

$$v_i \mapsto \phi(v_i), \qquad e_j \mapsto r_j\,.$$

The rest is the same, with ρ instead of ϕ.

The correctness argument is a routine application of polynomial identity testing: The expression $f(G;\rho)$ can be understood as the result of the following random process. Introduce a formal 'edge' variable y_e for each $e \in E$ and consider the expression

$$\sum_{w_1 \cdots w_k \in \mathscr{P}} y_{w_1 w_2} \cdots y_{w_{k-1} w_k} \cdot \phi(w_1 \ldots w_k) \tag{4.17}$$

as a polynomial of degree k in the variables $y_{e_1}, \ldots, y_{e_m}$. In a directed graph, every path is uniquely determined by its set of (directed) edges. Thus, if $\mathscr{P} \neq \emptyset$ then (4.17) is a nonzero polynomial. The walk-sum $f(G;\rho)$ is an evaluation of this polynomial at a random point $y_{e_1} =$

$r_1, \ldots, y_{e_m} = r_m$. By the DeMillo–Lipton–Schwartz–Zippel Lemma 3.3.3, $f(G;\rho)$ is nonzero with probability $\frac{1}{100}$. □

4.4.2. Group Algebras

Let us very quickly recall that group algebras over a group G are the formal linear combinations of group elements, thus having the group as a basis, and having the algebra multiplication defined as the bilinear extension of the group multiplication. For details, see Sect. 3.4.

If F has characteristic two, we can relate the exterior algebra to the group algebra $F[\mathbb{Z}_2^k]$ as follows.

Proposition 4.4.2. *Let F be of characteristic two and F^k the free vector space of dimension k with basis $\{\mathbf{e}_1, \ldots, \mathbf{e}_k\}$. Then, the group algebra $F[\mathbb{Z}_2^k]$ is isomorphic to $\Lambda(F^k)$.*

Proof. We denote with $\mathbf{e}_i \in \mathbb{Z}_2^k$ for $i \in \{1, \ldots, k\}$ the ith unit vector. The morphism induced by mapping $\Lambda(F^k) \ni \mathbf{e}_i \mapsto (1 + \mathbf{e}_i) \in F[\mathbb{Z}_2^k]$ is an isomorphism. □

Remark 4.4.3. The previous proposition shows that over fields of characteristic two, our exterior algebras specialize exactly to the group algebras used by Koutis and Williams [Kou08; Wil09], and therefore, the approach of using random edge-weights in the coding ρ from Sect. 4.4.1 specializes to Williams' algorithm [Wil09] over fields of characteristic two and sufficient size, albeit with deterministically chosen vectors at the vertices, which of course also could be done randomly without changing anything about the result.

Exterior Algebras as Quotients of Monoid Algebras

We have seen that the above group algebras are exterior algebras in characteristic two, and now consider the other direction. For $k \in \mathbb{N}$, consider the free monoid E^* over the generators $E := \{\mathbf{e}_1, \ldots, \mathbf{e}_k, \mu, \theta\}$, and impose these relations on E^*: The element θ is a zero, *i.e.*, $\theta x = x\theta = \theta$ for all $x \in E^*$, and μ central, *i.e.*, $\mu x = x\mu$ for all $x \in E^*$, and we shall have for all i that $\mathbf{e}_i^2 = \theta$. We further demand that

$\mathbf{e}_i\mathbf{e}_j = \mu\mathbf{e}_j\mathbf{e}_i$ and $\mu^2 = 1_E$ hold. Let S be the quotient of E^* by these relations, and consider $F[S]$. Let I_S be the ideal generated by $\{\theta, \mu+1\}$. Naturally in $F[S]/I_S$, we have $\theta = 0$ and $\mu = -1$, and hence $\mathbf{e}_i^2 = 0$ and $\mathbf{e}_i\mathbf{e}_j = -\mathbf{e}_j\mathbf{e}_i$. Thus, $F[S]/I_S$ is *precisely* the exterior algebra over F^k. Hence, not only are the above-mentioned group algebras a special case of an exterior algebra, but any exterior algebra arises as a quotient of some monoid algebra. Note that this representation of exterior algebras (and, more generally, Clifford algebras) as quotients of certain monoid (or group) algebras is folklore.

4.4.3. Labeled Walks

Consider now λ, the extensor-coding for labeled walks. That is, let y_e be variables associated with each directed edge $e \in E(G)$, and let a vector of variables $(x_i^{(1)}, \ldots, x_i^{(k)})^T$ be associated with each vertex $v_i \in V(G)$. The superscript index is referred to as the *label* of a vertex in a walk. Recall from Sect. 3.3 the following polynomial in the y_e and $x_i^{(j)}$:

$$P(x,y) = \sum_{w_1\cdots w_k \in \mathscr{W}} \sum_{\ell\in\mathfrak{S}_k} \prod_{i=1}^{k-1} y_{w_i w_{i+1}} \prod_{i=1}^{k} x_i^{\ell(i)}\,, \tag{4.18}$$

and the fact that over characteristic 2, the sum can be restricted to paths instead of walks:

$$P(x,y) = \sum_{w_1\cdots w_k \in \mathscr{P}} \sum_{\ell\in\mathfrak{S}_k} \prod_{i=1}^{k-1} y_{w_i w_{i+1}} \prod_{i=1}^{k} x_i^{\ell(i)}. \tag{4.19}$$

We may now observe that the inner sum is just a determinant of a suitably chosen matrix, namely the $k \times k$ matrix $X(w_1\cdots w_k) := (x_{w_j}^{(i)})_{i,j}$ indexed by pairs of numbers and vertices, and we can write

$$P(x,y) = \sum_{w_1\cdots w_k \in \mathscr{P}} \prod_{i=1}^{k} y_{w_i w_{i+1}} \det(X(w_1\cdots w_k))\,.$$

Here, the Matrix $X(P)$ for a path $P \in \mathscr{P}$ plays precisely the rôle that the matrix Φ_P played in the proof of Theorem 4.3.8, just that this time, it carries variable entries.

It is now easy to see that this is once again just the evaluation of the circuit computing the k-walk extensor over characteristic two by the property of the wedge product expressed in Equation (4.5). In short, the walk-sum $f(G;\lambda)$ for the extensor-coding λ achieves

$$f(G;\lambda) = P(x,y)$$

whenever F is of characteristic two.

This gives the connection between λ and ρ from Sect. 4.4.1 over characteristic two, and by the remark in Sect. 4.4.2, also the connection between the group-algebra approach, identifying the three techniques as one.

4.4.4. Color-coding

Let us see how the Color-Coding-technique by Alon, Yuster, and Zwick [AYZ95] can be seen as a special case of extensor-coding.

Consider a coding with with basis vectors of F^k taken uniformly and independently,

$$\chi(v) \in \{\mathbf{e}_1, \ldots, \mathbf{e}_k\}\,.$$

We may think of the basis vectors as k colors. Note that if $P = w_1 \cdots w_k$ is a path then its walk extensor $\chi(P)$ vanishes exactly if the $k \times k$-matrix whose columns are the random unit vectors $\chi(w_1) \cdots \chi(w_k)$ is singular. Thus,

$$\Pr(\chi(P) = 0) = \frac{k!}{k^k} \leq e^{-k}\,.$$

We lift χ to $\overline{\chi}\colon V(G) \to \Lambda^2(F^{2k})$, ensuring $\overline{\chi}(P) = \{0 \cdot \mathbf{e}_{[2k]}, 1 \cdot \mathbf{e}_{[2k]}\}$, to avoid cancellation.

Let us write $Z(F^k)$ for the subalgebra of $\Lambda(F^{2k})$ generated by $\{\overline{\mathbf{e}_1}, \ldots, \overline{\mathbf{e}_k}\}$, called the *Zeon-algebra*. It already made an appearance in graph algorithms in the work of Schott and Staples [SS11].

Lemma 4.4.4. *$Z(F^k)$ is commutative and of dimension 2^k, and its generators $\overline{\mathbf{e}_i}$ square to zero. Furthermore, addition and multiplication can be performed in $2^k \cdot \mathrm{poly}(n)$ field operations.*

Proof. Directly from the definition of the exterior algebra, $\overline{\mathbf{e}_i} \wedge \overline{\mathbf{e}_i} = 0$. Furthermore,

$$\begin{aligned}\overline{\mathbf{e}_i} \wedge \overline{\mathbf{e}_j} = \mathbf{e}_i \wedge \mathbf{e}_{i+k} \wedge \mathbf{e}_j \wedge \mathbf{e}_{j+k} = -\mathbf{e}_i \wedge \mathbf{e}_j \wedge \mathbf{e}_{i+k} \wedge \mathbf{e}_{j+k} = \\ \mathbf{e}_j \wedge \mathbf{e}_i \wedge \mathbf{e}_{i+k} \wedge \mathbf{e}_{j+k} = -\mathbf{e}_j \wedge \mathbf{e}_i \wedge \mathbf{e}_{j+k} \wedge \mathbf{e}_{i+k} = \\ \mathbf{e}_j \wedge \mathbf{e}_{j+k} \wedge \mathbf{e}_i \wedge \mathbf{e}_{i+k} = \overline{\mathbf{e}_j} \wedge \overline{\mathbf{e}_i}\,,\end{aligned}$$

and therefore $Z(F^k)$ is commutative. It is readily verified that the elements $\overline{\mathbf{e}_I}$ with $I \subseteq [k]$ form a basis of $Z(F^k)$. By renaming $\overline{\mathbf{e}_i}$ as, say, X_i, we recognize $Z(F^k)$ as the F-algebra of multilinear polynomials in variables $X_i, 1 \leq i \leq k$ with the relations $X_i^2 = 0$ for all $1 \leq i \leq k$. Addition is performed component-wise and can be done trivially in the required bound. By standard methods, such as Kronecker substitution and Schönhage–Strassen-multiplication (see, *e.g.*, [GG13]), or more directly, fast subset convolution [Bjö+07], multiplication of multilinear polynomials modulo X_i^2 can be performed in $Z(F^k)$ in the required time bound. □

Thus, we can evaluate the walk-sum $f(G; \overline{\chi})$ in time $2^k(n+m)\,\mathrm{poly}(k)$. Repeating the algorithm e^k times we arrive at time $(2\mathrm{e})^k(n+m)\,\mathrm{poly}(k)$, as in [AYZ95].

We apply these constructions in a similar setting in Sect. 4.7.

4.4.5. Representative Paths

The idea to represent the $\Omega(n^k k!)$ many k-paths in G by a family of only $f(k)\,\mathrm{poly}(n)$ many combinatorial objects goes back to the original k-path algorithm of Monien [Mon85]. Recent representative-sets algorithms [Fom+16; Zeh15], including the fastest deterministic k-path algorithms, follow this approach, maintaining representative families of subsets of a linear matroid.

One of those constructions is inspired by Lovász's proof of the Two-Families theorem, which is originally expressed in exterior algebra [Lov77]. In fact, as pointed out by Marx [Mar09a, Proof of Lemma 4.2], the column vectors in the matroid representation are exactly Vandermonde extensors. Fomin *et al.* [Fom+16, Theorem 1] develop this idea in detail for k-path, but their presentation abandons the exterior algebra and continues in the framework of uniform matroids.

For completeness, let us now state the deferred proof of Proposition 3.2.4, which we restate for convenience:

Proposition 3.2.4 (restated)**.** Let $\mathcal{F} \subseteq \binom{U}{p}$, *i.e.*, a set family containing only sets of size p. Then $\mathcal{F}$ has a q-representant of size $\binom{p+q}{p}$.

Proof of Proposition 3.2.4. Let U be of size n, enumerated as $U = \{u_1, \ldots, u_n\}$, and let $\mathcal{F} \subseteq \binom{U}{p}$ be a family of p-sets. Let $k = p + q$. For $u_i \in U$, let $v(u_i) = (1, i, i^2, \ldots, i^{k-1})$, and associate with $A \subseteq U$ the wedge product $w(A) = \bigwedge_{u \in A} v(u)$. Let $S = \{w(F) \mid F \in \mathcal{F}\}$, and let $\mathcal{B} \subseteq S$ be a linear basis, composed only of elements of S, of the subspace in $\Lambda^p(\mathbb{R}^k)$ spanned by S.

Since $\Lambda^p(\mathbb{R}^k)$ has dimension $\binom{k}{p}$, $\mathcal{B}$ is at most of size d, with $d \leq \binom{k}{p}$. Denote with $\mathcal{F}'$ the subset of $\mathcal{F}$, such that $\mathcal{B}$ is comprised of the corresponding wedge products, *i.e.*, $\mathcal{B} = \{w(F) \mid F \in \mathcal{F}'\}$. We claim that $\mathcal{F}'$ q-represents $\mathcal{F}$.

To see this, let $A \in \binom{U}{q}$ be disjoint to some $B \in \mathcal{F}$. The set $\{v(u)\}_{u \in A \cup B}$ is then linearly independent, and consequently, $w(A) \wedge w(B) = \pm w(A \cup B) \neq 0$. Enumerate $\mathcal{F}'$ as $\mathcal{F}' = \{B_1, \ldots, B_d\}$. Since

$$\mathcal{B} = \{w(B_1), \ldots, w(B_d)\}$$

is a basis of the space spanned by S, there are $\lambda_1, \ldots, \lambda_d$ such that $w(B) = \sum_{i=1}^{d} \lambda_i w(B_i)$, and since $w(A) \wedge w(B) \neq 0$, at least for one i, $w(A) \wedge w(B_i) \neq 0$, and therefore A and B_i are disjoint, as desired. □

Let us go further than just using this connection to the exterior algebra once, and then abandoning it alltogether and returning to the combinatorial point of view: We give a relatively short and complete

presentation of the representative set approach, entirely formulated over the exterior algebra and reinterpreting the combinatorial perspective that was provided in Sect. 3.2 purely algebraically. This has only expository value; the time bounds in this construction are not competitive.

For a set $\mathscr{R}$ of walks and an extensor coding ξ to $\Lambda(F^k)$ we define the *extensor span* $\langle\mathscr{R}\rangle$ via

$$\langle\mathscr{R}\rangle = \operatorname{span}\Big(\{\,\xi(R) \,:\, R \in \mathscr{R}\,\}\Big)\,,$$

that is, $\langle\mathscr{R}\rangle$ is the set of linear combinations over F of extensors viewed as 2^k-dimensional vectors. A set $\mathscr{R}$ of walks *represents* another set $\mathscr{P}$ of walks if $\xi(P) \in \langle\mathscr{R}\rangle$ for all $P \in \mathscr{P}$. For $p \in \{1,\ldots,k\}$, we write $\mathscr{P}_v^p$ for the set of length-p paths of G that end in v. We will use the Vandermonde coding ϕ for ξ.

Algorithm R (*Detect k-paths using representative paths.*) *Given directed graph G and integer k, this algorithm determines if G contains a k-path. For each $p \in \{1,\ldots,k\}$ and $v \in V(G)$, the algorithm computes a set $\mathscr{R}_v^p$ of paths such that*

$$\phi(P) \in \langle\mathscr{R}_v^p\rangle \qquad \textit{for each } P \in \mathscr{P}_v^p \tag{4.20}$$

and

$$|\mathscr{R}_v^p| \le 2^k\,. \tag{4.21}$$

R1 (First round.) Let $p = 1$. For each $v \in V(G)$, set $\mathscr{R}_v^1 = \{v\}$, the singleton set of 1-paths.

R2 (Construct many representative walks.) For each $v \in V(G)$, set

$$\mathscr{Q}_v^{p+1} = \big\{\, Rv \,:\, R \in \mathscr{R}_u^p \text{ and } uv \in E(G)\,\big\}\,. \tag{4.22}$$

R3 (Remove redundant walks.) For each $v \in V(G)$, set $\mathscr{R}_v^{p+1} = \emptyset$. For each $Q \in \mathscr{Q}_v^{p+1}$ in arbitrary order, if $\phi(Q) \notin \langle\mathscr{R}_v^{p+1}\rangle$, then add Q to $\mathscr{R}_v^{p+1}$. [Now $\phi(Q) \in \langle\mathscr{R}_v^{p+1}\rangle$.]

R4 (Done?) If $p+1 < k$ then increment p and go to R3. Otherwise return 'true' if and only if $\mathscr{R}_v^k \neq \emptyset$ holds for some $v \in V(G)$.

Proposition 4.4.5 (Theorem 1 in [Fom+16])**.** *Algorithm R is a deterministic algorithm for k-path with running time* $\exp(O(k))\operatorname{poly}(n)$.

Proof. We need to convince ourselves that the invariants (4.20) and (4.21) hold, and that the constructed sets $\mathscr{R}_v^p$ only contain paths from $\mathscr{P}_v^p$. For the size invariant (4.21), it suffices to observe that $\langle \mathscr{R}_v^{p+1} \rangle$ is a subspace of $\Lambda(F^k)$ and thus has dimension at most 2^k. Each element Q was added in Step R3 only if it increased the dimension of $\langle \mathscr{R}_v^{p+1} \rangle$, which can happen at most 2^k times.

We prove (4.20) by induction on p. For $p = 1$, we have $\mathscr{P}_v^1 = \mathscr{R}_v^1$ for all $v \in V(G)$ by Step R1. For the induction step, assume that p satisfies $\phi(\mathscr{P}_u^p) \subseteq \langle \mathscr{R}_u^p \rangle$ for all $u \in V(G)$. Let $v \in V(G)$ and consider a path Puv from $\mathscr{P}_v^{p+1}$. To establish the inductive claim, it remains to show that $\phi(Puv) \in \langle \mathscr{R}_v^p \rangle$ holds. Note that Pu belongs to $\mathscr{P}_u^p$, so the induction hypothesis implies $\phi(Pu) \in \langle \mathscr{R}_u^p \rangle$. Thus, there are coefficients $a_1, \ldots, a_d \in F$ and paths $R_1, \ldots, R_d \in \mathscr{R}_u^p$ such that

$$\phi(Puv) = \phi(Pu) \wedge \phi(v) = \left(\sum_{j=1}^{d} a_i \phi(R_i) \right) \wedge \phi(v) = \sum_{j=1}^{d} a_i \phi(R_i) \wedge \phi(v) = \sum_{j=1}^{d} a_i \phi(R_i v) \,. \quad (4.23)$$

From $R_i \in \mathscr{R}_u^p$ and $uv \in E(G)$ we obtain $R_i v \in \mathscr{Q}_v^{p+1}$ by construction (4.22). If $R_i v$ is not a path, then $\phi(R_i v) = 0 \in \langle \emptyset \rangle$ holds, which implies that $R_i v$ is not added to $\mathscr{R}_v^{p+1}$ in Step R3. Thus Step R3 only ever adds paths, which implies $\mathscr{R}_v^{p+1} \subseteq \mathscr{P}_v^{p+1}$ as required. Even if $R_i v$ is path, we may not have $R_i v \in \mathscr{R}_v^{p+1}$. Nevertheless Step R3 ensures that $\phi(R_i v) \in \langle \mathscr{R}_v^{p+1} \rangle$ holds at the end of the construction. Together with expression (4.23), this shows that $\phi(Puv)$ belongs to $\langle \mathscr{R}_v^{p+1} \rangle$, so that the representation invariant (4.20) holds.

It remains to prove the correctness of the algorithm. If the algorithm outputs true, then $\emptyset \neq \mathscr{R}_v^k \subseteq \mathscr{P}_v^k$ holds, and so there exists a k-path.

On the other hand, if there exists some k-path, say $P \in \mathscr{P}_v^k$, then $\phi(P) \neq 0$ follows from Lemma 4.3.1 and the fact that the extensors $\phi(v_i)$ are in general linear position. We have $\phi(P) \in \langle \mathscr{R}_v^k \rangle$ by (4.20), which implies that $\dim\langle \mathscr{R}_v^k \rangle \neq 0$ and $\mathscr{R}_v^k \neq \emptyset$ holds. Thus the algorithm correctly outputs 'true'.

For the running time, computation of $\mathscr{R}_v^p$ requires linear algebra on $2^k \times 2^k n$ matrices over F. This can be done in time $\exp(O(k))\,\mathrm{poly}(n)$. □

A more careful analysis of the linear and exterior algebra operations yields an upper bound of $2^{\omega k}\,\mathrm{poly}(n) \leq 5.19^k\,\mathrm{poly}(n)$ on the running time of algorithm R.

4.5. Random Determinants

Expressions for the higher moments of determinants of random matrices are available in the literature since the 1950s, see [NRR54]. Such results are considered routine, and follow from exercise 5.64 in Stanley [Sta99] or the general method laid out on page 45–46 in Girko's book [Gir90], but we have found no presentation that is quite complete. For a judicious choice of distribution, the arguments become quite manageable, so we include a complete derivation.

Let B denote a random $k \times k$ matrix constructed by choosing every entry independently and at random from the set $\{\pm\sqrt{3}, 0\}$ with the following probabilities:

$$\Pr(b_{ij} = -\sqrt{3}) = \Pr(b_{ij} = \sqrt{3}) = \tfrac{1}{6}, \qquad \Pr(b_{ij} = 0) = \tfrac{2}{3}\,.$$

It is clear that every matrix entry satisfies $\mathbb{E}b_{ij} = 0$ and $\mathbb{E}b_{ij}^2 = \frac{1}{3}3 = 1$ and $\mathbb{E}b_{ij}^4 = \frac{1}{3}9 = 3$.

We will investigate the second and fourth moments of $\det B$. By multiplicativity of the determinant, we can write $(\det B)^r = \det B^r$.

To see

$$\mathbb{E}\det B^2 = k! \tag{4.24}$$

expand $\det B$ by the first row. If we write B_{ij} for B with the ith row and jth column deleted, we have

$$\mathbb{E}\det B^2 = \mathbb{E}\sum_{i,j}(-1)^{i+j}b_{1i}b_{1j}\det B_{1i}\det B_{1j}\,.$$

The sum extends over all choices of $i, j \in \{1,\dots,k\}$, but the only nonzero contributions are from $i = j$. This is because for $i \neq j$, the factor $b_{1j}\det B_{1i}\det B_{1j}$ depends only on variables that are independent of b_{1i}, and the latter vanishes in expectation. Thus,

$$\mathbb{E}\det B^2 = \sum_i(-1)^{2i}\mathbb{E}b_{1i}^2\mathbb{E}\det B_{1i}^2 = k\mathbb{E}\det B_{11}^2\,,$$

because the distributions of $\det B_{1i}$ for $i \in \{1,\dots,k\}$ are the same. This can be viewed as a recurrence relation for $\mathbb{E}\det B^2$ as a function of the dimension k, which solves to (4.24).

To show

$$\mathbb{E}\det B^4 = \tfrac{1}{2}(k!)(k+1)(k+2),$$

we use the same kind of arguments. Write f_k for $\mathbb{E}\det B^4$. We have $f_1 = \mathbb{E}b_{11}^4 = 3$ and can compute $f_2 = 12$. For larger k, we expand the first row of B to obtain

$$f_k = \mathbb{E}\det B^4 = \mathbb{E}\sum_{i,j,l,m}(-1)^{i+j+l+m}b_{1i}b_{1j}b_{1l}b_{1m}\det(B_{1i}B_{1j}B_{1l}B_{1m})\,.$$

As before, if any of $\{\!\{i,j,l,m\}\!\}$ differs from the others, the corresponding term vanishes. The surviving contributions are of two kinds. Either $i = j = l = m$, in which case the contribution is

$$\sum_i(-1)^{4i}\mathbb{E}b_{1i}^4\mathbb{E}\det B_{1i}^4 = 3k\mathbb{E}\det B_{11}^4 = 3kf_{k-1}\,. \tag{4.25}$$

Otherwise there are 3 ways in which the multiset $\{\!\{i,j,l,m\}\!\}$ consists of two different pairs of equal indices. The total contribution from these

cases is

$$3\sum_{i\neq j}(-1)^{2i+2j}\mathbb{E}b_{1i}^2\mathbb{E}b_{1j}^2\mathbb{E}\det(B_{1i}^2B_{1j}^2) = 3k(k-1)\mathbb{E}\det(B_{11}^2B_{12}^2)\,. \tag{4.26}$$

We continue by expanding B_{11} and B_{12} along their first column. This is the second and first column, respectively, of the original B. To keep the index gymnastics manageable, we briefly need the notation $B_{I,J}$ for B without the rows in I and the columns in J.

The nonzero contributions are

$$\mathbb{E}\det(B_{11}^2B_{12}^2) = \mathbb{E}\sum_{i=2}^{k}\sum_{j=2}^{k} b_{i2}^2b_{j1}^2\det B_{\{1,i\},\{1,2\}}^2\det B_{\{1,j\},\{2,1\}}^2\,.$$

We note that both b_{i2}^2 and b_{j1}^2 appear independently, because the remaining submatrices avoid the first and second columns of B. Since their expectations are unity, they can be removed from the expression. For $i=j$, both matrices are the same, and the expression collapses to $(k-1)f_{k-2}$. For $i\neq j$, we introduce the shorthand

$$\Phi = \mathbb{E}\det(B_{\{1,i\},\{1,2\}}^2B_{\{1,j\},\{2,1\}}^2) \qquad (i\neq j)\,,$$

observing that all these distributions are the same. We arrive at

$$\mathbb{E}\det(B_{11}^2B_{12}^2) = (k-1)f_{k-2} + (k-1)(k-2)\Phi\,. \tag{4.27}$$

Combining (4.25), (4.26), and (4.27), we obtain

$$f_k = 3kf_{k-1} + 3k(k-1)^2\big(f_{k-2} + (k-2)\Phi\big)\,. \tag{4.28}$$

Using similar arguments from a different starting point, we obtain

$$f_{k-1} = \mathbb{E}\det B_{11}^4 = 3(k-1)f_{k-2} + 3(k-1)(k-2)\Phi\,, \tag{4.29}$$

by expanding B_{11} along the second column; the manipulations rely on the fact that in the definition of Φ, the order in which columns 1 and 2 are deleted plays (of course) no role. Combining (4.28) and (4.29)

yields

$$f_k = k(k+2)f_{k-1}$$

which solves to $f_k = \frac{1}{2}(k!)^2(k+1)(k+2)$.

Remark 4.5.1. We can use this distribution in Sect. 4.3.6 in place of the uniform distribution on $\{+1, -1\}$. The only thing we have to keep in mind is that we still have to be able to perform arithmetic operations in the field. While this is clear for ± 1, our use of irrational numbers here might create some confusion. However, note that we only have to calculate with values coming from the field extension $\mathbb{Q}[\sqrt{3}]$, which can be handled just like complex numbers in spirit (after all, $\mathbb{C} = \mathbb{R}[\sqrt{-1}]$) by representing a number $a + b\sqrt{3}$ by the two rational coordinates a, b, and performing multiplication according to $(a + b\sqrt{3})(c + d\sqrt{3}) = ac + 3bd + (ad + bc)\sqrt{3}$.

4.6. Generalization to Subgraphs

In this section, we formally prove Theorem 4.1.1. We use the homomorphism polynomial as a tool for the computation, and we will evaluate this polynomial over a commutative algebra $\mathcal{A}$ analogous to how this was done in Sect. 4.3.6. For two graph H and G, let $\mathrm{Hom}(H \to G)$ be the set of all functions $h : V(H) \to V(G)$ that are graph homomorphisms from H to G. Then the following is the homomorphism polynomial of H in G:

$$\sum_{h \in \mathrm{Hom}(H \to G)} \prod_{v \in V(H)} \zeta_{h(v)} \,. \tag{4.30}$$

The variables are ζ_v for all $v \in V(G)$. We first show in §4.6.1 that this polynomial has small algebraic circuits when the pathwidth or the treewidth is bounded, and in §4.6.2 we prove Theorem 4.1.1.

4.6.1. Tree Decompositions

Fomin *et al.* [Fom+12b, Lemma 1] construct an algebraic circuit that computes the homomorphism polynomial, based on a dynamic program-

ming algorithm (*e.g.*, [DST02]). We reproduce a proof for completeness.

Lemma 4.6.1. *Let H and G be graphs with $V(H) = \{1, \dots, k\}$ and $V(G) = \{1, \dots, n\}$. There is an algebraic circuit C of size $O\big(k \cdot n^{\mathrm{tw}(H)+1}\big)$ (and an algebraic skew-circuit C of size $O\big(k \cdot n^{\mathrm{pw}(H)+1}\big)$) in the variables $\zeta_1, \dots, \zeta_n$ such that C computes the homomorphism polynomial of H in G in the variables $\zeta_1, \dots, \zeta_n$, that is, we have*

$$C(\zeta_1, \dots, \zeta_n) = \sum_{h \in \mathrm{Hom}(H \to G)} \prod_{v \in V(H)} \zeta_{h(v)}\,. \tag{4.31}$$

The circuit can be constructed in time $O(1.76^k) + |C| \cdot \mathrm{polylog}(|C|)$.

We first need some preliminaries on tree decompositions. A *tree decomposition* of a graph G is a pair $(T, \mathfrak{b})$, where T is a tree and $\mathfrak{b}$ is a mapping from $V(T)$ to $2^{V(G)}$ such that, for all vertices $v \in V(G)$, the set $\{\, t \in V(T) : v \in \mathfrak{b}(t) \,\}$ is nonempty and connected in T, and for all edges $e \in E(G)$, there is some node $t \in V(T)$ such that $e \subseteq \mathfrak{b}(t)$. The set $\mathfrak{b}(t)$ is the *bag at t*. The *width* of $(T, \mathfrak{b})$ is the integer $\max\{\, |\mathfrak{b}(t)| - 1 : t \in V(T) \,\}$, and the *treewidth* $\mathrm{tw}(G)$ of G is the minimum possible width of any tree decomposition of G.

It will be convenient for us to view the tree T as being directed away from the root, and we define the following mappings $\mathfrak{s}, \mathfrak{c}, \mathfrak{a} : V(T) \to 2^{V(G)}$ for all $t \in V(T)$:

$$(\textit{separator at } t) \quad \mathfrak{s}(t) = \begin{cases} \emptyset\,, & t \text{ is the root of } T, \\ \mathfrak{b}(t) \cap \mathfrak{b}(s)\,, & s \text{ is the parent of } t \text{ in } T, \end{cases} \tag{4.32}$$

$$(\textit{cone at } t) \quad \mathfrak{c}(t) = \bigcup_{u \text{ is a descendant of } t} \mathfrak{b}(u), \tag{4.33}$$

$$(\textit{component at } t) \quad \mathfrak{a}(t) = \mathfrak{c}(t) \setminus \mathfrak{s}(t). \tag{4.34}$$

Proof of Lemma 4.6.1. We first compute a minimum-width tree decomposition $(T, \mathfrak{b})$ of H, for example using the $O(1.76^k)$ time algorithm by Fomin and Villanger [FV12]. We can assume it to be a *nice* tree decomposition, in which each node has at most two children;

the leaves satisfy $\mathfrak{b}(v) = \emptyset$, the nodes with two children w, w' satisfy $\mathfrak{b}(v) = \mathfrak{b}(w) = \mathfrak{b}(w')$ and are called *join* nodes, the nodes with one child w satisfy $\mathfrak{b}(v) = \mathfrak{b}(w) \cup \{x\}$ and are called *introduce* nodes, or $\mathfrak{b}(v) \cup \{x\} = \mathfrak{b}(w)$ and are called *forget* nodes.

Recall that $\mathfrak{c}(v)$ is the union of all bags at or below node v in the tree T. Let $S \subseteq V(H)$ and $\pi \in \mathrm{Hom}(H[S] \to G)$ be a partial homomorphism from H to G. To make the inductive definition of the algebraic circuit easier, we define the *conditional homomorphism polynomial* as follows.

$$\hom(\, H \to G \mid \pi \,) = \sum_{\substack{h \in \mathrm{Hom}(H \to G) \\ h \supseteq \pi}} \prod_{v \in V(H)} \zeta_{h(v)} \,. \tag{4.35}$$

The sum is over all homomorphisms h that extend π. With this definition, it is clear that

$$\hom(H \to G) = \sum_{\pi \in \mathrm{Hom}(H[S] \to G)} \hom(\, H \to G \mid \pi \,) \tag{4.36}$$

holds. Moreover, if S is a separator of H, the connected components of $H - S$ conditioned on the boundary constraints π are independent. More precisely, for all $\pi \in \mathrm{Hom}(H[S] \to G)$, we have

$$\hom(\, H \to G \mid \pi \,) = \prod_i \hom(\, H_i \to G \mid \pi \,) \,, \tag{4.37}$$

where $H_1, \ldots, H_\ell$ is a list of graphs such that $H_1 \cup \cdots \cup H_\ell = H$ holds, $V(H_i) \cap V(H_j) = S$ holds for all i, j with $i \neq j$, and $H_i - S$ is connected for all i.

We will construct the final circuit recursively over the tree decomposition (T, β). At node v of T, we construct algebraic circuits C_v^π for each $\pi \in \mathrm{Hom}(\mathfrak{b}(v) \to G)$ such that the following holds:

$$C_v^\pi = \hom(\, H[\mathfrak{c}(v)] \to G \mid \pi \,) \,. \tag{4.38}$$

Note already here that there are at most $n^{|\mathfrak{b}(v)|} \leq n^{\mathrm{tw}(H)+1}$ such functions π. Since each C_v^π represents a gate in our final circuit, and

we will be able to charge at most a constant number of wires to each gate, the number of gates and wires of C will be bounded by $O(|V(H)| \cdot n^{\mathrm{tw}(H)+1})$.

Leaf nodes. Let v be a leaf of T, which has $\mathfrak{c}(v) = \emptyset$, resulting in the trivial circuit $C_v^\pi = 1$ for the empty function $\pi : \emptyset \to V(G)$. Indeed, since $H[\mathfrak{c}(v)]$ has no vertices, the empty function is the unique homomorphism into G.

Introduce nodes. Let v be an *introduce* node of T. Let w be its unique child in the tree. Suppose the vertex $x \in V(H)$ is introduced at this node, that is, $\mathfrak{b}(w) \cup \{x\} = \mathfrak{b}(v)$. Let $\pi \in \mathrm{Hom}(H[\mathfrak{b}(v)] \to G)$ be a partial homomorphism at the bag of v. Then we define C_v^π using the circuit $C_w^{\pi'}$ where $\pi' = \pi \restriction_{\mathfrak{b}(w)}$:

$$C_v^\pi = C_w^{\pi'} \cdot \zeta_{\pi(x)} \,. \tag{4.39}$$

For the correctness, note that the right side of (4.41) is equal to

$$\hom(\, H[\mathfrak{c}(w)] \to G \mid \pi' \,) \cdot \zeta_{\pi(x)} \tag{4.40}$$

by the induction hypothesis (4.38). Since x is the unique vertex in $\mathfrak{c}(v) \setminus \mathfrak{c}(w)$ and π extends π' on x, the polynomial in (4.40) is equal to $\hom(\, H[\mathfrak{c}(v)] \to G \mid \pi \,)$.

Forget nodes. Let v be a *forget* node of T. Let w be its unique child in the tree. Suppose the vertex $x \in V(H)$ is forgotten at this node, that is, $\mathfrak{b}(w) \setminus \{x\} = \mathfrak{b}(v)$. Then the neighborhood of x is contained in $\mathfrak{c}(w)$. Let $\pi \in \mathrm{Hom}(H[\mathfrak{b}(v)] \to G)$. We define C_v^π using the circuits $C_w^{\pi'}$ as follows:

$$C_v^\pi = \sum_{\pi'} C_w^{\pi'} \,. \tag{4.41}$$

The sum is over all $\pi' \in \mathrm{Hom}(H[\mathfrak{b}(w)] \to H)$ that agree with π on the intersection $\mathfrak{b}(v) \cap \mathfrak{b}(w)$ of their domains. Since v is a forget node, this intersection is equal to $\mathfrak{b}(v)$. For the correctness, note that the right

side of (4.39) is equal to

$$\sum_{\pi'} \hom(\, H[\mathfrak{c}(w)] \to G \mid \pi' \,) \tag{4.42}$$

by the induction hypothesis (4.38). This is equal to $\hom(\, H[\mathfrak{c}(v)] \to G \mid \pi \,)$ due to the conditioning formula (4.36). For the size bound, note that x is the only vertex in $\mathfrak{b}(w) \setminus \mathfrak{b}(v)$. Thus, the sum in (4.41) has n terms, and so in this part of the construction we charge at most one wire to each $C_w^{\pi'}$.

Join nodes. Let v be a *join* node of T with children w and w' satisfying $\mathfrak{b}(v) = \mathfrak{b}(w) = \mathfrak{b}(w')$. Let $\pi \in \mathrm{Hom}(H[\mathfrak{b}(v)] \to G)$. We define the circuit simply as

$$C_v^\pi = C_w^\pi \cdot C_{w'}^\pi \,. \tag{4.43}$$

For the correctness, note that $\mathfrak{b}(v)$ is a separator for $H[\mathfrak{c}(v)]$, and so the induction hypothesis (4.38) together with the conditional independence (4.37) yields the correctness. This part of the construction introduces two wires which we charge to C_v^π.

The final circuit is $C = C_r^\emptyset$, where r is the root of T, the degenerate empty function is $\emptyset$, and we assume without loss of generality that $\mathfrak{b}(r) = \emptyset$. As already discussed, we have $O(|V(T)|n^{\mathrm{tw}(H)+1}) = O(|V(H)|n^{\mathrm{tw}(H)+1})$ gates C_v^π, each of which is responsible for $O(1)$ wires incident to it.

Finally, note that if there are no join nodes, then (T, β) is a path decomposition of H and the only multiplications occur in (4.39) and involved at least one variable. Thus, when (T, β) is a minimum-width path-decomposition, the algebraic circuit C constructed above is skew, and has size $O(kn^{\mathrm{pw}(H)+1})$. □

4.6.2. Proof of Theorem 4.1.1 and 4.1.2

Theorem 4.1.1 (restated)**.** There is a randomized algorithm that is given two graphs H and G, and a number $\varepsilon > 0$ to compute an

integer $\tilde{N}$ such that, with probability 99%,

$$(1-\varepsilon)\cdot \mathrm{Sub}(H,G) \leq \tilde{N} \leq (1+\varepsilon)\cdot \mathrm{Sub}(H,G)\,. \tag{4.44}$$

This algorithm runs in time $\varepsilon^{-2}\cdot 4^k n^{\mathrm{pw}(H)+1}\cdot \mathrm{poly}(k)$, where H has k vertices and pathwidth $\mathrm{pw}(H)$, and G has n vertices.

Proof sketch. Let H, G, and $\varepsilon > 0$ be given as input. Let n be the number of vertices of G. By Lemma 4.6.1, we can construct an algebraic skew circuit C that computes the homomorphism polynomial of H in G. The circuit has size $O(kn^{\mathrm{pw}(H)+1})$ and satisfies:

$$C(\zeta_1,\ldots,\zeta_n) = \sum_{h\in \mathrm{Hom}(H\to G)} \prod_{v\in V(H)} \zeta_{h(v)}\,. \tag{4.45}$$

Following exactly the setup of §4.3.6, we use the lifted Bernoulli extensor-coding $\overline{\beta}\colon V(G)\to F^{2k}$. We have

$$C(\overline{\beta}(v_1),\ldots,\overline{\beta}(v_n)) = \pm \sum_{h\in \mathrm{InjHom}(H\to G)} \prod_{v\in V(H)} \overline{\beta}(h(v))\,, \tag{4.46}$$

where $\mathrm{InjHom}(H\to G)$ is the subset of $\mathrm{Hom}(H\to G)$ that consists of all homomorphisms that are injective. Now we use Algorithm C, except that we replace $f(G;\overline{\beta})$ with $C(\overline{\beta}(v_1),\ldots,\overline{\beta}(v_n))$. The rest goes through as in Theorem 4.3.5. Note that this approach using (4.46) actually approximates the number of injective homomorphisms, which however gives rise to an approximation (of the same quality) for $\mathrm{Sub}(H,G)$ when we divide by the size $|\mathrm{Aut}(H)|$ of the automorphism group of H. The size of the automorphism group of H can be computed in advance, in time $O(1.01^k)$, by a well-known $\mathrm{poly}(k)$-time reduction to the graph isomorphism problem [Mat79], which in turn can be computed in time $\exp(\mathrm{poly}\log k) \leq O(1.001^k)$ [Bab16]. For the running time of the modified Algorithm C, note that C is a skew circuit, and skew multiplication in $\Lambda(F^{2k})$ takes time $O(4^k)$. Thus, the overall running time is $O(\varepsilon^{-2}4^k|C|)$. □

Theorem 4.1.2 in the case of paths is established through Theorem

4.3.8. For the more complicated case of general subgraphs, we can modify the polynomial (4.30) analogously so that it involves also the edge variables. By suitable modifications of the dynamic program that computes the corresponding circuit, Theorem 4.1.2 is established.

4.7. Proof of Theorem 4.1.3

In this section, we prove Theorem 4.1.3. This will follow by an application of our algebraic interpretation of color-coding from Sect. 4.4.4. In particular, we will again make use of the Zeon algebra $Z(F^k)$.

The next proposition is a trivial consequence of Lemma 4.4.4.

Proposition 4.7.1. *For any integer $t > 0$, an arithmetic circuit C over $\mathbb{Z}[\zeta_1, \ldots, \zeta_n]$ can be evaluated over $Z(\mathbb{Q}^t)$ in $2^t \cdot |C| \cdot \mathrm{poly}(n)$ operations over $\mathbb{Q}$.*

We are ready for the proof:

Proof of Theorem 4.1.3. We invoke Proposition 4.7.1 with Hüffner *et al.*'s [HWZ08] choice of $t = 1.3k$. One evaluation costs $2.4623^k \cdot |C| \cdot \mathrm{poly}(n)$ operations over $\mathbb{Q}$. The classical color-coding approach would evaluate C at the generators $\overline{\mathbf{e}_i}$, where $1 \leq i \leq t$ is chosen uniformly at random. In this way, all non-multilinear terms will vanish, but distinct monomials might cancel when being mapped to the same product of $\overline{\mathbf{e}_i}$. To avoid this, we randomly scale each generator, and plug in $\alpha_i \cdot \overline{\mathbf{e}_j}$ at the ith input of the circuit, for random $\alpha_i \in \{0, 1, \ldots, 100 \cdot k\}$ and random $1 \leq j \leq t$. The circuit C then evaluates to some multiple of $\overline{\mathbf{e}_t}$, and the coefficient of $\overline{\mathbf{e}_t}$ in the result is a multilinear polynomial in the $\alpha_i, 1 \leq i \leq n$. By the DeMillo–Lipton–Schwartz–Zippel Lemma 3.3.3, the polynomial evaluates to something non-zero with constant probability of 99%. Following Hüffner *et al.* [HWZ08, Theorem 1] (cf. Sect. 3.1.3), the probability that some multilinear term maps to a multiple of the t generators is at least $\Omega(1.752^{-k})$, and we derive the total running time of $4.32^k \cdot |C| \cdot \mathrm{poly}(n)$ operations in $\mathbb{Q}$. Now, if the circuit can be evaluated over $\mathbb{Z}$ in polynomial time (*i.e.*, all numbers stay of appropriate size), this costs only a polynomial overhead, and

the claim follows. However, by repeated squaring, the circuit C may generate numbers of value 2^{2^n}, so we calculate modulo some random prime. Numbers of bitlength $O(2^n)$ may have up to $O(2^n)$ prime factors, so choosing a random prime p from the first $\Omega(n2^n)$ primes, we find a value for p such that the resulting coefficient doesn't vanish modulo p with probability $1 - o(1)$. By the prime number theorem, the first $n2^n$ primes are of magnitude $2^n \operatorname{poly}(n)$, and we can thus randomly pick a number from $\{1, \ldots, P\}$, where $P = 2^n \operatorname{poly}(n)$, until we find a prime (which can be tested in randomized polynomial time). Then we perform all the above calculations modulo p (trivially, this can be done in time $\operatorname{poly}(n)$), and if the result doesn't vanish, the polynomial doesn't vanish over $\mathbb{Z}$. On the other hand, if it vanishes, we might have had bad luck, but as argued, this only happens with probability $1/n$, which is fine. □

5. Faster Deterministic Algorithms

5.1. Introduction

We have seen the prime application of Extensor-Coding to the longest path problem in the previous chapter. Despite yielding an interesting and conceptually very simple deterministic algorithm for this problem (cf. Proposition 5.3.4), the technique does not seem to give improvements over the state-of-the-art in this setting.

In this chapter, we will consider a range of problems where the method *does* give improved deterministic fixed-parameter tractable algorithms, or even the first fixed-parameter tractable algorithms at all.

To this end, we start from a novel and natural variant of the ubiquitous multilinear detection problem on polynomials computed by arithmetic circuits that is more general than the very well-studied case of cancellation-free circuits. This leads us to a rather generic approach of patching, if you will, algorithms involving Color-Coding, by replacing the Colors used in the algorithm by Extensors, *i.e.*, elements of the exterior algebra. This then yields an exponential speed-up.

5.1.1. Results

The algorithmic problems we study are the following: The *k-internal out-branching (k-IOB)* problem asks for the existence of an out-branching (also called a directed spanning tree or arborescence, among others) with at least k internal, *i.e.*, non-leaf, nodes. The *k-internal spanning (k-IST)* tree problem is formulated analogously for undirected

graphs. A subset of edges in an edge-colored graph is *k-colorful* if it contains edges of at least k distinct colors, and the definitions of the problems *k-colorful perfect matching* and the *k-colorful out-branching* are self-evident.

Our algorithmic advances in these problems are borne by progress in another domain: The *k-multilinear detection (k-MLD)* problem has as an input a multivariate polynomial represented through an arithmetic circuit, and asks whether the polynomial contains a monomial of degree k in which no variable appears with degree more than one. We focus our attention on the restriction of the problem to those instances where a multilinear monomial may only appear with positive coefficient. Since there is no known way to efficiently test this property, this turns the problem into a promise problem. The promise, however, is typically satisfied in combinatorial applications, where the studied polynomials are usually *multivariate generating functions* of the sought combinatorial objects. The decisive subtlety here is that, while the *polynomial* that is computed by the input circuit may not have negative coefficients in its multilinear part, the circuit *itself* may very well contain negative constants and make use of cancellations, and, in particular, it need *not* be monotone. We will make heavy use of this property in the above application problems, and it is decisive here for the following reason: These problems can be expressed using determinantal generating functions, and by a theorem of Jerrum and Snir [JS82], determinants do *not* have monotone circuits of subexponential size, such that cancellations are actually crucial for their efficient computation. To the best of our knowledge, this is the first fixed-parameter tractable algorithm for the problem.

We prove the following deterministic, exponential-space record time bounds, and defer the reader to Sect. 5.3 for a formal statement of the theorem.

Theorem 5.1.1 (Informal)**.** *There are deterministic algorithms to solve*

1. *the k-internal out-branching problem and the k-internal spanning tree problem in time* $3.21^k \cdot \mathrm{poly}(n)$*, and*

2. *the k-colorful perfect matching problem on planar graphs and the k-colorful out-branching problem in time $4^k \cdot \mathrm{poly}(n)$.*

Furthermore, there is a deterministic algorithm that solves the restriction of the multilinear detection problem to circuits computing polynomials with positive coefficients in their multilinear part (as laid out above) in time $4^k \cdot \mathrm{poly}(n)$ on skew arithmetic circuits, and in time $2^{\omega k} \cdot \mathrm{poly}(n) < 5.19^k \cdot \mathrm{poly}(n)$ on general circuits, where ω is the exponent of matrix multiplication.

Remark 5.1.2. The bound of 5.19 is not competitive; indeed it is easy to prove that an exponential basis of 4.312 can be obtained using a derandomization of Color-Coding, and Pratt [Pra18; Pra19] gives a randomized algorithm achieving 4.075. Note, however, that our bound depends on ω, and one can make the point (albeit moot in the foreseeable future) of this dependency making our bound potentially competitive.

Remark 5.1.3. Skewness, *i.e.*, the syntactic restriction of each multiplication gate having an input as an operand seems rather strong at a first glance. At a second glance, this impression does not hold up: Without concerning ourselves with the technicalities of algebraic complexity theory, suffice it to say that the polynomials that can be computed by efficient skew circuits are precisely those that are efficient projections of determinants, and determinantal generating functions are known for a variety of combinatorial objects. Equivalently, they are those polynomials that are computed by efficient algebraic branching programs (which is a widely studied and very natural computational model). Therefore, skew circuits are not as restrictive as it might seem.

The algorithms for the problems of detecting a k-internal as well as k-colorful out-branching (and spanning tree) are established, as demonstrated by Björklund *et al.* , via the Directed Matrix-Tree Theorem [BKK17]. We reuse the meticulous and very careful analysis of the parallel monomial sieving technique by Gutin *et al.* [Gut+18]. We can relatively easily replace the employed pseudorandom objects by

extensors. This suggests a rather generic way in which to speed up algorithms based on Color-Coding by instead using Extensor-Coding.

As far as the k-colorful planar matching problem is concerned, as in [Gut+18], we use a Pfaffian computation. However, we cannot perform the square root extraction that is necessary to make use of the determinantal identites for the Pfaffian, but rely on a result of Flarup *et al.* [FKL07] for a direct computation of Pfaffians with skew arithmetic circuits.

5.1.2. Related work

The k-internal out-branching and spanning tree problems have attracted a significant amount of attention over the last years [Fom+12a; Gut+18; Bjö+17b; BKK17; Li+17; Zeh17; Zeh15; Fom+13; Dal11; Coh+10; GRK09; FGR09; PS05]. The current deterministic record bounds for all the abovementioned graph problems were recently given in Gutin *et al.* [Gut+18], using monomial sieving in combination with Color-Coding and suitable pseudorandom objects. The bounds they obtain (in the exponential-space setting) for the k-internal out-branching and spanning tree problems are $3.41^k \cdot \text{poly}(n)$, and $4.32^k \cdot \text{poly}(n)$ for the k-colorful perfect matching and out-branching problems.

The detection of k-multilinear terms in polynomials computed by arithmetic circuits lies at the heart of the design of the fastest randomized algorithms for a host of parameterized problems, such as the longest path problem, the k-tree problem, the t-dominating set problem and the m-dimensional k-matching problem, with a record bound of $2^k \cdot \text{poly}(n)$ for randomized k-multilinear detection [KW16; Wil09]. The crux is that, for the arithmetic circuits in these algorithms, it is required that they be *monotone*, i.e., do not involve any cancellations of terms. On this class of monotone arithmetic circuits, the k-multilinear detection problem can be solved deterministically in time $3.85^k \cdot \text{poly}(n)$, using the combinatorial notion of representative sets [Fom+14].

Recently, the first fixed-parameter tractable *randomized* algorithms were developed for the problem on general arithmetic circuits [BDH18], which has sparked further work in the area, announcing a polynomial-

space version and $4.08^k \cdot \mathrm{poly}(n)$ as a new record bound on general circuits [Arv+18b; Arv+18a; Pra18; Pra19].

5.2. Monomial Detection Problems

We will first give a rather direct, but very useful application to the multilinear detection problem in a special case. Recall that the general problem presents itself as follows: As input, it obtains a multivariate (and now again commutative) polynomial $f \in \mathbb{C}[X_1, \ldots, X_n]$ in n indeterminates, encoded as an arithmetic circuit. Our task is now to decide whether or not f contains a monomial of degree k such that no indeterminate appears twice.

A variation of this is to ask whether f contains a monomial such that at least k distinct indeterminates appear in it.

In this chapter, we restrict our attention to a semantically defined subclass of arithmetic circuits: Those that compute a polynomial f such that every multilinear monomial that appears in f does so with positive coefficient. It is crucial to note that this does *not* mean a monotonicity restriction for the input circuit, and it may well involve cancellations of terms and negative constants. Let us formally define the set of circuits we are interested in:

Definition 5.2.1. Let C be an arithmetic circuit. We call C *combinatorial* if the polynomial computed by C has non-negative coefficients on its multilinear part, and C can be evaluated over $\mathbb{Z}$ at numbers of absolute value at most τ using $\mathrm{poly}(\tau)$ bit operations.

We remark that the last condition of this definition is a barely concealed crutch to avoid having to think about subtleties regarding a possible doubly exponential blowup of inputs in general arithmetic circuits, which are irrelevant in our applications. For a very similar reason, we only speak about evaluation over $\mathbb{Z}$; namely, in order to ignore potential issues with representations of complex numbers.

5.2.1. Multilinear Detection

As promised, we will now start out with an easy application of Extensor-Coding. Speaking of promises, it is in order to remark that the problems discussed henceforth are promise problems, in the sense that there is no known efficient method of checking whether an input circuit satisfies the condition of being combinatorial.

We state the upcoming Theorem 7.5.2, which we state here without proof:

Theorem 7.5.2. Given two elements $x, y \in \Lambda(\mathbb{C}^k)$ as a list of basis coefficients, their product $x \wedge y \in \Lambda(\mathbb{C}^k)$ as a list of basis coefficients can be computed using $2^{\omega k/2} \cdot \mathrm{poly}(k)$ arithmetic operations over $\mathbb{C}$.

Additionally, if the size of coefficients of x and y is bounded by 2^τ, then their product can be computed in $2^{\omega k/2} \cdot \mathrm{poly}(\tau)$ bit operations.

This is enough to prove:

Theorem 5.2.2. *There is a deterministic algorithm that, given a combinatorial arithmetic circuit C of size s and an integer k, decides whether or not the polynomial computed by C contains a multilinear monomial of degree k in time $2^{\omega k} \cdot \mathrm{poly}(s) < 5.19^k \cdot \mathrm{poly}(s)$.*

Proof. Consider the lifted Vandermonde coding $\overline{\phi}$, and assume that the polynomial computed by C is n-variate.

The algorithm then simply evaluates C at $(\overline{\phi}(1), \ldots, \overline{\phi}(n))$, and outputs 'yes' if and only if the coefficient of $\mathbf{e}_{[2k]}$ in the resulting element of $\Lambda(V \oplus V)$ is non-zero.

The running time is immediate from the definition and the upcoming Theorem 7.5.2, and correctness can be seen as follows. We first have to take care of the fact that our algebra is not commutative, strictly speaking, but nevertheless we evaluate a commutative polynomial over it. This is remedied either by a standard degree-k homogenization argument on C, and, depending on k, a subsequent single sign correction of the result. Alternatively, one can observe that the signs are consistent across each degree individually, and monomials of different degrees are linearly independent, so that the different signature arising when

evaluating over the image of $\overline{\phi}$ will not make a difference. As a third alternative, we can instead consider elements $\phi(c) \otimes \phi(c) \in \Lambda(\mathbb{C}^k)^{\otimes 2}$, which generate a commutative subalgebra. This will become important in Chap. 8.

From the fact mentioned in the discussion about Vandermonde codings, every monomial containing an indeterminate twice will vanish. The parts of degree less than k do not enter into the coefficient of $\mathbf{e}_{[2k]}$, and parts of degree more than k will go to zero anyways. Now, let L be the set of multilinear monomials in the polynomial computed by f. Each such monomial m identifies a subset of $\{1, \ldots, n\}$, and by abuse of notation, we will not distinguish between a monomial and a subset. Furthermore, we denote with c_m the coefficient of m, which is non-negative by the assumption on C, and let V_m be the $2k \times 2k$ matrix $(\overline{\phi}(i))_{i \in m}$. Then the coefficient of $e\mathbf{e}_{[2k]}$ can be seen to be equal to $\sum_{m \in L} c_m \cdot \det(V_m)^2$, which is non-zero if and only if one of the determinants is non-zero. This in turn happens if and only if there is a multilinear monomial of degree k in the polynomial computed by C. □

We also obtain the more useful skew variant:

Theorem 5.2.3. *There is a deterministic algorithm that, given a skew combinatorial arithmetic circuit C of size s and an integer k, decides whether or not the polynomial computed by C contains a multilinear monomial of degree k in time $4^k \cdot \mathrm{poly}(s)$.*

Proof. Follows verbatim like Theorem 5.2.2 after replacing Theorem 7.5.2 by Proposition 4.2.1. □

Remark 5.2.4. Unlike Theorem 5.2.2, this bound is actually competitive, and we are not aware of a possibility to exploit skewness using other approaches.

5.2.2. k-Distinct Detection

We will now turn to the monomial detection problem that will later on be used in applications, namely the *k-distinct detection problem*. Again,

the input here is an arithmetic circuit, but this time, the task is to decide whether there exists a monomial containing at least k distinct indeterminates. Using the folklore trick of replacing every variable x_i by $1 + t \cdot x_i$ with a formal indeterminate t, turning a nilpotent variable x_i into an (almost) idempotent one, and then extracting the coefficient of t^k, we obtain:

Theorem 5.2.5. *There is a deterministic algorithm that, given a skew combinatorial arithmetic circuit C of size s that computes a polynomial that only has non-negative coefficients, and an integer k, decides whether or not the polynomial computed by C contains a monomial with at least k different variables in time $4^k \cdot \mathrm{poly}(s)$.*

Proof. Add after every input gate x_i a gate of the form $1 + tx_i$, where t is a fresh formal indeterminate commuting with everything. Since $(1 + tx_i)^2 = 1 + 2tx_i + t^2x_i^2$, when plugging in extensors, we are left with $1 + 2tx_i$, since the square vanishes. Therefore, in an arbitrary s-product, when $x_i^2 = 0$ for all i, we have

$$\prod_{i=1}^{s}(1 + tx_i) = \sum_{S \subseteq \{1,\ldots,s\}} t^{|S|} \prod_{s \in S} x_s$$

and this is non-zero in t-degree k if and only if there is one term containing at least k different variables. Note that if C was combinatorial, it will again be combinatorial after applying this modification. Since all appearing coefficients are positive by assumption, this substitution will not introduce unwanted cancellations. It also keeps the circuit skew (or at least, it can be easily made skew again). We can then just apply the algorithm for multilinear detection from before, noting that arithmetic in this new ring with t adjoined, for the skew case, can again be performed in the required time bound. □

Equipped with these observations, we may now turn to our application problems.

5.3. Graph Problems

We will make use of the classic Directed Matrix-Tree Theorem, following the presentation in [BKK17]. Let us first define the *Laplacian* of a directed graph $G = (D, A)$. To this end, let $X = \{x_a \mid a \in A\}$ be a set of formal indeterminates labeled with the arcs of a graph, and define the matrix $L = (\ell_{uv})_{u,v \in V}$ through

$$\ell_{uv} = \begin{cases} \sum_{w \in V : wu \in A} x_{wu} & \text{if } u = v \\ -x_{uv} & \text{if } uv \in A \\ 0 & \text{if } uv \notin A \end{cases} .$$

After fixing a root $r \in V$, we will consider L_r, the *Laplacian punctured at* r, which is defined as the matrix obtained from L by striking row r and column r. With these definitions in place, we have the following well-known theorem, and just as [BKK17], we refer to the corresponding chapter of Gessel and Stanley in the Handbook of Combinatorics [GS95] for a proof.

Theorem 5.3.1 (Directed Matrix-Tree Theorem)**.** *Let $G = (D, A)$ be a directed graph. For all $r \in V$, the following holds.*

$$\det L_r = \sum_{\substack{T = (V, B) \text{ is an} \\ \text{out-branching of } G \\ \text{rooted at } r}} \prod_{b \in B} x_b \, .$$

In other words, the determinant of L_r is the multivariate generating function of the set of out-branchings rooted at r. The important insight is now the following: All known (randomized) efficient algorithms for detecting k-multilinear terms—and, by extension, k-distinct terms—in the polynomial computed by an arithmetic circuit rely on this circuit not involving cancellations in their computation, *i.e.*, they need to be *monotone*.

However, by a theorem of Jerrum and Snir [JS82], computing $\det L_r$ using such a monotone circuit requires circuits of *exponential size* in n.

On the other hand, there are efficient skew arithmetic circuits (this

time with cancellations) for computing the $n \times n$ determinant polynomial:

Theorem 5.3.2 ([Tod92])**.** *There is a family of skew arithmetic circuits $(C_n)_{n\in\mathbb{N}}$ such that C_n computes the $n \times n$ determinant polynomial, and the size $s(n)$ of C_n satisfies $s(n) \leq \mathrm{poly}(n)$. Furthermore, there is an algorithm that, upon input 1^n, outputs a description of C_n in time $\mathrm{poly}(n)$, and every circuit can be evaluated over $\mathbb{Z}$ in polynomial time in the length of the input representation.*

In fact, the $n \times n$ determinant polynomial is complete—for a suitable notion of reduction—for the adequately named class VDET of polynomial families computable by $\mathrm{poly}(n)$-sized skew arithmetic circuits, defined *ibid.* Importantly, this together with the Matrix-Tree Theorem 5.3.1 also shows that $\det L_r$ is a combinatorial polynomial.

5.3.1. Out-Branchings

Let us now proceed gently with a first application of what we have gathered so far.

Theorem 5.3.3. *There is a deterministic algorithm that, given a directed edge-colored graph D on n vertices and an integer k, decides whether D has a k-colorful out-branching in time $4^k \cdot \mathrm{poly}(n)$.*

Proof. First, replace every variable x_a by a fresh variable corresponding to its color $c(a)$, say $y_{c(a)}$, and denote the corresponding symbolic matrix with $L_r(y)$. Since $\det L_r$ is combinatorial and skew, so is $\det L_r(y)$, and we can perform the k-distinct detection from Theorem 5.2.5 in the claimed running time. The existence of a monomial with k distinct variables in $\det L_r(y)$ is now clearly equivalent to the existence of a k-colorful out-branching in D. □

Theorems 5.3.1, 5.3.2 and Theorem 5.2.5 immediately yield a deterministic algorithm for the problem, running in time $4^k \cdot \mathrm{poly}(n)$. We note that this is already a significant improvement over the time bound of $5.14^k \cdot \mathrm{poly}(n)$ on the runner-up algorithm of [Zeh15].

Proposition 5.3.4 (Superseded by [Gut+18])**.** *There is a deterministic algorithm that, given a directed graph D on n vertices and an integer k, decides whether D has a k-internal out-branching in time $4^k \cdot \mathrm{poly}(n)$.*

Proof. First, replace every variable x_{uv} be a fresh variable corresponding to its tail, say y_u, and denote the corresponding symbolic matrix with $L_r(y)$. Since $\det L_r$ is combinatorial and skew, so is the n-variate polynomial $\det L_r(y)$, and we can perform the k-distinct detection from Theorem 5.2.5 in the claimed running time. It has been observed by Björklund *et al.* [BKK17] that this is neatly equivalent to the input instance containing a k-internal out-branching. □

To speed this up, we can more or less just plug in an Extensor-Coding into the analysis from [Gut+18] and set a new record bound for the k-internal out-branching problem.

Theorem 5.3.5. *There is a deterministic algorithm that, given a directed graph D on n vertices and an integer k, decides whether D has a k-internal out-branching in time $3.22^k \cdot \mathrm{poly}(n)$.*

The corresponding results for k-internal spanning trees of undirected graphs follow immediately by standard reductions to the directed case (see [Gut+18]).

Proof. We follow very closely the proof of Theorem 9 in [Gut+18], and refer the reader thither for all necessary definitions and terminology.

There, Gutin *et al.* iterate over a set of splitters and color a subset of the variables, (namely the vertices that are not contained in a particular perfect matching), of the associated Matrix-Tree determinant according to the current splitter in each iteration.

Then, they apply a monomial sieving argument on the remaining variables, but do so in an efficient fashion: Instead of iterating through all of the subset lattice of the set of variables of each considered monomial every time, one can make use of the union of the subset lattices having large intersections, and compute the sieving in time proportional in the number of subsets that are sieved over *in total*. This was shown by Björklund *et al.* [Bjö+10].

The approach we choose is slightly different: Instead of using a splitter to color the variables (which are called y_i in [Gut+18]), we assign to those variables values similar to Vandermonde codings, just as we do it with the inputs of the circuit in our algorithm for k'-distinction in the proof of Theorem 5.2.5. We leave the remainder of the indeterminates, which are labelled x_i in [Gut+18], intact. The coefficient of $\mathbf{e}_{[2k]}$ in the resulting expression is then again a polynomial in the remaining variables x_i, and we then use the trimmed monomial sieving technique (*i.e.*, compute the different evaluations of $Q_F(X, Y)$ in the language of [Gut+18]).

The running time analysis from [Bjö+10] then has to be adapted slightly: Of course, the bounds on the size of $\mathcal{J}_c$ are still valid. However, the factor corresponding to the size of the splitter and $2^{\alpha^\star k'}$, whose product is bounded by $4.312^{k'}$, becomes irrelevant. Instead, we have to take into account the cost of computation when evaluating the Q_F. The evaluation of the appearing determinants Q_F can be performed in time $4^{k'}$, as argued in Sect. 5.3. Effectively, this replaces the factor 4.312 by a factor of 4, and this leads to the claimed time bound. □

5.3.2. Colorful Planar Perfect Matchings

Just like out-branchings, perfect matchings in planar graphs have an efficiently computable multivariate generating function, namely the *Pfaffian* $\operatorname{Pf} A$ of a suitable skew-symmetric matrix A. Gutin *et al.* [Gut+18] employ a determinantal identity for the Pfaffian, and do so by evaluating the determinant polynomial in question over the integers in a black-box fashion. Namely, they exploit that $\operatorname{Pf}(A)^2 = \det(A)$. In the very last step, this requires a square root extraction, which is a perfectly viable path over the integers, but not over more general algebras. Therefore, instead of construing $\operatorname{Pf}(A)$ as $\sqrt{\det(A)}$, we rely on an observation of Flarup *et al.* [FKL07], who show that the Pfaffian has efficient skew circuits. Putting this together, we immediately obtain:

Theorem 5.3.6. *There is a deterministic algorithm that, given an undirected edge-colored planar graph G on n vertices and an integer k, decides whether G has a k-colorful perfect matching in time $4^k \cdot \operatorname{poly}(n)$.*

Part II.

Related Algebraic Questions

6. Introduction

Die Linie ist ein Punkt, der spazieren geht.[1]

Paul Klee

The algorithms of the first part relied heavily on computations in the exterior algebra. In this part, we will consider in detail the algebraic foundations of these algorithms. We will do so in two directions:

On the one hand, we will consider the more general problem of computing arbitrary wedge products. Despite this operation's pervading all of multilinear algebra, to the best of our knowledge, there is no explicit account of how to perform it quickly. We address this gap in the literature and give a streamlined account of the best known method to compute the wedge product. Unfortunately, we are not able to improve upon the state-of-the-art, and leave this as a challenging open problem. All this is done in Chapt. 7.

On the other hand, we consider a more specialized algebraic object, which arises naturally from a more detailed analysis of the Extensor-Coding method. Namely, this is a subalgebra of the tensor square of an exterior algebra, generated by certain elements associated with points on the rational normal curve. We determine precisely the Hilbert function of this (commutative) subalgebra, and to do so, give a description as a quotient of a polynomial algebra. We finally sketch some recently uncovered connections to apolar algebras that are outside the scope of this thesis. This is done in Chapt. 8.

[1] "The line is a point taking a walk."

7. Computing in the Exterior Algebra

Generally speaking, the question for the complexity of multiplication in some fixed algebra is classic within algebraic complexity theory, and we refer the reader to the equally classic book of Bürgisser, Clausen and Shokrollahi [BCS97] for a thorough introduction to this field.

In this thesis, we have previously seen a variety of algorithmic applications of the exterior algebra, all employing Extensor-Coding. One obstacle to make this technique competitive for a wider range of problems is the efficient implementation of the multiplication map in the exterior algebra, which we will now study.

Formally, we will consider the following problem. Let k be a natural number, and let x, y be two elements of the exterior algebra $\Lambda(\mathbb{C}^k)$, given as a list of coordinates, such that each coordinate can be represented using at most τ bits. What is, in terms of k and τ, the complexity of performing the multiplication in $\Lambda(\mathbb{C}^k)$, *i.e.*, computing the list of coefficients of the wedge product $x \wedge y$?

By the sheer input size (recall that x and y have 2^k coordinates each), this complexity bound will be of the form $c^k \cdot \mathrm{poly}(\tau)$ for some constant $c \geq 2$. More precisely, we are mostly interested in determining a good upper bound for c, and since constant factors are hidden in the $\mathrm{poly}(\tau)$-term, we may without loss of generality assume that k is even, say with $k = 2s$.

In this chapter, we will give a streamlined account of the state-of-the-art for this problem, summarizing and simplifying several related results. In particular, we will give a unified and explicit account of the results implicit in the works of Leopardi [Leo+05] and Włodarczyk

[Wło16].

We will first turn to introducing a new family of algebras related to the exterior algebra, which are called *Clifford algebras*, and detail the algorithms that can be used to permorm multiplication in these algebras. Then, we will shed some light on this relationship and see how exterior algebras can be obtained from Clifford algebras. Finally, we will put this relation to use to obtain the best known algorithms for performing the multiplication in the exterior algebra, *i.e.*, the wedge product. Afterwards, we will also spell out the relation between the well-known and fundamental techniques for computing different kinds of subset convolutions.

7.1. Complex Clifford Algebras

Let us first introduce a family of algebras, $Cl_k(\mathbb{C})$ for $k \in \mathbb{N}$, the *complex Clifford algebras*. There is a rich and very general theory about these algebras, but we concern ourselves mainly with one special case that is most relevant in all applications. We will proceed as concretely as possible with our definition of $Cl_k(\mathbb{C})$, reminiscient of the way the exterior algebra was introduced back in Subsection 4.2.1. Since our algorithms will concern only complex Clifford algebras, we will restrict our attention to this case. In fact, as a *linear* space, $Cl_k(\mathbb{C})$ and $\Lambda(\mathbb{C}^k)$ are indistinguishable: We again extend $\mathbb{C}^k$ with standard basis $\{\mathbf{e}_1, \ldots, \mathbf{e}_k\}$ to a 2^k-dimensional vector space $Cl_k(\mathbb{C})$ by defining the basis vectors $\mathbf{e}_I$ of $Cl_k(\mathbb{C})$ by a subset I of indices from $\{1, \ldots, k\}$. By still confusing $\mathbf{e}_i$ with $\mathbf{e}_{\{i\}}$ for $i \in \{1, \ldots, k\}$, we can view $\mathbb{C}^k$ as a subspace of $Cl_k(\mathbb{C})$ spanned by the singleton basis vectors.

The way in which $Cl_k(\mathbb{C})$ is turned into an *algebra* is very similar to the case of $\Lambda(\mathbb{C}^k)$. Indeed, we define a multiplication, denoted by the ordinary symbol $\cdot$ or by juxtaposition of factors, on the elements of $Cl_k(\mathbb{C})$ just as in the case of $\Lambda(\mathbb{C}^k)$ (*i.e.*, associative, multilinear etc.), with only one crucial difference: On a pair of basis vectors $\mathbf{e}_I$ and $\mathbf{e}_J$, we set $\mathbf{e}_I \cdot \mathbf{e}_J = \pm\mathbf{e}_{I \Delta J}$, where Δ is the symmetric set difference. To determine the sign, note that $\operatorname{sgn}(I, J)$, defined as the

sign of "the" permutation that brings the sequence $i_1, \ldots, i_r, j_1, \ldots, j_s$ into increasing order, is now not well-defined anymore: It can now happen that $i_t = j_u$ for some t, u, and in this case, we can produce at least two permutations of differing parity by swapping i_t and j_u. To this end, we extend the definition of $\operatorname{sgn}(I, J)$ for non-disjoint I and J to refer to what is known as a *stable* ordering in algorithms: In the permuted sequence, i_s is required to appear before j_t whenever $i_t = j_u$. Then, $\operatorname{sgn}(I, J)$ is the parity of *the* permutation that brings $i_1, \ldots, i_r, j_1, \ldots, j_s$ into this order.

This retains the property of the exterior algebra that $\mathbf{e}_1 \wedge \mathbf{e}_2 = -\mathbf{e}_2 \wedge \mathbf{e}_1$. Altering the example given for the exterior algebra, we can calculate for instance $(\mathbf{e}_1 \cdot \mathbf{e}_3) \cdot (\mathbf{e}_4 \cdot \mathbf{e}_1) = \mathbf{e}_1\mathbf{e}_3\mathbf{e}_4\mathbf{e}_1 = -\mathbf{e}_1\mathbf{e}_3\mathbf{e}_1\mathbf{e}_4 = \mathbf{e}_1\mathbf{e}_1\mathbf{e}_3\mathbf{e}_4 = \mathbf{e}_3\mathbf{e}_4$.

From a computational point of view, it is trivial to add and scale elements of $Cl_k(\mathbb{C})$, just as it was for $\Lambda(\mathbb{C}^k)$, by using component-wise operations. We record this as a proposition.

Proposition 7.1.1. *Let A be any one of $Cl_k(\mathbb{C})$ or $\Lambda(\mathbb{C}^k)$. For any two elements $x, y \in A$ and $\lambda \in \mathbb{C}$ given as a list of basis coefficients, the sum $x + y$ and the scalar multiple $\lambda \cdot x$ can be computed in 2^k arithmetic operations over $\mathbb{C}$*

Additionally, if the size of each coefficient of x, y and λ are of bitlength bounded by τ, then the sum $x + y$ and the scalar multiple $\lambda \cdot x$ can be computed in $2^k \cdot \operatorname{poly}(\tau)$ bit operations.

In the remainder of this chapter, we will study the question of how to efficiently multiply two *arbitrary* elements of $Cl_k(\mathbb{C})$ or $\Lambda(\mathbb{C}^k)$. To reiterate, this means mainly determining a good upper bound c for the running time bound $c^k \cdot \operatorname{poly}(\tau)$ to perform this product on elements with coefficient length bounded by τ. Trivial exhaustive application of the distributive law and the rules of multiplying basis elements (of both $Cl_k(\mathbb{C})$ and $\Lambda(\mathbb{C}^k)$) yields $c \leq 4$. A slightly less trivial approach can be used to bring this down to $c \leq 3$, by spending roughly 2^i time for the computation of the coefficient of some basis element that is composed of i generators, and hence $\sum_{i=0}^{k} \binom{k}{i} 2^i = 3^k$ time in total.

Improving over this will lead us to considering so-called *representations* of $Cl_k(\mathbb{C})$: As a useful analogy, recall that every finite group G is isomorphic to a subgroup of the group of permutations on the symbols G. This goes by realizing that for each fixed $g \in G$, the left multiplication $G \to G, x \mapsto gx$ is itself a permutation on G. Similarly, every finite dimensional algebra A is isomorphic to a subalgebra of the full matrix algebra of $\dim A \times \dim A$ matrices, since for each fixed $a \in A$, the left multiplication map $A \to A, x \mapsto ax$ is linear, which yields the sought isomorphism after choosing a basis for A. Note that the dimension of the full matrix algebra is $\dim(A)^2$, *i.e.*, we are actually constructing an isomorphism to a $\dim A$-dimensional subalgebra of $\mathbb{C}^{\dim A \times \dim A}$, which, in our case, will turn out to be isomorphic to a full matrix algebra of matrices of size $\sqrt{\dim A} \times \sqrt{\dim A}$, say via an isomorphism $\rho : A \to \mathbb{C}^{\sqrt{\dim A} \times \sqrt{\dim A}}$.

The idea is now to make use of this: We can hope to be able to compute efficiently the isomorphism ρ and its inverse, meaning in time roughly linear in $\dim(A)$ and the representation length of the input coordinates. In this case, to multiply two elements with coordinates of bitlength τ of A given by their basis representation, it suffices to compute their images in the matrix algebra under ρ, then multiply two matrices using fast matrix multiplication, and then invert ρ on the result to get back an element of A as a list of basis coefficients. This takes time $\dim(A)^{\omega/2} \operatorname{poly}(\tau)$ in total, which in the case of $A = Cl_k(\mathbb{C})$ results in an algorithm using $2^{\omega k/2} \cdot \operatorname{poly}(\tau)$ bit operations, *i.e.*, $c \leq 2^{\omega/2}$.

Let us remark that the recent breakthrough result of Umans [Uma19] on generalized Fast Fourier Transforms can be used to recover, up to an infinitesimal factor of $2^{\varepsilon k}$ in the running time, the algorithms for Clifford algebras in this section. More precisely, even though Clifford algebras are not group algebras, they are quotients of group algebras much in the same way exterior algebras are quotients of monoid algebras (see Sect. 4.4.2). We nevertheless present the specialized case of Clifford algebras for concreteness.

7.2. Explicit Matrix Representations of Clifford Algebras

Luckily, the representation theory of Clifford algebras is exceptionally well-understood. There is a well-known (see e.g. [Por81]) isomorphism, which presents itself particularly simple over the complex numbers. Namely, recalling that k is even with $k = 2s$,

$$Cl_k(\mathbb{C}) \cong \mathbb{C}^{2^s \times 2^s} .$$

That is, a Clifford algebra of dimension $2^k = 2^{2s}$ is indeed isomorphic to a full matrix algebra of dimension $2^s \times 2^s$. In the sequel, we construct these isomorphisms and their inverses explicitly for all even k. There are similar isomorphisms for the odd case, but since we only need the claim for even k, we shall concern ourselves only therewith.

First, note that $Cl_2(\mathbb{C})$ admits a homomorphism into the full matrix algebra $\mathbb{C}^{2\times 2}$, via

$$\mu_2 : Cl_2(\mathbb{C}) \to \mathbb{C}^{2\times 2}, \quad \mathbf{e}_1 \mapsto \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}, \mathbf{e}_2 \mapsto \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} .$$

In particular, one checks that

$$\mu(\mathbf{e}_1)^2 = \mu(\mathbf{e}_2)^2 = I_2, \ \mu(\mathbf{e}_1\mathbf{e}_2) = -\mu(\mathbf{e}_2\mathbf{e}_1) . \tag{7.1}$$

Let

$$\hat{\mu} = i\mu(\mathbf{e}_2)\mu(\mathbf{e}_1) = \begin{pmatrix} 0 & i \\ -i & 0 \end{pmatrix} ,$$

and consider the following mapping.

$$\begin{aligned} \mu_{k+2} : Cl_{k+2}(\mathbb{C}) &\to \mathbb{C}^{2^{s+1}\times 2^{s+1}}, \\ \mathbf{e}_i &\mapsto \hat{\mu} \otimes \mu_k(\mathbf{e}_i) \text{ for } i \leq k, \\ \mathbf{e}_{k+1} &\mapsto \mu_2(\mathbf{e}_1) \otimes I_{2^s}, \\ \mathbf{e}_{k+2} &\mapsto \mu_2(\mathbf{e}_2) \otimes I_{2^s} . \end{aligned}$$

Here, $A \otimes B$ is the Kronecker product of A and B, defined as a block matrix with blocks the size of B. Namely $(A \otimes B)_{ij} = a_{ij} \cdot B$. If A is of dimension $n \times n$ and B of dimension $m \times m$, then we will identify this matrix in the obvious way with the $nm \times nm$-matrix that corresponds to $A \otimes B$. The Kronecker product can be thought of as a tensor product of matrices. In particular, $(A \otimes B)(C \otimes D) = AC \otimes BD$.

Proposition 7.2.1. *The family of mappings $(\mu_i)_{i\in\mathbb{N}}$ is a family of algebra morphisms.*

Proof. We proceed with induction over s. The case of $s = 1$ was already treated above in Eq. (7.1), so let $s > 1$ and assume the claim holds for all $s' < s$, or equivalently, all even $k' < k+2$. Consider $k+2 = 2(s+1)$. Observe that $\hat{\mu}^2 = I_2$, and by induction hypothesis, $\mu_k(\mathbf{e}_i)^2 = I_{2^s}$. Recalling the 2×2-case above and using the rule for the Kronecker product, we have for all i that $\mu_{k+2}(\mathbf{e}_i)^2 = I_{2^{s+1}}$. Now, for $i \neq j$, if both $i, j \leq k$, then again by inductive hypothesis, $\mu_{k+2}(\mathbf{e}_i\mathbf{e}_j) = -\mu_{k+2}(\mathbf{e}_j\mathbf{e}_i)$. On the other hand, if $i, j \in \{k+1, k+2\}$, the claim follows from the case of $k = 2$. Note that $\hat{\mu}$ anticommutes with either one of $\mu(\mathbf{e}_1)$ and $\mu(\mathbf{e}_2)$. Therefore, if exactly one of i, j is greater than k, the relations continue to hold as well. □

To make this more concrete, we can expand the Kronecker products and write first

$$A_1 = \begin{pmatrix} 0 & i\mu_k(\mathbf{e}_j) \\ -i\mu_k(\mathbf{e}_j) & 0 \end{pmatrix}, A_2 = \begin{pmatrix} -I_{2^s} & 0 \\ 0 & I_{2^s} \end{pmatrix}, A_3 = \begin{pmatrix} 0 & -I_{2^s} \\ -I_{2^s} & 0 \end{pmatrix}.$$

Then, we have

$$\mu_{k+2}(\mathbf{e}_j) = \begin{cases} A_1 & \text{for } j \leq k, \\ A_2 & \text{for } j = k+1, \\ A_3 & \text{for } j = k+2. \end{cases} \tag{7.2}$$

Let us first work out what μ_{k+2} does to elements x that belong to the smaller Clifford algebra $Cl_k(\mathbb{C}) \subseteq Cl_{k+2}(\mathbb{C})$. To this end, we fix the following notation. For an element x of any Clifford algebra, x^+ and x^- shall denote the projection of x to the basis elements corresponding to sets of even and odd sizes, respectively.

Lemma 7.2.2. *The following holds for all $x \in Cl_k(\mathbb{C})$.*

$$\mu_{k+2}(x) = \begin{pmatrix} \mu_k(x^+) & i\mu_k(x^-) \\ -i\mu_k(x^-) & \mu_k(x^+) \end{pmatrix} . \tag{7.3}$$

Proof. Consider first a basis element $\mathbf{e}_S$ of $Cl_k(\mathbb{C})$, say $\mathbf{e}_S = \mathbf{e}_{i_1} \cdots \mathbf{e}_{i_{|S|}}$ with $i_1 < \ldots < i_{|S|}$. By definition and the properties of the Kronecker product,

$$\begin{aligned} \mu_{k+2}(\mathbf{e}_S) = \prod_{j=1}^{|S|} \mu_{k+2}(\mathbf{e}_{i_j}) = \prod_{j=1}^{|S|} \hat{\mu} \otimes \mu_k(\mathbf{e}_{i_j}) = \\ \hat{\mu}^{|S|} \otimes \left(\prod_{j=1}^{|S|} \mu_k(\mathbf{e}_{i_j}) \right) = \hat{\mu}^{|S|} \otimes \mu_k(\mathbf{e}_S) . \end{aligned} \tag{7.4}$$

Since $\hat{\mu}^2 = I_2$, the left factor of this last expression depends only on the parity of $|S|$: It is equal to I_2 for even $|S|$, and $\hat{\mu}$ for odd $|S|$.

Let now x be an arbitrary element of $Cl_k(\mathbb{C})$. Of course, $x = x^+ + x^-$, and Eq. (7.4) together with bilinearity of the Kronecker product then yield:

$$\begin{aligned} \mu_{k+2}(x) =& \mu_{k+2}(x^+) + \mu_{k+2}(x^-) \\ =& I_2 \otimes \mu_k(x^+) + \hat{\mu} \otimes \mu_k(x^-) \\ =& \begin{pmatrix} \mu_k(x^+) & 0 \\ 0 & \mu_k(x^+) \end{pmatrix} + \begin{pmatrix} 0 & i\mu_k(x^-) \\ -i\mu_k(x^-) & 0 \end{pmatrix} . \end{aligned}$$

Summing up the last line into one matrix yields the expression from the statement of the lemma. □

We can now turn to considering a general element

$$x = \sum_{S \subseteq \{1,\ldots,k+2\}} \alpha_S \mathbf{e}_S \in Cl_{k+2}(\mathbb{C})\,.$$

Recall that we assume that the coefficients α_S are chosen with signs according to the assumption that the basis element corresponding to a set S is the product of the basis elements corresponding to its elements in ascending order.

We can decompose any such element uniquely into four parts, according to which elements from $\{k+1, k+2\}$ appear in S:

$$x_< = \sum_{T \subseteq \{1,\ldots,k\}} \alpha_T \mathbf{e}_T, \qquad \textit{i.e.}, \text{ neither } k+1 \text{ nor } k+2 \text{ appear in } T,$$

$$x_1 = \sum_{\{k+1\} \subseteq U \subseteq \{1,\ldots,k+1\}} \alpha_U \mathbf{e}_{U-\{k+1\}}, \qquad \textit{i.e.}, \text{ only } k+1 \text{ appears in } U,$$

$$x_2 = \sum_{\{k+2\} \subseteq V \subseteq \{1,\ldots,k,k+2\}} \alpha_V \mathbf{e}_{V-\{k+2\}}, \qquad \textit{i.e.}, \text{ only } k+2 \text{ appears in } V,$$

$$x_{12} = \sum_{\{k+1,k+2\} \subseteq W \subseteq \{1,\ldots,k+2\}} \alpha_W \mathbf{e}_{W-\{k+1,k+2\}} \qquad \textit{i.e.}, \text{ both } k+1 \text{ and } k+2 \text{ appear in } W.$$

Observe that $x_<, x_1, x_2, x_{12} \in Cl_k(\mathbb{C})$. The indices of the basis elements $\mathbf{e}_S$ in the summation and the signs of the coefficients α_S were chosen in such a way that the following holds:

$$x = x_< + x_1\mathbf{e}_{k+1} + x_2\mathbf{e}_{k+2} + x_{12}\mathbf{e}_{k+1}\mathbf{e}_{k+2}\,.$$

For the sake of readability, we shall define $\mu_k(M)$ to mean an entrywise application of μ_k to the matrix entries of M, and we will write down the entries of M without parentheses. Since μ_k is an algebra homomorphism, this is compatible with matrix and scalar multiplication. With the notation in place, we can state:

Proposition 7.2.3. *The following holds for all $x \in Cl_k(\mathbb{C})$.*

$$\mu_{k+2}(x) = \\ \mu_k \begin{pmatrix} x_<^+ - x_1^+ + i(x_{12}^- - x_2^-) & -x_2^+ - x_{12}^+ + i(x_<^- + x_1^+) \\ x_{12}^+ - x_2^+ + i(x_1^- - x_<^-) & x_1^+ + x_<^+ + i(x_2^- + x_{12}^-) \end{pmatrix} . \tag{7.5}$$

Proof. On x, the mapping μ_{k+2} has the following effect.

$$\begin{aligned} \mu_{k+2}(x) = \quad & \mu_{k+2}(x_<) \\ & + \mu_{k+2}(x_1)\mu_{k+2}(\mathbf{e}_{k+1}) \\ & + \mu_{k+2}(x_2)\mu_{k+2}(\mathbf{e}_{k+2}) \\ & + \mu_{k+2}(x_{12})\mu_{k+2}(\mathbf{e}_{k+1})\mu_{k+2}(\mathbf{e}_{k+2}) \,. \end{aligned}$$

Plugging in the matrix expressions from (7.2), we obtain:

$$\begin{aligned} \mu_{k+2}(x) = \quad & \mu_{k+2}(x_<) \\ & + \mu_{k+2}(x_1) \cdot \begin{pmatrix} -I_{2^s} & 0 \\ 0 & I_{2^s} \end{pmatrix} \\ & + \mu_{k+2}(x_2) \cdot \begin{pmatrix} 0 & -I_{2^s} \\ -I_{2^s} & 0 \end{pmatrix} \\ & + \mu_{k+2}(x_{12}) \cdot \begin{pmatrix} 0 & -I_{2^s} \\ I_{2^s} & 0 \end{pmatrix} . \end{aligned}$$

As already noted, $x_<, x_1, x_2, x_{12} \in Cl_k(\mathbb{C})$, and Eq. (7.3) gives

$$\begin{aligned} \mu_{k+2}(x) = \quad & \mu_k \begin{pmatrix} x_<^+ & ix_<^- \\ -ix_<^- & x_<^+ \end{pmatrix} \\ & + \mu_k \begin{pmatrix} x_1^+ & ix_1^- \\ -ix_1^- & x_1^+ \end{pmatrix} \cdot \begin{pmatrix} -I_{2^s} & 0 \\ 0 & I_{2^s} \end{pmatrix} \\ & + \mu_k \begin{pmatrix} x_2^+ & ix_2^- \\ -ix_2^- & x_2^+ \end{pmatrix} \cdot \begin{pmatrix} 0 & -I_{2^s} \\ -I_{2^s} & 0 \end{pmatrix} \\ & + \mu_k \begin{pmatrix} x_{12}^+ & ix_{12}^- \\ -ix_{12}^- & x_{12}^+ \end{pmatrix} \cdot \begin{pmatrix} 0 & -I_{2^s} \\ I_{2^s} & 0 \end{pmatrix} . \end{aligned}$$

Multipliying this out results in the following identity:

$$\begin{aligned}\mu_{k+2}(x) = \quad & \mu_k \begin{pmatrix} x^+_< & ix^-_< \\ -ix^-_< & x^+_< \end{pmatrix} \\ & + \mu_k \begin{pmatrix} -x^+_1 & ix^-_1 \\ ix^-_1 & x^+_1 \end{pmatrix} \\ & + \mu_k \begin{pmatrix} -ix^-_2 & -x^+_2 \\ -x^+_2 & ix^-_2 \end{pmatrix} \\ & + \mu_k \begin{pmatrix} ix^-_{12} & -x^+_{12} \\ x^+_{12} & ix^-_{12} \end{pmatrix} .\end{aligned}$$

Collecting the entries then yields the recursive expression for μ_{k+2} from the statement of the lemma. □

Let us turn to the inverse map of μ_{k+2}. Given the established abundance of sub- and superscripts containing minus signs and ones, we shall, with some typographical waggishness, simply denote the inverse of μ with an h, *i.e.*, $\mu^{-1}_{k+2} = h_{k+2}$. *A priori*, it is not clear that an inverse h_{k+2} even exists. We shall deal with this first, and do so by constructing explicitly the mapping h_{k+2}, which is helpful for our computational ends.

Proposition 7.2.4. *For all k, μ_k is an isomorphism.*

Proof. We proceed inductively. For the sake of explicitness, let show the case $s = 1$. For any complex 2×2-matrix

$$Y = \begin{pmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{pmatrix} ,$$

we consider the linear system

$$Y = \lambda_1 \mu_2(\mathbf{e}_1) + \lambda_2 \mu_2(\mathbf{e}_2) + \lambda_{12} \mu_2(\mathbf{e}_1)\mu_2(\mathbf{e}_2) + \lambda_0 \mu_2(1) ,$$

which, after plugging in the represenations μ_2, boils down to

$$\begin{pmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{pmatrix} = \begin{pmatrix} -\lambda_2 + \lambda_0 & -\lambda_1 - \lambda_{12} \\ -\lambda_1 + \lambda_{12} & \lambda_{12} + \lambda_0 \end{pmatrix} .$$

This system admits the unique solution

$$\lambda_1 = -\frac{y_{21} + y_{12}}{2},\ \lambda_2 = \frac{y_{22} - y_{11}}{2},\ \lambda_{12} = \frac{y_{21} - y_{12}}{2},\ \lambda_0 = \frac{y_{11} + y_{22}}{2} ,$$

and shows that indeed, μ_2 is an isomorphism of algebras.

Now, assume the claim holds true for $s > 1$, and consider some 2×2 block matrix $Y \in \mathbb{C}^{2^{s+1} \times 2^{s+1}}$, say

$$Y = \begin{pmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{pmatrix} .$$

We want to compute $h_{k+2}(Y)$, that is, construct the unique $x \in Cl_{k+2}(\mathbb{C})$ with $\mu_{k+2}(x) = Y$. Of course, any such x will decompose into $x_<, x_1, x_2$ and x_{12} as before, and the recursion (7.5) then implies that

$$\begin{aligned} Y_{11} =& \mu_k \left(x_<^+ - x_1^+ + i(x_{12}^- - x_2^-) \right) \\ Y_{12} =& \mu_k \left(-x_2^+ - x_{12}^+ + i(x_<^- + x_1^+) \right) \\ Y_{21} =& \mu_k \left(x_{12}^+ - x_2^+ + i(x_1^- - x_<^-) \right) \\ Y_{22} =& \mu_k \left(x_1^+ + x_<^+ + i(x_2^- + x_{12}^-) \right) . \end{aligned}$$

Let $h_{ij} = h_k(Y_{ij})$, which exists by inductive hypothesis. With the aid of a linear algebra system, or nerves of steel, one then checks that

$$x_< = -ih_{12}^- + \frac{1}{2}(h_{11}^+ + h_{22}^+) \tag{7.6}$$

$$x_1 = -i(h_{12}^- + h_{21}^-) + \frac{1}{2}(h_{22}^+ - h_{11}^+) \tag{7.7}$$

$$x_2 = \frac{i}{2}(h_{11}^- - h_{22}^-) - \frac{i}{4}h_{11}^+ - \frac{1}{2}(h_{12}^+ + h_{21}^+) + \frac{i}{4}h_{22}^+ \tag{7.8}$$

$$x_{12} = -\frac{i}{2}(h_{11}^- + h_{22}^-) + \frac{i}{4}(h_{22}^+ - h_{11}^+) + \frac{1}{2}(h_{21}^+ - h_{12}^+) , \tag{7.9}$$

and this solution is unique. Therefore, μ_{k+2} is an isomorphism as

well. □

7.3. Fast Computation of Representations and Inverses

We now put to use computationally the identities proved in the previous subsection:

Proposition 7.3.1. *Given an element $x \in Cl_k(\mathbb{C})$ as a list of basis coefficients, the matrix $\mu_k(x)$ can be computed with $2^k \cdot \mathrm{poly}(k)$ arithmetic operations over $\mathbb{C}$.*

Additionally, if the size of coefficients of x is bounded by τ, then $\mu_k(x)$ can be computed in $2^k \cdot \mathrm{poly}(\tau)$ bit operations.

Proof. First, compute the four parts $\mu_k(x_<), \mu_k(x_1), \mu_k(x_2)$ and $\mu_k(x_{12})$ recursively. From Eq. (7.3), the resulting 2×2 block matrix in each case will have the projection to the even part in the top left and bottom right entries, the odd part, up to a scaling by $\pm i$ in the bottom left and upper right corner. Additions and scalar multiplications of the resulting matrices can be performed in 2^k arithmetic operations over $\mathbb{C}$, and there are sixteen of those to perform when computing μ_{k+2} from μ_k. Therefore, the amount $T_\mu(k+2)$ of arithmetic operations that have to be carried out to compute μ_{k+2} is $T_\mu(k+2) = 4 \cdot T(k) + 16 \cdot 2^k$ and therefore $T_\mu(k) \leq 2^k \cdot \mathrm{poly}(k)$. Clearly, the numbers occurring when computing μ_k stay of polynomial size. □

Proposition 7.3.2. *Given a matrix $Y \in \mathbb{C}^{2^s \times 2^s}$, the element $h_k(Y) \in Cl_k(\mathbb{C})$ as a list of basis coefficients can be computed with $2^k \cdot \mathrm{poly}(k)$ arithmetic operations over $\mathbb{C}$.*

Additionally, if the size of coefficients of x is bounded by τ, then $h_k(x)$ can be computed in $2^k \cdot \tau$ bit operations.

Proof. We first compute the four smaller subproblems $h_k(Y_{ij})$, and with that, obtain also the $h^{\pm}_{ij}$. Then, we reconstruct $x_<, x_1, x_2, x_{12}$ according to Equations (7.6) through (7.9) using a constant number

of linear combinations (*i.e.*, sums of scalar multiples) of the computed $h_k(Y_{ij})^{\pm}$. Each such operation costs at most 2^k arithmetic operations over $\mathbb{C}$, via Proposition 7.1.1. All that is then left to do is to compute $x = x_< + x_1\mathbf{e}_{k+1} + x_2\mathbf{e}_{k+2} + x_{12}\mathbf{e}_{k+1}\mathbf{e}_{k+2}$. Again, these are four additions, costing 2^k operations each, and multiplication with basis elements $\mathbf{e}_i$ corresponds to a basis shift, which also takes 2^k bit operations (and actually, no arithmetic operations). Therefore, the amount of arithmetic operations $T_h(k+2)$ it takes to compute $h_{k+2}(Y)$ is $T_h(k+2) \leq 4 \cdot T_h(k) + C \cdot 2^k$ and thus $T_h(k) \leq 2^k \cdot \mathrm{poly}(k)$ for some constant $C > 0$. Again, it is clear that the numbers occurring when computing h_k stay of polynomial size □

7.4. General Arithmetic in Clifford Algebras

The preceding arguments culminate in the following statement, which was shown for real clifford algebras by Leopardi in [Leo+05]. In fact, Włodarczyk relies on [Leo+05] to obtain his result.

Theorem 7.4.1. *Given two elements $x, y \in Cl_k(\mathbb{C})$ as a list of basis coefficients, their product $xy \in Cl_k(\mathbb{C})$ as a list of basis coefficients can be computed using $2^{\omega k/2}\,\mathrm{poly}(k)$ arithmetic operations over $\mathbb{C}$.*

Additionally, if the size of coefficients of x and y are bounded by τ, then their product can be computed in $2^{\omega k/2} \cdot \mathrm{poly}(\tau)$ bit operations.

Proof. The algorithm first computes the two matrices $X = \mu_k(x)$ and $Y = \mu_k(y)$. Then, we use fast matrix multiplication to obtain $Z = X \cdot Y$. Finally, we compute $h_k(Z)$. By Lemma 7.2.4, this h_k exists, and by Lemma 7.2.1, the algorithm is correct. performing two applications of μ_k, one matrix multiplication, and one application of h_k.

By Propositions 7.3.1 and 7.3.2, computing X, Y and Z can each be performed well within the required time bound. Furthermore, fast matrix multiplication takes time $2^{\omega s} \cdot \mathrm{poly}(n)$, and $2^{\omega s}$ arithmetic operations, on matrices of size $2^s = 2^{k/2}$, which is also within the required time bound. This proves the theorem. □

The fact that the embedding of a Clifford algebra into a full matrix algebra and its inverse can be computed in time linear in their dimension proves in particular the following trivial, but highly important, connection to matrix multiplication.

Proposition 7.4.2. *Two matrices of dimension $n \times n$ can be multiplied in time n^c for some constant c, if and only if two general elements of a Clifford algebra of dimension n can be multiplied in time n^c.*

7.5. General Arithmetic in the Exterior Algebra

We now turn to applying the computational insights gained on Clifford algebras to implement fast arithmetic in the exterior algebra. First, let us elaborate on the kind of connection that exists between Clifford algebras and exterior algebras.

To this end, recall that an algebra is called *graded* if it admits a direct sum composition $A = \bigoplus_{i \in \mathbb{N}} A_i$ as a vector space, and in such a way that $A_i \cdot A_j \subseteq A_{i+j}$. It is easy to check that the exterior algebra admits such a decomposition by degree, *i.e.*, the number of generators needed to express the linear basis element $\mathbf{e}_S$ as their product (and this number is just $|S|$). However, in the Clifford algebra, the degree-one element $(\mathbf{e}_1 + \mathbf{e}_2)$ squares not to a degree-two element, as would be required for a grading, but, being $2 + 2\mathbf{e}_1\mathbf{e}_2$, instead contains also the element 2 of degree zero.

A weaker condition is being *filtered*: Here, we demand not that the sum be direct, *i.e.*, there is a sequence $F_{-1} = \{0\} = F_1 \subseteq F_2 \subseteq \ldots \subseteq A$ of linear subspaces of A such that $\bigcup_{i \in \mathbb{N}} F_i = A$ holds, and again multiplication is compatible with this, *i.e.*, $F_i \cdot F_j \subseteq F_{i+j}$. Of course, all graded algebras are filtered by letting $F_i = \bigcup_{j=0}^{i} A_j$, but for example, while $Cl_k(\mathbb{C})$ is not graded, it sure is filtered, again letting F_i be the set of elements of $Cl_k(\mathbb{C})$ that are composed of products of at most i generators.

Given some filtered algebra A with filtered parts F_i, we can obtain

from it its so-called *associated graded algebra* $\mathrm{gr}(A)$ as follows. Let $Q_i = F_i/F_{i-1}$ for all $i \geq 0$ be the linear quotient space, which means Q_i contains all those elements that are products of *exactly* i generators, but not less. Per definition, the sum $\bigoplus_{i\in\mathbb{N}} Q_i$ is direct, and we define $\mathrm{gr}(A)$ to be this sum: $\mathrm{gr}(A) = \bigoplus_{i\in\mathbb{N}} Q_i$. Let $x \in Q_i, y \in Q_j$, *i.e.*, $x = a + F_{i-1}, y = b + F_{j-1}$ for some $a \in F_i, b \in F_j$. Then, $xy = (a + F_{i-1})(b + F_{j-1}) = ab + aF_{j-1} + bF_{i-1} + F_{i-1}F_{j-1}$. Of course, $ab \in F_{i+j}$, and $aF_{j-1}, bF_{i-1}, F_{i+j-2} \subseteq F_{i+j-1}$, such that xy is a subset of $ab+F_{i+j-1} \in Q_{i+j}$. Intuitively speaking, this corresponds to performing the multiplication of x and y in $\mathrm{gr}(A)$ separately for each homogeneous component of x and y of degree i and j, respectively, and then forgetting those parts of the result that dropped in degree.

It should become clear after a moment's thought (and is a generally known fact), that the following holds.

Proposition 7.5.1. *The algebras* $\mathrm{gr}(Cl_k(\mathbb{C}))$ *and* $\Lambda(\mathbb{C}^k)$ *are isomorphic.*

In order to multiply two elements u and v in the exterior algebra $\Lambda(\mathbb{C}^k)$, we first consider the decomposition of u and v into homogeneous parts, *i.e.*, $u = u_0+u_1+\cdots+u_k$ and $v = v_0+v_1+\cdots+v_k$, where u_i and v_i are of degree i, and thus $uv = \sum_{i,j=0}^{k} u_i v_j$. Let $\iota : \Lambda(\mathbb{C}^k) \to Cl_k(\mathbb{C})$ be the natural *linear* isomorphism between $Cl_k(\mathbb{C})$ and $\Lambda(\mathbb{C}^k)$, which satisfies $\iota(\Lambda(\mathbb{C}^k)_i) \subseteq F_i$. Let $p_{ij} = \iota(u_i)\iota(v_j)$. By definition, $p_{ij} \in F_{i+j}$. We can thus consider the image $\eta(p_{ij})$ of p_{ij} under the canonical projection η of F_{i+j} to Q_{i+j}, which just corresponds to forgetting all low-degree coefficients. Let $\kappa : Q_i \to \Lambda(\mathbb{C}^k)_i$ be the canonical isomorphism. We then have:

$$uv = \sum_{i,j=0}^{k} \kappa(\eta(\iota(u_i)\iota(v_j))) \,. \tag{7.10}$$

Włodarczyk [Wło16] formulated this as an observation in a different language, and termed the phenomenon—or rather, its algorithmic exploitation—*size-grouping.*

We can thus re-derive Włodarczyk's result:

Theorem 7.5.2 ([Wło16])**.** *Given two elements $x, y \in \Lambda(\mathbb{C}^k)$ as a list of basis coefficients, their product $x \wedge y \in \Lambda(\mathbb{C}^k)$ as a list of basis coefficients can be computed using $2^{\omega k/2} \cdot \mathrm{poly}(k)$ arithmetic operations over $\mathbb{C}$.*

Additionally, if the size of coefficients of x and y is bounded by 2^τ, then their product can be computed in $2^{\omega k/2} \cdot \mathrm{poly}(\tau)$ bit operations.

Proof. The maps ι and κ do not actually have to be computed, and there is not much to compute in any case. Their application just indicates that u and v should be understood as elements of a different algebra.[1] Computing the $(k+1)^2$ products p_{ij} is doable within the required bounds by Theorem 7.4.1. The map η can be computed in time $\dim \Lambda(\mathbb{C}^k) = 2^k$, since, as mentioned, applying η is only a formal way of saying to forget all those coefficients that resulted from a degree drop when multiplying in $Cl_k(\mathbb{C})$. Then by Proposition 7.1.1, computing the final sum in Eq. (7.10) then takes another $(k+1) \cdot 2^k$ arithmetic operations, and at most a $\mathrm{poly}(n)$ factor additional bit operations. □

Remark 7.5.3. Observe that reducing multiplication in exterior algebras to multiplication in Clifford algebras is, in a way, mathematically perverse: Clifford algebras are much more general objects than exterior algebras, and the latter arise as a degeneration of the former. It seems wasteful to compute the product in exterior algebras in this fashion, and yet, it is not at all clear how to perform multiplication in exterior algebras faster than by this reduction. Failing this, we could of course try to reduce the exterior product to the Clifford product, and therefore, in particular in light of Proposition 7.4.2, prove a—possibly surprising—lower bound on the complexity of multiplication in exterior algebras.

We will discuss this phenomenon a little more in the next section, when we relate the problems of this section to the far more well-studied ones of computing subset convolutions.

[1] We could also have defined $Cl_k(\mathbb{C})$ and $\Lambda(\mathbb{C}^k)$ to be identical as sets, but with two different notations for multiplication, say, $\wedge$ and $\vee$.

7.6. Fast Subset Convolution

In this section, we give a quick exposition of the results of Björklund *et al.* on the so-called *fast subset convolution* [Bjö+08]. We will briefly state the problem in its original formulation and then re-derive the solution of Björklund *et al.*, highlighting the conceptual connections to the solution for the fast multiplication in Clifford algebras.

Let us start with the problem definition. Given two functions $f, g : 2^{\{1,\dots,k\}} \to \mathbb{C}$ on the power set of $\{1,\dots,k\}$, define their *subset convolution* as

$$f * g : 2^{\{1,\dots,k\}} \to \mathbb{C}, S \mapsto \sum_{U \dot\cup V = S} f(U) \cdot g(V)\,.$$

We reconcile this combinatorial expression with our previous algebraic perspective as follows: We will, yet again, define an algebra over the basis $\mathbf{e}_1,\dots,\mathbf{e}_k$ of the space $\mathbb{C}^k$. This time, however, we demand only that $\mathbf{e}_i^2 = 0$ hold true for all i, and we impose no anticommutativity constraint, but instead a commutativity constraint, *i.e.*, $\mathbf{e}_i\mathbf{e}_j = \mathbf{e}_j\mathbf{e}_i$. We have seen this algebra before, namely in form of the Zeons $Z(\mathbb{C}^k)$ of Sect. 4.4.4, and we will keep with this notation also here. Observe that now, in contrast to the exterior algebra case, squares of vectors, *i.e.*, degree-one elements, do not vanish anymore. For instance, $(\mathbf{e}_i + \mathbf{e}_j)^2 = 2\mathbf{e}_i\mathbf{e}_j$, which is non-zero whenever $i \neq j$.

Recall that $Z(\mathbb{C}^k)$ still has a linear basis of cardinality 2^k, and again, we may index the basis elements using subsets of $\{1,\dots,k\}$ as $\mathbf{e}_S = \prod_{s\in S} \mathbf{e}_s$ for $S \subseteq \{1,\dots,k\}$. Given any two elements $x, y \in Z(\mathbb{C}^k)$, we can write them consequently as $x = \sum_{S\subseteq\{1,\dots,k\}} \alpha_S \mathbf{e}_S, y = \sum_{S\subseteq\{1,\dots,k\}} \beta_S \mathbf{e}_S$, and these representations are unique. These elements correspond in the obvious way to functions $f, g : 2^{\{1,\dots,k\}} \to \mathbb{C}$, taking each S to the coefficient of $\mathbf{e}_S$ in x and y, respectively. Let $z = xy = \sum_{S\subseteq\{1,\dots,k\}} \gamma_S \mathbf{e}_S$ be their product in $Z(\mathbb{C}^k)$. Then, it is easy to check that the function $h : 2^{\{1,\dots,k\}} \to \mathbb{C}$ that corresponds to z is precisely given as $h = f * g$. Therefore, computing the subset convolutions is equivalent to computing general products in $Z(\mathbb{C}^k)$, and in fact, this was remarked already in the proof of Lemma 4.4.4 where we in turn referred

to fast subset convolution for implementing arithmetic in $Z(\mathbb{C}^k)$.

We give an exposition of how this goes, analogous to the computation in $Cl_k(\mathbb{C})$ and $\Lambda(\mathbb{C}^k)$, to which we will refer to as the *signed case*, whereas the operations in this section constitute the *unsigned case*. The algebra $Z(\mathbb{C}^k)$ will play the rôle of $\Lambda(C^k)$, and the natural analogon of $Cl_k(\mathbb{C})$ is defined again as a commutative variant of $Cl_k(\mathbb{C})$, where we still demand $\mathbf{e}_i^2 = 1$, but also $\mathbf{e}_i\mathbf{e}_j = \mathbf{e}_j\mathbf{e}_i$ for all i, j. It is easy to see that this algebra is isomorphic to $\mathbb{C}[\mathbb{Z}_k^2]$, the complex group algebra of $\mathbb{Z}_2^k$, and we shall not invent a name for this algebra, and will denote it only in this way.

As in the signed case, we can observe that $Z(\mathbb{C}^k)$ is naturally a graded algebra, and that $\mathbb{C}[\mathbb{Z}_k^2]$ is not graded, but filtered by degree. The degree of a group element corresponds to the number of one-entries in it, as an element of $\mathbb{Z}_2^k$. Similarly, we have a corresponding statement to Proposition 7.5.1.

Proposition 7.6.1. *The algebras $Z(\mathbb{C}^k)$ and $\operatorname{gr}(\mathbb{C}[\mathbb{Z}_2^k])$ are isomorphic.*

This makes it possible to apply the exact same reduction in the unsigned case that was used to reduce computation in the exterior algebra to computation in the Clifford algebra, and indeed, it is then obvious that we can implement one multiplication in $Z(\mathbb{C}^k)$ using $O(k^2)$ multiplications and additions in $\mathbb{C}[\mathbb{Z}_2^k]$.

Thankfully, the computation in group algebras—particularly in commutative ones—is well-understood, and can be realized by multidimensional fast Fourier transforms. In essence, this is fast multipoint evaluation and interpolation at the multiplicative hypercube, *i.e.*, $\{-1, 1\}^k \cong \mathbb{Z}_2^k$, and this can be performed in time $2^k \cdot \operatorname{poly}(k)$.

These arguments are a sketch to prove:

Theorem 7.6.2 ([Bjö+08, Theorem 2])**.** *Let $f, g : 2^{\{1,\dots,k\}} \to \mathbb{C}$ be two complex-valued functions that take values that can be represented using τ bits. Then, the subset convolution $f * g : 2^{\{1,\dots,k\}}$ (as a table of 2^k function values) can be computed in $2^k \cdot \operatorname{poly}(k, \tau)$ bit operations.*

Remark 7.6.3. Let us put this into perspective: Computation of $f * g$ is multiplication in $Z(\mathbb{C}^k)$, which is an unsigned analogon of $\Lambda(\mathbb{C}^k)$.

The fact that this can be performed in quasilinear time in the dimension might give us hope that we can somehow adapt the approach to the signed case, in order to obtain quasilinear running time also there.

Unfortunately, this is *already* what we do: Also in the unsigned case, the fastest known way to compute a product is by reducing to multiplication in a filtered algebra, which actually needs to take into account many more subproducts than appear the actual graded algebra we're interested in. In this sense, the mathematical perversion of the signed case, referred to in Remark 7.5.3, appears also in the unsigned case. The only thing that saves us is that, in the unsigned case, multiplication in the filtered algebra can be performed fast enough.

This observation, in combination with Proposition 7.4.2, make the case for $2^{\omega k/2}$ being the correct basis for multiplication in $\Lambda(\mathbb{C}^k)$ more plausible.

8. Algebraic Barriers for Extensor-Coding

Broadly speaking, when aiming towards deterministic decision algorithms, as in Chap. 5, Extensor-Coding essentially works by evaluating a multivariate polynomial associated with the input over an exterior algebra. This algebra is of dimension 4^k, where k will typically be the parameter of the input instance (while it formally is half the dimension of the underlying vector space).

In this way, it is similar to a method introduced by Koutis [Kou08]. It differs, however, in the points at which the polynomial associated with the input instance is evaluated. While Koutis, and later, Williams [Wil09], rely on random evaluation points, in [BDH18], certain carefully constructed vectors are used. One can readily observe that "evaluating a polynomial" involves, on an arithmetic level, nothing but multiplications and additions. In particular, if one evaluates a polynomial over any algebraic structure that is closed under multiplication and addition, one will always obtain again an element of this algebraic structure after evaluation. Turning this around, one can always restrict one's attention to the closure of (*i.e.*, the substructure generated by) the set of evaluation points. In particular, it might be far easier to actually implement the arithmetic operations only over this substructure than over the entire structure.

In this chapter, we will examine the subalgebra generated by the above mentioned special evaluation points (*i.e.*, the smallest set closed under addition, scaling with a field constant, and multiplication containing all the evaluation points). Note now that the dimension of an algebra provides a trivial lower bound on the cost of general computation in it,

simply because one has to write down its elements' coordinates at some point. Conversely, the dimension itself can be used to derive a trivial upper bound for the cost of computation as well, just by a look-up of the structural constants of the algebra, but this bound is generally far from optimal. For example, naive multiplication of degree-d polynomials (which can be modeled as objects of a $O(d)$-dimensional algebra over some field) takes around $O(d^2)$ field operations, while (over compatible fields) the Fast Fourier Transform method yields the classic bound of $O(d \log d)$. A running time of the latter form, *i.e.*, quasilinear in the dimension of the algebra, is of course the best one can hope for.

We prove that, surprisingly (for reasons that will be expanded on later), the dimension of the subalgebra generated by the evaluation points used in Extensor-Coding is of dimension *exponentially smaller* than 4^k, *i.e.*, the dimension of the full algebra. At first, this seems to open up a tantalizing new point of attack on one of the most prominent open problems in the area of parameterized algorithms: To exhibit a *deterministic* algorithm for the k-path problem that matches the running time of $2^k \cdot \mathrm{poly}(n)$ for the best *randomized* algorithms [Wil09; Kou08]. Now, *a priori*, we could hope for the studied subalgebra to be of dimension 2^k (but not much smaller, by a result of Koutis-Williams [KW16]). Then, a quasilinear multiplication algorithm would give a bound of $2^k \cdot \mathrm{poly}(k)$ field operations, and (assuming all coefficients stay of moderate size) hence produce a deterministic algorithm solving the problem in time $2^k \cdot \mathrm{poly}(n)$.

Unfortunately, we share the fate of Tantalos:[1] The low-hanging fruit moves out of reach with our result.

8.1. The Barrier

Let us first elaborate more formally on the fundamental insight that motivates our result: Let f be a complex polynomial, such as the

[1] *Tantalizing* is often liberally used quite synonymously to *tempting*, *alluring*, sometimes even *exciting* or *electrifying*. With this rather extravagant piece of prose, we wish to draw attention to the fact that there is more meaning in this word, owing to its mythological origin, than is commonly attached to it.

polynomial computed by a circuit in, say, Theorem 5.2.5. During the evaluation of this polynomial f (or rather the circuit computing it) at points from the image of $\overline{\phi}$, only sums and wedge products of elements of the image of $\overline{\phi}$ are ever formed.

We can say this rigorously and concisely by stating that during such an evaluation, all computation may only occur in the *subalgebra of* $\Lambda(V \oplus V)$ *generated by the image of* $\overline{\phi}$, which is—by definition—the set of all sum-wedge product combinations of the generating set. It is precisely the subalgebra generated by these elements $\overline{\phi}(c)$ that we will study. The fundamental quantity associated with this subalgebra is its dimension, that is, the dimension of the subalgebra as a complex vector space. Let us stress again that any potential progress on the technique in its present form hinges on the dimension of this subalgebra: It provides both strict lower bounds and a good guide towards the upper bounds one can hope for when solving problems using Extensor-Coding. We will also point out possible points of attack to circumvent the limitations by modifying the technique.

A Family of Related Subalgebras. The decisive property that makes Vandermonde codings so useful is that any distinct k of them are linearly independent, which is commonly referred to the set of Vandermonde vectors being in general position. This, however, is not only a property enjoyed by Vandermonde vectors. First of all, any finite random set of vectors is in general position almost certainly.

This suggests that instead of considering the subalgebra generated by Vandermonde codings, one might as well just take any n random vectors and proceed with them. Extensive computational experiments have shown, however, that the expected dimension of this subalgebra is almost as high as the dimension of the full algebra. Curiously, it seems to be equal (and in fact this was the only case that ever occurred during all our experiments) to the k-th Catalan number, which grows asymptotically faster than $4^{(1-\varepsilon)k}$ for all $\varepsilon > 0$.

It is worth pondering about this phenomenon for a little while. From a matroid perspective, the Vandermonde codings just correspond to

a representation of the k-uniform matroid over an n-element universe. One might now hope that the dimension of the subalgebra generated by the lifts of the columns of this representation matrix is a matroid invariant; however, this is *not* the case. Even more, most representations are just as bad as the worst case.

Remarkably, the extraordinarily well-behaved case of the Vandermonde representation transfers quite directly to a related family of subalgebras. Consider any set $P = \{p_1, \ldots, p_k\}$ of formal univariate polynomials that are linearly independent and of degree less than k. In other words, a basis of the set of polynomials of degree less than k. We can form, for any n, a $k \times n$ evaluation matrix of this set P, where the i-th row is given as $(p_i(1), p_i(2), \ldots, p_i(n))$. If we pick as P the standard monomial basis, this is just the Vandermonde representation. However, we may as well pick any other basis, and the resulting matrix will again have the property that any subset of k columns is linearly independent. This works, provided the p_i do not have a common root at the evaluation point. We can take care of this easily by just picking a different, appropriate set of evaluation points. Note that this set exists (and in fact, almost certainly, any random set will do) by the fact that the p_i are distinct polynomials. Now, we can again study the subalgebra of $\Lambda(V \oplus V)$ generated by the lifts of the column vectors of this evaluation matrix. Surprisingly, the argument for the upper dimension bound carries over immediately, but we could not find a corresponding proof for the lower bound. This opens up the exciting possibility of finding lower-dimensional algebras in this family, that could lead to even faster algorithms. Note that despite the main motivation for studying this algebra may stem from the flagship k-path problem, due to the connection to multilinear detection, finding these algorithms has impacts for all the problems that reduce to this special case of multilinear detection, including those studied in this work. Let us state our result formally.

Theorem 8.1.1. *Let $V = \mathbb{C}^k$ and let F_{2k+1} be the $(2k+1)$th Fibonacci number. The subalgebra of $\Lambda(V \oplus V)$ generated by the image $\{\overline{\phi}(c) \mid c \in \mathbb{C}\}$ of $\overline{\phi}$ is of dimension exactly equal to F_{2k+1}.*

Furthermore, for any linear basis P of the univariate polynomials of degree at most k, denote for $c \in \mathbb{C}$ with $P(c)$ the column vector obtained by evaluating all polynomials in P at c. Then, the subalgebra generated by $\{\overline{P(c)} \mid c \in \mathbb{C}\}$ is of dimension at most F_{2k+1}.

The rest of this chapter is devoted to a proof of this theorem. First, we will recall some additional algebraic notions.

Tensor Products and Ideals

For two algebras A and A', the *tensor product* $A \otimes A'$ of A and A' is the vector space $A \otimes A'$ made into a ring by component-wise multiplication, *i.e.*, $(a \otimes b) \cdot (a' \otimes b') = (aa') \otimes (bb')$ for all $a, a' \in A, b, b' \in A'$. An *ideal* I in an algebra is a linear subspace such that $ax \in I$ for all $x \in I$. For an ideal I in A, the quotient space A/I is again an algebra with multiplication $(a + I)(b + I) = ab + I$. Given some subset S of an algebra A, we denote the subalgebra or ideal S generates using round or angular brackets, respectively. That is, we write (S) or $\langle S \rangle$ for the smallest algebra or ideal, respectively, contained in A that contains S.

Graded Algebras

Recall that an algebra A is *graded* if, as a vector space, $A = \bigoplus_{i \in \mathbb{N}} A_i$ and for $i, j \in \mathbb{N}$ it holds that $A_i \cdot A_j \subseteq A_{i+j}$, where $A_i \cdot A_j$ is the set of all products $a \cdot b$ with $a \in A_i$ and $b \in A_j$. In this setting, the *Hilbert function* h_A of A is given by $h_A : \mathbb{N} \to \mathbb{N},\ d \mapsto \dim A_d$. An element contained in any A_i is *homogeneous*, and a *homogeneous* ideal is one that is generated by homogeneous elements. In this situation, A/I is again a graded algebra, with $(A/I)_d = A_d/I_d$. A homomorphism of algebras is *graded* if it respects the grading, *i.e.*, the image of the degree-d part is of degree d. The algebra $\mathbb{C}[x_1, \ldots, x_n]$ is the canonical example for a graded algebra, with the degree of the indeterminates being 1.

8.1.1. The Protagonist Algebras

Let $k > 2$ be a natural number. We write $R = \mathbb{C}[t_2, \ldots, t_{2k}]$ for the ring of complex polynomials in $2k-1$ variables, where we set $t_i = 0$ whenever $i < 2$ or $i > 2k$.

The central object of study we concern ourselves with in this section is not directly the algebra $\Lambda(V \oplus V)$ and its subalgebras, but instead the tensor square $\Lambda(V) \otimes \Lambda(V)$. To unclutter our notation, we define

$$G = \Lambda(V) \otimes \Lambda(V)\,.$$

We remark that the elements of $\Lambda(V)^{\otimes 2}$ are, strictly speaking, not extensors, but there is an obvious linear isomorphism between this algebra and $\Lambda(V \oplus V)$. The only difference is, as already explained, that the result of a multiplication will at times differ in sign in the two algebras. But this leaves the dimensions of the graded parts, and thus also the Hilbert functions, intact. The results proved here on $\Lambda(V)^{\otimes 2}$ therefore carry over directly to $\Lambda(V \oplus V)$, and therefore suffice to fully prove Theorem 8.1.1. It is but a matter of convenience to use the former description of the algebra. In particular, it allows us to use without additional care some notions from commutative algebra, which we would otherwise have had to adapt quite meticulously to meet our ends.

Definition 8.1.2. We write Δ for the set $\{v \otimes v \mid v \in \{\sum_{i=1}^{k} c^i e_i \mid c \in \mathbb{R}\}\} \subset G$. The *moment algebra of* V, denoted by M, is the subalgebra of G generated by Δ, *i.e.*,

$$M = (\ \Delta\ )\,.$$

Remark 8.1.3. Even though G is not graded, M is again a graded algebra with the elements of Δ being of degree 1. Nevertheless, we will speak of the degree-d part of G, and mean with it the linear space generated by those elements $e_S \otimes e_T$ such that $|S| = |T| = d$.

The definition of M as given above has turned out to be rather unwieldy when it comes to determining h_M. For this reason, we will

proceed to give two different descriptions of M. The first is again a subalgebra of M.

Definition 8.1.4. For $s = 2, \ldots, 2k$, the *s-th skew diagonal* $d_s \in G$ is defined as

$$d_s = \sum_{\substack{i+j=s \\ 1 \leq i,j \leq k}} e_i \otimes e_j \,.$$

The subalgebra S of G is generated by all skew diagonals, that is,

$$S = (d_s \mid 2 \leq s \leq 2k), \,.$$

As a commutative algebra of finite dimension, M is just a quotient of a polynomial algebra, namely—as we shall see—the following.

Definition 8.1.5. For $s \in \mathbb{N}$ with $2 \leq s \leq 4k$, define elements $C_s \in R$ by $C_s = \sum_{i+j=s} t_i t_j$, and let $I_C = \langle C_s \mid 2 \leq s \leq 4k \rangle$. Then, H is defined as

$$H = R/I_C \,.$$

Remark 8.1.6. Since C_s are all homogeneous of degree two, I_C is a homogeneous ideal, and consequently, H is again graded algebra.

8.1.2. A Generalization of M

The definition of M lends itself to the following obvious generalization, as already sketched before.

Definition 8.1.7. Let $A \subseteq V$ be a subset of V. Then $G(A)$ is defined as $G(A) = (a \otimes a \mid a \in S)$.

Remark 8.1.8. Informally, it is evident that $G(A)$ will be typically of dimension much higher than F_{2k+1}. Here, *typically* can mean, for example, a set A of $n \geq \omega(k)$ random vectors over a sufficiently large sample space. Then, the relations imposed by the exterior algebra will lead to $G(A)$ being of dimension equal to the k-th Catalan number C_k almost surely as n tends to infinity, where $C_k \geq \Omega(4^{\delta k})$ for all $\delta < 1$.

The last remark notwithstanding, we will now present a special family of sets A that do *not* show this behaviour, but instead behave at most as bad as M. Indeed, M arises as special case in this family, and as one for which we can also show a lower bound.

Definition 8.1.9. Let $P = \{p_1, \ldots, p_k\} \subset \mathbb{R}[X]$ be a set of k linearly independent univariate real polynomials in X of degree at most k and without constant terms. Denote $V(P) = \left\{ \sum_{i=1}^{k} p_i(x) e_i \mid x \in \mathbb{R} \right\}$, and let $G(P) = G(V(P))$.

Remark 8.1.10. These algebras $G(P)$ are just as suitable for solving the longest path problem as is $M = G(\{X, X^2, \ldots, X^k\})$: It is easy to see that when evaluating the k-walk-polynomial $f(G)$ over $G(P)$, the equivalence between the vanishing of $f(G)$ and the non-existence of a k-path continues to hold. This requires that at the points at which the p_i are evaluated when evaluating $f(G)$, none of the p_i may vanish, but this is easy to accomplish.

We will see that any such algebra $G(P)$ can only ever be of dimension at most equal to that of M. However, a matching lower bound currently seems elusive, giving hope that there might exist choices for P that lead to lower-dimensional algebras and possibly allow to circumvent the barriers described here.

8.2. The Barriers

The following Theorems prove Theorem 8.1.1.

Theorem 8.2.1. *For all $d \in \mathbb{N}$,*

$$h_M(d) = h_H(d) = \binom{2k-d}{d}.$$

Having equal dimension of course isn't sufficient for algebras to be isomorphic. We will however observe rather easily that M is a homomorphic image of H, which then implies the following.

Corollary 8.2.2. *H and M are isomorphic as algebras.*

The combinatorial identity $\sum_{a=1}^{b} \binom{b-a}{a} = F_{b+1}$ and Theorem 8.2.1 imply then:

Corollary 8.2.3. $\dim H = \dim M = F_{2k+1}$.

We finally revisit the definition of M and prove that the more general family of related algebras $G(P)$ behaves at least as nicely as M.

Theorem 8.2.4. *Let $P = \{p_1, \ldots, p_k\}$ be a set of k linearly independent univariate real polynomials of degree at most k and without constant terms. Then, the following holds for all $d \in \mathbb{N}$:*

$$h_{G(P)}(d) \leq h_M(d) .$$

Corollary 8.2.5. $\dim G(P) \leq F_{2k+1}$.

8.3. Proof of Theorem 8.2.1

The procedure of the proof is collected in the following lemmas. They then immediately yield Theorem 8.2.1 and its corollaries.

Lemma 8.3.1. *$S = M$.*

Lemma 8.3.2. *For all $d \in \mathbb{N}$,*

$$h_H(d) \leq \binom{2k-d}{d} .$$

Lemma 8.3.3. *S is a graded homomorphic image of H. In particular, for all $d \in \mathbb{N}$,*

$$h_S(d) \leq h_H(d) .$$

Lemma 8.3.4. *For all $d \in \mathbb{N}$,*

$$\binom{2k-d}{d} \leq h_S(d) .$$

We prove each of them in a separate subsection. Before doing so, we define two further combinatorial objects.

Definition 8.3.5. We call a monomial $m \in R$ a *comb monomial* if it is multilinear, and for all i, $t_i t_{i+1}$ does not divide m. The set of comb monomials is denoted by Ш, and the subset of Ш of degree d is written $Ш_d$.[2]

Let us register for later use a simple fact about Ш.

Proposition 8.3.6. *The number of comb monomials of degree d on $a-1$ variables is given by $\binom{a-d}{d}$.*

For any monomial in the t_i, we associate with it its vector of exponents, that is, the $2k-1$ natural numbers corresponding, for each $2 \leq i \leq 2k$, to the multiplicity with which t_i appears in the monomial. Given such a vector α, we denote by $|\alpha|$ the sum of its entries, which is simply the degree of the monomial belonging to α. We now define a total ordering on the vectors of exponents, and thereby also on the monomials, as follows.

Definition 8.3.7. Let α and β be the associated vectors of exponents of two monomials. Then $\alpha \succ_{\text{grevlex}} \beta$ holds by definition if either

1. $|\alpha| > |\beta|$, or
2. $|\alpha| = |\beta|$ and in addition, the rightmost non-zero entry of $\alpha - \beta$ is negative.

This order is called the *graded reverse lexicographic monomial ordering*.

Some of the notions we employ to prove the theorem, such as the ordering $\succ_{\text{grevlex}}$, stem from the theory of Gröbner bases, and for background on this theory and the borrowed notions, we refer to one of the many textbooks on the topic, such as the one of Kreuzer and Robbiano [KR08].

[2] The Cyrillic Ш is called and, roughly, pronounced *Shah*. However, it was chosen for the iconic rather than phonetic value it carries for representing a comb, an association that is not totally unheard of: The *Dirac comb* (which is, rather unsurprisingly, also called the *Shah function*) is sometimes denoted in the same way.

8.3.1. Equality of S and M

Proof of Lemma 8.3.1. Let U be the linear space generated by all skew diagonals, and let W be the linear space generated by Δ. It suffices to show that $U = W$. First of all, we note that the skew-diagonals are all linearly independent and thus form a basis for U. Consider then a generator of M, that is, an element $v \otimes v \in \Delta$, say $v = \sum_{i=1}^{k} c^i e_i$ for some real non-zero c.

By the properties of the tensor product and after regrouping the summands by their c-degree, we can rewrite $v \otimes v$ as

$$v \otimes v = \sum_{i,j=1}^{k} c^{i+j} e_i \otimes e_j = \sum_{s=2}^{2k} c^s d_s \,. \tag{8.1}$$

We quickly take note of the fact that this shows that $W \subseteq U$ is true.

To demonstrate that $U \subseteq W$ holds, we observe that the coordinates of $v \otimes v$ with respect to the basis $\{d_s\}_{2 \leq s \leq 2k}$ are a scaled Vandermonde vector. Thereby follows the linear independence of any set of $2k - 1$ generators from Δ, provided they all stem from different values of c. This means that, for each $2 \leq s \leq 2k$, there exists a set $\{\lambda_v^{(s)}\}_{v \in M}$ of coefficients, only a finite subset of which are non-zero, witnessing

$$\sum_{v \in M} \lambda_v^{(s)} v \otimes v = d_s \,.$$

Since the left-hand side of this equality obviously lies in W, so does the right-hand side, and $U = W$ follows. □

8.3.2. An upper bound for $\dim H$

Here, we prove Lemma 8.3.2.

Definition 8.3.8. Let $f \in R$ be a polynomial. The *initial monomial of* f $\mathrm{in}(f)$ is the maximum of the monomials appearing in f, with respect to $\succ_{\mathrm{grevlex}}$. For an ideal I of R, its *initial ideal* $\mathrm{in}(I)$ is defined as the ideal generated by all the initial monomials of polynomials in I.

In symbols,

$$\operatorname{in}(I) = \langle \operatorname{in}(f) \mid f \in I \rangle .$$

One quickly observes:

Lemma 8.3.9. *Let $2 \leq s \leq 2k$. Then,*

$$\operatorname{in}(C_s) = \begin{cases} t_{s/2}^2 & \text{if } s \text{ is even, and} \\ t_{(s-1)/2} \cdot t_{(s+1)/2} & \text{if } s \text{ is odd.} \end{cases}$$

Lemma 8.3.10. *The comb monomials form a linear basis of $R/\langle \operatorname{in}(C_s) \mid 2 \leq s \leq 4k \rangle$.*

Proof. This follows directly[3] from the definitions and Lemma 8.3.9. □

Lemma 8.3.11. *Let G be a subset of an ideal I, and let $G' = \langle \operatorname{in}(g) \mid g \in G \rangle$. Then for all $d \in \mathbb{N}$, the following holds.*

$$h_{R/\operatorname{in}(I)}(d) \leq h_{R/G'}(d) .$$

Proof. The fact that $\langle \operatorname{in}(g) \mid g \in G \rangle$ is a subset of $\operatorname{in}(I)$ implies the claim. □

The following is a standard fact.

Lemma 8.3.12. *Let I be an ideal of R. Then,*

$$h_{R/I} = h_{R/\operatorname{in}(I)} .$$

Proof of Lemma 8.3.2. Follows from Lemmas 8.3.9, 8.3.10, 8.3.11, 8.3.12 and Proposition 8.3.6. □

8.3.3. Bounding $\dim M$ by $\dim H$ from above

Here, we prove Lemma 8.3.3. Define a homomorphism φ through

$$\varphi : \ R \to S,\ t_s \mapsto d_s \text{ for } 2 \leq s \leq 2k \tag{8.2}$$

[3] *Directly* means in particular that this is not an application of the Macaulay Basis Theorem, since we only consider the initial terms of the ideal generators in the first place.

and extend φ uniquely to all of R by its universal property. The following lemma immediately implies Lemma 8.3.3.

Lemma 8.3.13. *The algebra homomorphism φ is a graded homomorphism, and* $\ker\varphi \supseteq I_C$.

Proof. Gradedness of φ is evident from the definition. For the inclusion to hold true, it suffices if for the generators C_q of I_C, it holds that

$$\varphi(C_q) = 0$$

for all q. By virtue of Theorem 8.3.1, we may do this as well by thinking of S as M, therefore letting $v \otimes v \in \Delta$, say $v = \sum_{i=1}^{k} c^i e_i$ for some real c. By the properties of the exterior algebra, it holds for all such $v \otimes v$ that $(v \otimes v)^2 = 0$. Using the representation from Eq. (8.1) of $v \otimes v$ in terms of d_s, this implies

$$(v \otimes v)^2 = \left(\sum_{s=2}^{2k} c^s d_s \right)^2 = \sum_{r,s=2}^{2k} c^{r+s} d_r d_s = 0 \,.$$

Sorting once again by c-degree, one obtains

$$\sum_{q=4}^{4k} c^q \cdot \left(\sum_{r+s=q} d_r d_s \right) = 0 \,.$$

The left-hand side in this equation is a univariate polynomial in c of degree at most $4k$, with coefficients from M. Since the choice of c in the definition of $v \otimes v$ was arbitrary, we conclude that the equality in fact holds true for all such c. Of course, a non-zero polynomial of degree $4k$ can have at most $4k$ roots, and we may thus conclude that the entirety of the coefficients $\sum_{r+s=q} d_r d_s$ must vanish. It is crucial to bear in mind that, even while the coefficients come from a nilpotent algebra, this argument refers to c as the indeterminate, and c only ever assumes real values. If one were to be pedantic, one might say that the single equation above actually corresponds to $h_M(2)$ equations on the real coordinates of the degree-two part of M, leading to the very same

conclusion. Namely, that

$$\varphi(C_q) = \varphi\left(\sum_{i+j=q} t_i t_j\right) = \sum_{i+j=q} \varphi(t_i)\varphi(t_j) = \sum_{i+j=q} d_i d_j = 0$$

for all q. □

8.3.4. A lower bound for $\dim S$

We show that the images of comb monomials under φ, as defined in (8.2), remain linearly independent. To this end, we shall exhibit, for each degree d, an explicit subspace U_d of dimension $\binom{2k-d}{d}$ of the degree-d part of G such that the set $\{\varphi(m)\}_{m\in \text{Ш}_d}$ remains linearly independent after projecting it onto U_d.

The construction goes as follows. We assume the sets of naturals appearing in the sequel to be in ascending order, for ease of notation—which is not to say that they were ordered sets, just that we write them down in a specific way.

Let $S \subseteq \{1, \ldots, k\}$ be of size d. We say that S has its *first upward gap at i* if S is of the form $\{1, 2, \ldots, i-1, i, j, \ldots, s\}$ with $j > i+1$. By convention, sets of the form $\{j, \ldots, s\}$ with $j > 1$ have their first upward gap at 0, while $\{1, \ldots, d\}$ has its first upward gap at d.

Analogously, we say that S has its *first downward gap at i* if S is of the form $\{s, \ldots, k-j, k-i, k-(i-1), \ldots, k-1, k\}$, where again, $j > i+1$. We make the corresponding conventions for downward gaps.

Definition 8.3.14. Let $S, S' \subseteq \{1, \ldots, k\}$ be of size d.

1. The *ordered pair* (S, S') is *gap-compatible* if S has its first upward gap at i, and S' has its first downward gap at j for some $j \geq d-i$.

2. The linear subspace $U_d \subseteq G$ is the subspace generated by all $e_S \otimes e_{S'}$, where (S, S') is gap-compatible.

Example 1. The following is to make lither the rather bulky definitions above.

- Both $\{1,2,3,6,7\}$ and $\{1,2,3,5,8\}$ have their first upward gap at 3.
- $\{1,2,4,5,8,9\}$ has its first downward gap at 1 if $k=9$, and at 0 if $k>9$.
- If S has its first upward gap at d, then (S,S') is gap-compatible for every S'.
- If S has its first upward gap at 0, then $(S,\{k-d,k-d+1,\ldots,k\})$ is the only gap-compatible pair.

For a comb monomial m in R, let us denote with $\sigma(m)$ the sum of indices of the generators it is composed of. With some slight abuse of notation, we also mean, for any basis element $e_S \otimes e_T$ of G, the sum of the elements in S and T whenever we write $\sigma(e_S \otimes e_T)$. Formally, σ then becomes a map $\sigma : Ш \cup B \to \mathbb{N},\ x \mapsto \sigma(x)$, where B the standard basis of G. For any element x of G, let us write $\operatorname{supp} x$ to indicate the set of standard basis elements that appear in the basis representation of x with non-zero coefficient.

Lemma 8.3.15. *Let $m \in Ш$, and let $b \in \operatorname{supp}\varphi(m)$. Then, the following holds.*

$$\sigma(m) = \sigma(b)$$

Proof. Let $m = t_{i_1} \cdots t_{i_d} \in Ш$. Consider $\varphi(m) = \delta_{i_1} \cdots \delta_{i_d}$. By definition, each δ_{i_j} is a sum of basis elements $e_s \otimes e_t$ satisfying $s+t=i_j$. Therefore, $\sigma(b') = i_j$ for each $b' \in \operatorname{supp}\delta_{i_j}$. Observe that every element in $\operatorname{supp}\varphi(m)$ is a product consisting of exactly one element from each $\operatorname{supp}\delta_{i_j}$. It is easy to see σ is additive with respect to the product of basis elements, provided the result is non-zero, and the claim follows. □

We define a mapping

$$\kappa : G \mapsto R,\ x \mapsto \kappa(x)\,,$$

defined on the standard basis B of G through $\kappa(e_S \otimes e_T) = t_{i_1} \cdots t_{i_d}$, where $i_j = a_j + b_j$ for all j. Here, a_j and b_j are the respective elements

of S and T. By convention, we assume that $a_1 < \ldots < a_d$ and $b_1 < \ldots < b_d$.

For any s, we define G_d^s to be the subspace of the degree-d part of G generated by those $e_S \otimes e_T$ such that $\sigma(e_S \otimes e_T) = s$, and analogously $V_d^s \subseteq R$ as the space generated by the comb monomials m having $\sigma(m) = s$. Lemma 8.3.15 then implies that the restriction of φ to

$$\varphi_d^s : V_d^s \to G_d^s,\ x \mapsto \varphi(x)\,,$$

is well-defined. Furthermore, we set $U_d^s = G_d^s \cap U_d$, and denote with

$$\pi_d^s : G_d^s \to U_d^s$$

the projection of G_d^s onto U_d^s, which just corresponds to keeping only a subset of basis vectors. We write Φ_d^s for the transpose of the matrix representing the linear map $\pi_d^s \circ \varphi_d^s$ over the respective standard bases of U_d^s and G_d^s. Hence, the rows of Φ_d^s are indexed by $Ш_d^s$, and its columns by the standard basis of U_d^s.

Let $Ш_d^s$ be the set of comb monomials m of degree d and having $\sigma(m) = s$, and let B_d^s the standard basis of U_d^s. Let $m = t_{i_1} \cdots t_{i_d} \in Ш_d^s$, where we assume that $i_1 < i_2 < \cdots < i_d$ holds. We associate with m through $\iota_d^s : Ш_d^s \to B_d^s,\ b \mapsto \iota_d^s(b)$ the basis element $\iota_d^s(m) = e_S \otimes e_T$, where S and T are, in a certain sense, chosen of minimal value with respect to the numbers they contain, viz.: Letting r be the largest j with the property that $i_j \leq k$, S is defined by

$$S = \{1, \ldots, r\} \cup \{i_j - k + d - j\}_{j>r}\,,$$

and T is then already determined by this to satisfy

$$T = \{i_1 - 1, \ldots, i_r - r\} \cup \{k - d + j\}_{j>r}\,.$$

Note that directly from the definitions and Lemma 8.3.15, the image of ι_d^s is contained in U_d^s, so that the mapping is well-defined.

Lemma 8.3.16. *The mapping ι_d^s is an injection, and is inverted on*

its image by κ.

Proof. The fact that m is a comb monomial entails that

$$i_1 - 1 < i_2 - 2 < \ldots < i_d - d \tag{8.3}$$

holds. It is clear that the sequence of the i_j can be reconstructed from S and T in the stated way using κ, but only if $\prod_{s\in S} e_s$ and $\prod_{t\in T} e_t$ are both non-zero—that is, provided that S and T are actually *sets* in that every element appears only once. That this is the case in the enumeration used to produce the definition of S and T is implied by (8.3), and ι_d^s is hence injective. □

Lemma 8.3.17. *For all d and s, the matrices Φ_d^s are square, and consequently, ι_d^s is a bijection.*

Proof. It holds that

$$\dim U_d = \sum_{i=0}^{d} \binom{k-i-1}{k-d-1} \cdot \binom{k-d+i}{k-d}. \tag{8.4}$$

This can be seen as follows: The sum is sorted by the position of the first upward gap. For each i, $\binom{k-i-1}{k-d-1}$ is the number of sets having their first upward gap at i, and $\binom{k-d+i}{k-d}$ is then the number of sets having their first downward gap at j for some $j \geq d - i$.

A variant of the classical Chu-Vandermonde identity then yields:

$$\sum_{i=0}^{d} \binom{k-i-1}{k-d-1} \cdot \binom{k-d+i}{k-d} = \binom{2k-d}{d}. \tag{8.5}$$

Combining (8.5) and (8.4) with Lemma 8.3.15 shows that the direct matrix sum $\bigoplus_s \Phi_d^s$, *i.e.*, the block diagonal matrix with blocks Φ_d^s, is square. Lemma 8.3.16 then shows that each of the summand matrices has at least as many columns as it has rows. These two statements imply the claim. □

Lemma 8.3.18. *Let m be a comb monomial of degree d with $\sigma(m) = s$. Consider the represenation of $\varphi(m)$ in the standard basis of G. Then, the coefficient of $\iota_d^s(m)$ in this representation equals one.*

Proof. We proceed by induction on d, for fixed k. For convenience, we set $\iota = \iota_d^s$. If $d = 1$, then the claim is trivially true: $\iota(m) = e_1 \otimes e_{i_1-1}$, or $e_{i_1-k} \otimes e_k$ if $i_1 > k$, is the relevant basis element, and it appears per definition in δ_{i_1} with coefficient one. Assume now that $d > 1$ holds, and that the claim holds for $d-1$. Let $m' = m/t_{i_d} = t_{i_1} \cdots t_{i_{d-1}}$. By inductive hypothesis, m' contains $\iota(m')$ with coefficient one. We now will argue two things: (1) $\iota(m')$ can be extended (*i.e.*, multiplied from the right) by exactly one element of $\operatorname{supp} \delta_{i_d}$ to yield $\iota(m)$, and (2) all other elements of $\operatorname{supp} \varphi(m')$ cannot be extended in the same way.

Towards the first claim, observe that, similar to the case of $d = 1$, the element $b = e_d \otimes e_{i_d-d}$, or $b = e_{i_d-k} \otimes e_k$ if $i_d > k$, does the trick (this does *not* mean that e_k can appear more than twice on the right side of the tensor product, which would make the product vanish): We have $\iota(m') \cdot b = \iota(m)$. This extension b is unique: Take any other element of $\operatorname{supp} \delta_{i_d}$, and it will produce either zero, or some other basis element that is distinct from $\iota(m)$.

Towards the second claim, let S and T correspond to m as in the definition of ι, and let $e_Q \otimes e_R$ be some basis element with $e_Q \otimes e_R \neq \iota(m')$. We assume that $i_d \leq k$. Ignoring the signs, right multiplication with an element $e_s \otimes e_t \in \operatorname{supp} \delta_{i_d}$ corresponds to disjoint set union with Q and R, and we seek s and t such that $S = Q \cup \{s\}$ and $T = R \cup \{t\}$. Therefore, by Lemma 8.3.15, $Q = \{1, \ldots, s-1, s+1, \ldots, d\}$ and $R = \{i_1 - 1, \ldots, i_{s-1} - s + 1, i_{s+1} - s - 1, \ldots, i_d - d\}$. The sum of the entries in Q and R is thus

$$\begin{aligned} &\left(\sum_{j=1}^{d} j\right) - s + \left(\sum_{j=1}^{d} (i_j - j)\right) - (i_s - s) \\ = &\left(\sum_{j=1}^{d} i_j\right) - i_s \neq \left(\sum_{j=1}^{d} i_j\right) - i_d = \sigma(m')\,, \end{aligned}$$

which implies that $e_Q \otimes e_R \notin \operatorname{supp} \varphi(m')$ by Lemma 8.3.15. The argument goes along the same lines if $i_j \geq k$ for some j. □

Define a mapping

$$P : \mathfrak{S}_d \times G_d \to R,\ (\pi, e_S \otimes e_T) \mapsto t_{i_1} \cdots t_{i_d}\,,$$

where $i_j = a_j + b_{\pi(j)}$ for all j and $(a_j)_j$ and $(b_j)_j$ are strictly increasing, as in the definition of κ. Observe that $P(\mathrm{id}, \cdot) = \kappa$, and hence $P(\mathrm{id}, e_S \otimes e_T) \in \text{Ш}$ whenever (S, T) is gap-compatible.

Lemma 8.3.19. *Consider an element $e_S \otimes e_T$ of the standard basis of G_d. Assume $e_S \otimes e_T \in \operatorname{supp} \varphi(m)$ for some $m \in \text{Ш}$. Then, there must be a permutation $\pi \in \mathcal{S}_d$ with $P(\pi, e_S \otimes e_T) = m$.*

Proof. This follows directly from the definitions of the δ_j and the multiplication in G. □

Lemma 8.3.20. *Consider an element $e_S \otimes e_T$ of the standard basis of G_d. Then, the coefficient of $e_S \otimes e_T$ in the standard representation of $\varphi(m)$ equals zero whenever $m \succ_{\text{grevlex}} \kappa(e_S \otimes e_T)$.*

Proof. Let $\kappa_{S,T} = \kappa(e_S \otimes e_T) = t_{j_1} \cdots t_{j_d}$. As before, let

$$S = \{a_1, \ldots, a_d\},\ T = \{b_1, \ldots, b_d\}$$

listed in ascending order. With Lemma 8.3.19, it suffices to show that

$$\kappa_{S,T} \succ_{\text{grevlex}} P(\pi, e_S \otimes e_T) \text{ for all } \pi \in \mathcal{S}_d \text{ with } \pi \neq \mathrm{id}\,.$$

To this end, let $\pi \neq \mathrm{id}$ be such a permutation. Let $f \in [d]$ be the greatest index such that f is not a fixed point of π, which exists since $\pi \neq \mathrm{id}$ was assumed. Then, by assumption, for all $g > f$, we have $\pi(g) = g$ and $a_g + b_{\pi(g)} = a_g + b_g = j_g$, such that $P(\pi, e_S \otimes e_T)$ agrees with $\kappa_{S,T}$ on all indices greater than f. Then, because f is maximal, $\pi(f) \neq f$ implies that $\pi(f) < f$, and so, since the b_i are sorted in ascending order, it follows that $a_f + b_{\pi(f)} < a_f + b_f = j_f$, so that the first non-zero entry from the right in the resulting exponent vectors

will be negative, which is the very definition of being smaller than $\kappa_{S,T}$ with respect to the order $\succ_{\text{grevlex}}$. □

Lemma 8.3.21. *The rows and columns of the matrix Φ_d^s, which are indexed by comb monomials and gap-compatible pairs, respectively, can be permuted such that the resulting matrix is lower-triangular with ones on the main diagonal, and hence of full rank.*

Proof. According to Lemma 8.3.17, Φ_d^s is square. According to Lemma 8.3.20, the columns of Φ_d^s, which are indexed by gap-compatible pairs, are zero above a certain point, to wit, in all rows indexed by a comb monomial that is greater than $\kappa(e_S \otimes e_T)$ with respect to $\succ_{\text{grevlex}}$. Furthermore, due to Lemmas 8.3.16 and 8.3.18, they have their topmost 1 all in different rows. Taking these things together, and reordering columns accordingly, then proves the claim. □

Proof of Lemma 8.3.4. Follows immediately from Lemma 8.3.21. □

8.4. Proof of Theorem 8.2.4

The theorem follows directly from the following lemma.

Lemma 8.4.1. *Let $P = \{p_1, \ldots, p_k\} \subset \mathbb{R}[X]$ be a set of k linearly independent univariate real polynomials in X of degree at most k and without constant terms. Then, $G(P)$ is the image of H under a graded homomorphism.*

Proof. Denote the coefficient of X^s in the product $p_i \cdot p_j$ by c_{ij}^s, that is, $\{c_{ij}^s\}_s$ is the unique set of reals that witnesses $p_i \cdot p_j = \sum_{s=2}^{2k} c_{ij}^s X^s$ for all i, j. Defining now $\tau_q = \sum_{i,j} c_{ij}^s e_i \otimes e_j$ and, as in (8.2), a homomorphism φ_P through

$$\varphi_P : \; R \to G(P),\, t_s \mapsto d_s \text{ for } 2 \leq s \leq 2k\,, \tag{8.6}$$

one can the see that the $\{\tau_q\}_q$ behave very much the same as the $\{d_q\}_q$ did before: The generator $v \otimes v = \sum_{i=1}^{k} p_i(x) e_i$ for $v \in S(P)$ and such

that $p_i(x) \neq 0$ for all i, is expressible as

$$v \otimes v = \sum_{s=2}^{2k} x^s \tau_s \,.$$

Furthermore, the relation $(v \otimes v)^2 = 0$ again proves that $\ker \varphi_P$ contains the generators of I_C, and the claim follows. □

Remark 8.4.2. The lower bound argument does not extend to $G(P)$ in a straightforward way. This is hopeful insofar as it does not immediately rule out the existence of even smaller algebras than M that possibly enjoy the properties needed for the longest path problem.

8.5. Extensions

Pratt [Pra19; Pra18] has recently taken a seemingly entirely different approach to algebraic algorithms for subgraph detection problems. Namely, he uses so-called *apolar ideals* to construct an algebra of dimension $\sqrt{6.75}^k < 2.6^k$ that can be used much in the same way to solve subgraph detection and approximate counting as the algebras studied here. To wit, this algebra is the apolar algebra of the determinant polynomial of a generic Hankel matrix. In short, this is the quotient of the algebra of commuting differential operators by the annihilator of the generic Hankel matrix under the natural action of differential operators on polynomials.

It is at first not at all clear what connection, if any, there is, between these algebras and the objects studied in this chapter. Without proof and definitions, let us sketch this connection, which will be fully developed in [Bra]:

The algebras of this chapter are in particular not *Gorenstein*. In our setting, this means as much as saying that there are elements in H of degree i that are annihilated by all elements of degree $k - i$, and are, from a computational perspective, therefore somehow useless. In contrast, apolar algebras are Gorenstein, and MacCaulay's inverse systems [Mac94] asserts, roughly, that every (local Artinian) Gorenstein

algebra arises as an apolar algebra.

The catch is now that, if we consider the ideal generated by all those aforementioned annihilators in H, and quotient them out, what we obtain is obviously a Gorenstein algebra, and is in fact isomorphic to the apolar algebra studied by Pratt.

On the other hand, we can obtain H back from the apolar setting by not taking the quotient by *all* differential operators that annihilate the generic Hankel matrix, but only the quotient by the ideal that is generated by differential operators in degree two. This algebra is then not Gorenstein anymore, and is isomorphic to H.

This is interesting from a mathematical perspective: It is a typical question to ask for apolar ideals whether they are generated in degree two, and this is indeed the case for generic determinants, permanents, generic symmetric determinants, and generic symmetric permanents [Sha13; Sha15]. However, it seems not to be the case for generic Hankel matrices. Using the isomorphism results from this section allows for a much more concrete treatment of the appearing algebraic objects. This, in connection with Conca's results on spaces of minors of generic Hankel matrices [Con98], presents a promising approach to proving the lower bounds missing in Pratt's work.

Part III.

Counting Forests and the Tutte Plane

9. Introduction

> *Let me see: four times five is twelve, and four times six is thirteen, and four times seven is—oh dear! I shall never get to twenty at that rate!*
>
> Alice in Lewis Carroll's *Alice in Wonderland*

Counting combinatorial objects is at least as hard as detecting their existence, and often it is harder. Valiant [Val79] introduced the complexity class #P to study the complexity of counting problems and proved that counting the number of perfect matchings in a given bipartite graph is #P-complete. By a theorem of Toda [Tod91], we know that $\mathsf{PH} \subseteq \mathsf{P}^{\#\mathsf{P}}$ holds; in particular, for every problem in the entire polynomial-time hierarchy, there is a polynomial-time algorithm that is given access to an oracle for counting perfect matchings. This theorem suggests that counting is much harder than decision.

There are now two possibilities: Either one accepts one's fate and tries to make the best out of the situation, by trying to find algorithms that are maybe not fast, but at least not as bad as the brute force solution. For problems that are solvable in time $2^{O(n)}$, for example (where n is the input size), we might look for algorithms that solve the problem in time $2^{o(n)}$. We then want to determine the exponential complexity of a problem as precisely as possible, *i.e.*, either give better algorithms, or prove better lower bounds. This is done in the field of exact algorithms.

On the other hand, we might opt to take a step back and relax our requirements. One very popular thing to do is to relax these requirements with respect to the solution quality. For instance, we may consider approximation algorithms. Another option—and the one we consider here—is to be content with an algorithm that is sufficiently

fast only on *some* instances. Of course, it is very easy to produce an algorithm that is fast on some arbitrary set of trivial instances of the problem. In order to make this task non-trivial, one wants to *parameterize* the difficulty of instances in some way. This is the approach chosen by parameterized algorithms and complexity.

At the danger of seeming somewhat heterodox, we collect both of these approaches under the term of fine-grained algorithms and complexity. In this part, we will analyze the fine-grained complexity of the problem of counting the number of forests (*i.e.*,, all acyclic subsets of edges) of a given size, which here means the number of edges.

To both approaches, the following definitions are crucial. A *k-forest* is an acyclic graph consisting of k edges and a *k-tree* is a connected k-forest. We say that two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are *isomorphic* if there is a bijection $\varphi : V_1 \to V_2$ such that for all $u, v \in V_1$, $\{u, v\} \in E_1$ if and only if $\{\varphi(u), \varphi(v)\} \in E_2$. A *k-matching* of a graph $G = (V, E)$ is a subset of k edges such that no pair of edges has a common vertex.

A *weighted graph* (G, w) is an undirected graph G in which every edge $e \in E(G)$ has a weight w_e, which is an element of some ring. We use the *multivariate forest polynomial*, defined e.g. by Sokal [Sok05, (2.14)] as follows:

$$F(G; w) = \sum_{A \in \mathcal{F}(G)} \prod_{e \in A} w_e \,. \tag{9.1}$$

Setting all weights w_e to the single variable x yields the *univariate forest polynomial*:

$$F(G; x) = \sum_{A \in \mathcal{F}(G)} x^{|A|} = \sum_{k=0}^{|E(G)|} a_k(G) x^k \,,$$

where $a_k(G)$ is the number of forests with k edges in G. For all $x \in \mathbb{R} \setminus \{1\}$, the formal relation between $T(G; x, 1)$ and the univariate forest

polynomial is given by the identity

$$\begin{aligned} T(G;x,1) &= (x-1)^{|V|-k(E)} \sum_{A\in\mathcal{F}(G)} (x-1)^{-|A|} \\ &= (x-1)^{|V|-k(E)} \cdot F\Big(G;\frac{1}{x-1}\Big)\,. \end{aligned} \tag{9.2}$$

The first equality follows from (10.2) and the discussion preceding it. Thus, evaluating the forest polynomial and evaluating the Tutte polynomial for $y = 1$ are polynomial-time equivalent.

10. The Exponential Complexity of Counting Forests

10.1. Introduction

When faced with a problem that is NP-hard or #P-hard, the area of exact algorithms strives to find the fastest exponential-time algorithm for a problem, or find reasons why faster algorithms might not exist. For example, Björklund proved the following.

Theorem 10.1.1 ([Bjö12])**.** *There is an algorithm for counting perfect matchings in n-vertex graphs in time $2^{n/2}\operatorname{poly}(n)$.*

It has been hypothesized that no $1.99^{n/2}$-time algorithm for the problem exists. But we do not know whether such an algorithm has implications for the strong exponential time hypothesis, which states: For all $\varepsilon > 0$, there is some k such that the problem of deciding satisfiability of boolean formulas in k-CNF on n variables does not have an algorithm running in time $(2-\varepsilon)^n$. However, Dell *et al.* proved that the term $O(n)$ in the exponent is asymptotically tight, in the following sense.

Theorem 10.1.2 ([Del+14])**.** *Unless rETH fails, there is no $\exp\big(o(n)\big)$-time algorithm for counting perfect matchings in n-vertex graphs.*

would violate the (randomized) exponential time hypothesis (ETH) by Impagliazzo and Paturi [IP01]. Using the idea of block interpolation, Curticapean [Cur15] strengthened the hardness by showing that a

$\exp\big(o(n)\big)$-time algorithm for counting perfect matchings would violate the (deterministic) counting exponential time hypothesis (#ETH).

We prove hardness results under #ETH for the problem of counting all forests in a graph, that is, its acyclic subgraphs. In particular, we show that, if #ETH holds, then this problem does not have an algorithm running in time $\exp(o(n))$ even in simple n-vertex graphs of bounded average degree. We use this result to lift two known "FP vs. #P-hard" dichotomy theorems to their more refined and asymptotically tight "FP vs. #ETH-hard" variants. Here, FP is the class of functions computable in polynomial time. Since ETH implies #ETH, our results could also be stated under ETH.

10.1.1. The Tutte Polynomial under #ETH

The Tutte polynomial of a graph G with vertices V and edges E is the bivariate polynomial $T(G;x,y)$ defined via

$$T(G;x,y) = \sum_{A\subseteq E} (x-1)^{k(A)-k(E)}(y-1)^{k(A)+|A|-|V|}\,, \qquad (10.1)$$

where $k(A)$ is the number of connected components of the graph (V,A). The Tutte polynomial captures many combinatorial properties of a graph in a common framework, such as the number of spanning trees, forests, proper colorings, and certain flows and orientations, but also less obvious connections to other fields, such as link polynomials from knot theory, reliability polynomials from network theory, and (perhaps most importantly) the Ising and Potts models from statistical physics. We make no attempt to survey the literature or the different applications for the Tutte polynomial, and instead refer to the upcoming CRC handbook on the Tutte polynomial [EMon].

Since $T(G;-2,0)$ corresponds to the number of proper 3-colorings of G, we cannot hope to compute all coefficients of $T(G;x,y)$ in polynomial time, unless $\#P = \mathsf{P}$. Instead, we focus on the complexity of evaluating the Tutte polynomial at fixed evaluation points. That is, for

each $(x, y) \in \mathbb{Q}^2$, we consider the function $T_{x,y}$ defined as

$$G \mapsto T(G; x, y)\,.$$

Jaeger, Vertigan, and Welsh [JVW90] proved that the behavious of this function is *dichotomous*: It is either #P-hard to compute, or has a polynomial-time algorithm—*i.e.*, there are no intermediacies.

In particular, if (x, y) satisfies $(x-1)(y-1) = 1$, then $T_{x,y}$ corresponds to the 1-state Potts model and has a polynomial-time algorithm. If (x, y) is one of the four points $(1, 1)$, $(-1, -1)$, $(0, -1)$, or $(-1, 0)$, it also has a polynomial-time algorithm. The most interesting point here is $T(G; 1, 1)$, which corresponds to the number of spanning trees in G.

A naïve algorithm to compute the Tutte polynomial of a graph with m edges runs in time $\exp\big(O(m)\big)$. Björklund *et al.* [Bjö+08] gave an algorithm running in time $\exp\big(O(n)\big)$, where n is the number of vertices. Dell *et al.* [Del+14] proved for all hard points, except for points with $y = 1$, that an $\exp\big(o(n/\log^3 n)\big)$-time algorithm for $T_{x,y}$ on simple graphs would violate #ETH. Distressingly, this result not only left open one line, but also left a gap in the running time. Curticapean [Cur15] introduced the technique of block interpolation to close the running time gap: Under #ETH, there is no $\exp\big(o(n)\big)$-time algorithm for $T_{x,y}$ on simple graphs at any hard point (x, y) with $y \neq 1$.[1]

We resolve the complexity of the missing line $y = 1$ under #ETH. On this line, the Tutte polynomial counts forests weighted in some way, and the main result is the following theorem.

Theorem 10.1.3 (Forest counting is hard under #ETH)**.** *If #ETH holds, then there exist constants $\varepsilon, C > 0$ such that no $\exp(\varepsilon n)$-time algorithm can compute the number of forests in a given simple n-vertex graph with at most Cn edges.*

The fact that the problem remains hard even on simple sparse graphs makes the theorem stronger. The previously best known lower bound under #ETH was that forests cannot be counted in time $\exp(n^\delta)$ where

[1] The conference version of [Cur15] does not handle the case $y = 0$, but the full version [Cur18] does.

$\delta > 0$ is some constant depending on the instance blow-up caused by the known #P-hardness reductions for forest counting; the thesis of Taslaman [Tas13] shows a detailed proof for $\delta = \frac{1}{8}$. Our approach also yields a #P-hardness proof for forest counting that is simpler than the proofs we found in the literature, such as the proof appearing in "Complexity of Graph Polynomials" by Steven D. Noble, chapter 13 of [GM07].[2]

Combined with all previous results [JVW90; Del+14; Cur15], we can now formally state a complete #ETH dichotomy theorem for the Tutte polynomial over the reals.

Theorem 10.1.4 (Dichotomy for the real Tutte plane under #ETH)**.** *Let* $(x, y) \in \mathbb{Q}^2$*. If* (x, y) *satisfies*

$$(x-1)(y-1) = 1 \text{ or } (x, y) \in \big\{(1,1), (-1,-1), (0,-1), (-1,0)\big\},$$

then $T_{x,y}$ *can be computed in polynomial time. Otherwise* $T_{x,y}$ *is #P-hard and, if #ETH is true, then there exists* $\varepsilon > 0$ *such that* $T_{x,y}$ *cannot be computed in time* $\exp(\varepsilon n)$*, even for simple graphs.*

The result also holds for sparse simple graphs. We stated the results only for rational numbers in order to avoid discussions about how real numbers should be represented.

For the proof of Theorem 10.1.3, we establish a reduction chain that starts with the problem of counting perfect matchings on sparse graphs, which is known to be hard under #ETH. As an intermediate step, we find it convenient to work with the multivariate forest polynomial as defined, for example, by Sokal [Sok05]. After a simple transformation of the graph, we are able to extract the number of perfect matchings of the original graph from the multivariate forest polynomial of the transformed graph, even when only two different variables are used. Subsequently, we use Curticapean's idea of block interpolation [Cur15] to reduce the problem of computing all coefficients of the bivariate forest polynomial to the problem of evaluating the univariate forest

[2] For a different and not fully published proof, Jaeger, Vertigan and Welsh [JVW90] refer to private communication with Mark Jerrum as well as the PhD thesis of Vertigan [Ver91].

polynomial on multigraphs where all edge multiplicities are bounded by a constant. Finally, we replace parallel edges with parallel paths of constant length to reduce to the problem of evaluating the univariate forest polynomial on simple graphs.

The exponential time hypothesis (ETH) by Impagliazzo and Paturi [IP01] is that satisfiability of 3-CNF formulas cannot be computed substantially faster than by trying all possible assignments. The counting version of this hypothesis [Del+14], which is a weaker assumption (clearly, counting the number of solutions entails deciding existence of a solution), reads as follows:

(#ETH) *There is a constant $\varepsilon > 0$ such that no deterministic algorithm can compute* #3-SAT *in time* $\exp(\varepsilon n)$, *where n is the number of variables.*

A different way of formulating #ETH is to say no algorithm can compute #3-SAT in time $\exp(o(n))$. The latter statement is clearly implied by the formal statement, and it will be more convenient for discussion to use this form.

The sparsification lemma by Impagliazzo, Paturi, and Zane [IPZ01] is that every k-CNF formula φ can be written as the disjunction of $2^{\varepsilon n}$ formulas in k-CNF, each of which has at most $c(k,\varepsilon)n$ clauses. Moreover, this disjunction of sparse formulas can be computed from φ and ε in time $2^{\varepsilon n}\operatorname{poly}(m)$. The density $c = c(k,\varepsilon)$ is the *sparsification constant*, and the best known bound is $c(k,\varepsilon) = (k/\varepsilon)^{3k}$ [CIP06]. It was observed [Del+14] that the disjunction can be made so that every assignment satisfies at most one of the sparse formulas in the disjunction, and so the sparsification lemma applies to #ETH as well. In particular, #ETH implies that #3-SAT cannot be computed in time $\exp(o(m))$, where m is the number of clauses.

We also make use of the following result, whose proof is based on block interpolation.

Theorem 10.1.5 (Curticapean [Cur15])**.** *If #ETH holds, then there are constants $\varepsilon, D > 0$ such that computing the number of perfect matchings of G has no $\exp(\varepsilon n)$-time algorithms on n-vertex graphs G,*

even if G is simple and of maximum degree at most D:

10.2. Counting Forests is #ETH-hard

Let $\mathcal{F}(G)$ be the set of all *forests* of G, that is, edge subsets $A \subseteq E(G)$ such that the graph $(V(G), A)$ is acyclic. Consider now the definition of the Tutte polynomial in (10.1). For $y = 1$, the expression $(y-1)$ is of course zero. Conventionally, $0^t = 1$ if and only if $t = 0$, and 0 otherwise. Therefore, the only non-vanishing summands on the right side of (10.1) are those where $(y-1)$ appears with an exponent of zero. For a single summand, this is the case precisely if $k(A) + |A| - |V| = 0$. This, in turn, is equivalent to A being a forest, which allows us to conclude:

$$T(G; x, 1) = \sum_{A \in \mathcal{F}(G)} (x-1)^{k(A)-k(E)}\,. \tag{10.2}$$

The goal of this section is prove that, for every fixed $x \neq 1$, computing the value $T(G; x, 1)$ for a given graph G is hard under #ETH. More formally, we show the following theorem.

Theorem 10.2.1. *Let $x \in \mathbb{R} \setminus \{1\}$. If* #ETH *holds, then there exist $\varepsilon, C > 0$ such that the function that maps simple n-vertex graphs G with at most Cn edges to the value $T(G; x, 1)$ cannot be computed in time $2^{\varepsilon n}$.*

In particular, this is true for $T(G; 2, 1)$, which is the number of forests in G. Thus, Theorem 10.2.1 yields Theorem 10.1.3 as its special case with $x = 2$.

For a forest $A \in \mathcal{F}(G)$, let $\mathcal{C}(A)$ be the family its connected components. A connected component of A is a maximal set $U \subseteq V(G)$ that is connected in $(V(G), A)$. Clearly $\mathcal{C}(A)$ is a partition of $V(G)$, each element is a (maximal) tree of A, and trees U with $|U| = 1$ are allowed.

Lemma 10.2.2 (Adding an apex)**.** *Let (G, w) be a weighted graph. Let (G', w') be obtained from it by adding a new vertex a and joining it*

with each vertex $v \in V(G)$ using edges of weight z_v. Then

$$F(G'; w') = \sum_{A \in \mathcal{F}(G)} \left(\prod_{e \in A} w_e \prod_{U \in \mathcal{C}(A)} \left(1 + \sum_{u \in U} z_u \right) \right). \tag{10.3}$$

Moreover, if we set $z_v = -1$ for all $v \in V(G)$ and $w'_e = w$ for all $e \in E(G)$, then the coefficient of $w^{n/2}$ in $F(G'; w')$ equals the number of perfect matchings of G in absolute value.

Proof. We first define a projection ϕ that maps any forest A' in the graph G' to a forest $\phi(A')$ in the original graph G. In particular, ϕ simply removes all edges added in the construction of G', that is, we define $\phi(A') = E(G) \cap A'$ for all $A' \in \mathcal{F}(G')$. Clearly, $\phi(A')$ is a forest in G.

Next we characterize the forests A' that map to the same $\phi(A')$. Let A be a fixed forest in G. Then a forest A' in G' maps to A under ϕ if and only if the set $X := A' \setminus A$ satisfies the following property:

(P) For every tree $U \in \mathcal{C}(A)$, at most one edge of X is incident on U.

The forward direction of this claim follows from the fact that A' is a forest, and so in addition to the trees $U \in \mathcal{C}(A)$ it can contain at most one edge connecting each U to a; otherwise the tree and the two edges to a would contain a cycle in A'. For the backward direction of the claim, observe that adding a set X with the property (P) to A cannot introduce a cycle.

Finally, we calculate the weight contribution of all A' that map to the same A. Let A' be a forest in G, let $A = \phi(A')$ and $X = A' \setminus A$. The weight contribution of A' in the definition of $F(G')$ is $\prod_{e \in A'} w'_e$. For all $e \in A$, we have $w'_e = w_e$. For each $e \in X$, we have $e = \{a, v_e\}$ for some $v_e \in V(G)$, and thus $w'_e = z_{v_e}$. Summing over all weight terms for A' with image A yields

$$\sum_{\substack{A' \in \mathcal{F}(G') \\ \phi(A') = A}} \prod_{e \in A} w'_e = \prod_{e \in A} w_e \cdot \sum_{X} \prod_{e \in X} z_{v_e} = \prod_{e \in A} w_e \cdot \prod_{U \in \mathcal{C}(A)} \left(1 + \sum_{u \in U} z_u \right). \tag{10.4}$$

The sum in the middle is over all X with the property (P), and the first equality follows from the bijection between forests A' and sets X with property (P). For the second equality, we use property (P) and the distributive law. We obtain (10.3) by taking the sum of equations (10.4) over all $A \in \mathcal{F}(G)$.

For the moreover part of the lemma, we set $w'_e = w$ for all $e \in E(G)$ and $z_v = -1$ for all $v \in V(G)$, and observe

$$F(G'; w') = \sum_{A \in \mathcal{F}(G)} w^{|A|} \prod_{U \in \mathcal{C}(A)} (1 - |U|)\,.$$

The coefficient of $w^{n/2}$ in $F(G')$ satisfies

$$[w^{n/2}]F(G') = \sum_{\substack{A \in \mathcal{F}(G) \\ |A| = n/2}} \prod_{U \in \mathcal{C}(A)} (1 - |U|)\,. \tag{10.5}$$

If $(V(G), A)$ is an acyclic graph with exactly $n/2$ edges, then either it is a perfect matching or it contains an isolated vertex. If it contains an isolated vertex v, then $\{v\} \in \mathcal{C}(A)$ and thus the product in (10.5) is equal to zero for this particular A. It follows that A does not contribute to the sum if it is not a perfect matching. On the other hand, if A is a perfect matching, then $|U| = 2$ holds for all $U \in \mathcal{C}(A)$, so the product in (10.5) is equal to 1 or -1, depending on the parity of $n/2$. Overall, we obtain that $[w^{n/2}]F(G')$ is equal in absolute value to the number of perfect matchings of G. □

Lemma 10.2.2 shows that computing the coefficients of the multivariate forest polynomial is at least as hard as counting perfect matchings; moreover, this is true even if at most two different edge weights are used. Next we reduce from the multivariate forest polynomial with at most two distinct weights to the problem of evaluating the univariate polynomial in multigraphs. We do so via a so-called oracle serf-reduction, whose queries are sparse multigraphs in which each edge has at most a constant number of parallel edges.

Lemma 10.2.3 (From two weights to small weights)**.**

Let x and y be two variables, and let $z \in \mathbb{R} \setminus \{0\}$ be fixed. There is an algorithm as follows:

1. *Its input is a weighted graph (G, w) with $w_e \in \{x, y\}$ for all $e \in E(G)$, and a real $\varepsilon > 0$.*
2. *It outputs all coefficients of $F(G; w)$, which is a bivariate polynomial in x and y.*
3. *It runs in time $\exp(\varepsilon|E(G)|)$.*
4. *It has access to an oracle that, given w', returns the real number $F(G; w')$.*
5. *There is a constant $C_\varepsilon \in \mathbb{N}$ that only depends on ε such that all oracle queries w' made by the algorithm satisfy $w'_e \in \{0 \cdot z, \ldots, C_\varepsilon \cdot z\}$ for every edge $e \in E(G)$.*

Proof. Let (G, w) with $w_e \in \{x, y\}$ for all $e \in E(G)$ and $\varepsilon > 0$ be given as input. Define $C_\varepsilon \in \mathbb{N}$ as a large enough constant to be determined later, and let m be the smallest multiple of C_ε that is at least $|E(G)|$. This implies $|E(G)| \leq m < |E(G)| + C_\varepsilon$.

By the definition (9.1) of the multivariate forest polynomial, $F(G; w)$ is a bivariate polynomial over the variables x and y and with total degree at most m. We partition the edge set $E(G)$ into blocks of size at most C_ε in such a way that edges e and e' with $w_e \neq w_{e'}$ never belong to the same block. Since x and y occur at most m times, the number of blocks is at most m/C_ε. For each block, we introduce new variables x_i and y_i, and based on the edge partition, we obtain a new weight function v with the following properties: If $w_e = x$, then $v_e \in \{x_1, \ldots, x_{m/C_\varepsilon}\}$. If $w_e = y$, then $v_e \in \{y_1, \ldots, y_{m/C_\varepsilon}\}$. And each x_i and y_i occurs as v_e for at most C_ε different edges e. With these weights, the multivariate forest polynomial $F(G; v)$ is a polynomial p over the $2m/C_\varepsilon$ variables $\{ x_i, y_i : 1 \leq i \leq m/C_\varepsilon \}$ and each variable has individual degree at most C_ε. Moreover, the polynomial $F(G; w)$ can be recovered from $F(G; v)$ by replacing each x_i with x and each y_i with y.

The goal of the desired algorithm is to compute the *coefficients* of the bivariate polynomial $F(G; w)$, and it is able to query the *values* $F(G; w')$ for any real vector w'. Since p is a polynomial with

$$p(x_1, \ldots, x_{m/C_\varepsilon}, y_1, \ldots, y_{m/C_\varepsilon}) = F(G; v)\,,$$

the oracle allows us to query values $p(\xi)$ for real vectors $\xi \in \mathbb{R}^{2m/C_\varepsilon}$. The resulting vectors w' that we query satisfy $w'_e = \xi_j$ if e belongs to block j of the partition. The algorithm is as follows:

1. Given: (G, w) and $\varepsilon > 0$.
2. Construct a vector v over the variables x_i and y_i as discussed. [Now $F(G; v)$ is a polynomial p in $2m/C_\varepsilon$ variables and with individual degree at most C_ε.]
3. For all points $\xi \in \{0 \cdot z, \ldots, C_\varepsilon \cdot z\}^{2m/C_\varepsilon}$, use the oracle to obtain the value $p(\xi)$.
4. Use multivariate Lagrange interpolation to compute the coefficients of the polynomial p.
5. Replace each occurrence of x_i with x and each occurrence of y_i with y to recover the coefficients of $F(G; w)$.

First note that all resulting oracle queries $F(G; w')$ are indeed of the form that is required since each entry of ξ and thus w' is in $\{0 \cdot z, \ldots, C_\varepsilon \cdot z\}$. In step 3, we evaluate the polynomial $F(G; v)$ on all points of a $(2m/C_\varepsilon)$-dimensional grid dilated by z and with side-length $C_\varepsilon + 1$ in each dimension. The evaluations on such a grid are sufficient to perform multivariate Lagrange interpolation (see, e.g., [Cur15]) for $F(G; v)$, which yields the coefficients of the multivariate polynomial $F(G; v)$ and thus of the bivariate polynomial $F(G; w)$.

The running time of the algorithm is polynomial in the size $(C_\varepsilon + 1)^{2m/C_\varepsilon}$ of the grid. Now, choose C_ε large enough, depending on ε, such that this is at most $\exp(\varepsilon m)$. □

The graphs queried by the algorithm can be turned into multigraphs where each edge has weight exactly z as follows. If an edge e of

a graph G has weight w'_e with $w'_e = \mu_e \cdot z$ for some $\mu_e \in \mathbb{N}$, then we can simulate this weight by replacing e with μ_e parallel edges of weight z. Doing this for all edges yields a multigraph G_μ such that $F(G_\mu; z) = F(G; w')$ holds. In particular, $F(G_\mu; 1)$ is the number of forests in G_μ, so Lemma 10.2.3 for $z = 1$ can be seen as a reduction from two weights to forest counting in multigraphs where each edge has at most $O(1)$ parallel copies.

Thus, the combination of Lemma 10.2.2 and Lemma 10.2.3 shows, for all fixed $x \neq 0$, that it is hard to evaluate $F(G; x)$ for multigraphs with at most a constant number of parallel edges. Next we apply a stretch to make the graphs simple. To this end, we calculate the effect of a k-stretch on the univariate forest polynomial of a graph.

Lemma 10.2.4 (The forest polynomial under a k-stretch)**.** *Let G be an arbitrary multigraph on m edges, all having the same weight $w \in \mathbb{R}$. Let $k \geq 2$ be an integer such that the number $g_k(w)$ with*

$$g_k(w) = \frac{w^k}{(w+1)^k - w^k}$$

is well-defined. Let G' be the simple graph obtained from G by replacing every edge by a path of k edges. Then,

$$F(G'; w) = \left((w+1)^k - w^k\right)^m \cdot F\big(G; g_k(w)\big)\,.$$

Proof. Define a mapping ϕ that maps forests in G' to forests in G as follows: Given a forest A' of G', the image $\phi(A')$ contains the edge $e \in E(G)$ if and only if A' contains all the k edges of G' that e got stretched into. These edges then form a forest in G. That is, subgraphs A' that only differ by edges in "incomplete paths" are mapped to the same multigraph by ϕ.

Clearly, ϕ partitions $\mathcal{F}(G')$ into sets of forests with the same image under ϕ. Let A be a forest in G, and let us describe a way to generate all A' with $\phi(A') = A$. First, for each $e \in A$, add its corresponding path in G' of length k to A'. Moreover, for each edge $e \in E(G) \setminus A$, we can add to A' any proper subset of edges from the k-path in G' that corresponds

to e. Therefore, at each $e \in E(G) \setminus A$ independently, there are $\binom{k}{i}$ ways to extend A' by i edges to a forest in G', for $i \in \{0, \ldots, k-1\}$. A forest A' can be obtained in this fashion if and only if $\phi(A') = A$ holds.

For a fixed A, let us consider all summands $w^{|A'|}$ in $F(G'; w)$ with $\phi(A') = A$. By the above considerations, the total weight contribution of these summands is

$$w^{k|A|} \cdot \left(\sum_{i=0}^{k-1} \binom{k}{i} w^i \right)^{m-|A|} = w^{k|A|} \cdot \Big((w+1)^k - w^k \Big)^{m-|A|}$$

by the binomial theorem. These remarks justify the following calculation for the forest polynomial:

$$\begin{aligned} F(G'; w) &= \sum_{A \in \mathcal{F}(G)} \sum_{\substack{A' \in \mathcal{F}(G') \\ \phi(A') = A}} w^{|A'|} = \sum_{A \in \mathcal{F}(G)} w^{k \cdot |A|} \cdot \Big((w+1)^k - w^k \Big)^{m-|A|} \\ &= \Big((w+1)^k - w^k \Big)^m \cdot \sum_{A \in \mathcal{F}(G)} \left(\frac{w^k}{(w+1)^k - w^k} \right)^{|A|} . \end{aligned}$$

Since the sum in the last line is equal to $F\big(G; g_k(w)\big)$, this concludes the proof. □

We are now in position to formally prove the main theorem of this section.

Proof of Theorem 10.2.1. Let $x \in \mathbb{R} \setminus \{1\}$, and let $t = (x-1)^{-1}$. Suppose that, for all $\varepsilon > 0$, there exists an algorithm B_ε to compute the mapping $G \mapsto T(G; x, 1)$ in time $2^{\varepsilon n}$ for given simple graphs G with at most $C'_\varepsilon n$ edges, where C'_ε will be chosen later. By (9.2), algorithm B_ε can be used to compute values $F(G; (x-1)^{-1})$ with no relevant overhead in the running time. Given such an algorithm (or family of algorithms), we devise a similar algorithm for counting perfect matchings, which together with Theorem 10.1.5 implies that #ETH is false.

Let G be a simple n-vertex graph with at most Cn edges. Let G' be the graph obtained from G as in Lemma 10.2.2 by adding an apex,

labeling the edges incident to the apex with the indeterminate z, and all other edges with the indeterminate w. By Lemma 10.2.2, the coefficients of the corresponding bivariate forest polynomial of G' are sufficient to extract the number of perfect matchings of G, so it remains to compute these coefficients.

To obtain the coefficients, we use Lemma 10.2.3, keeping in mind the remark following its proof. The reduction guaranteed by the lemma produces $2^{\varepsilon m}$ multigraphs H, all with the same vertex set $V(G')$. Moreover, each H has at most $C_\varepsilon|E(G')| = C_\varepsilon(|E(G)| + n) \leq O(C_\varepsilon n)$ edges, and the multiplicity of each edge is at most C_ε. Finally, each edge of each H is assigned the same weight z, which we will choose later.

For each H, the reduction makes only one query, where it asks for the value $F(H; z)$. Our assumed algorithm however only works for simple graphs, so we perform a 3-stretch to obtain a simple graph H' with at most $3|E(H)| \leq O(C_\varepsilon n)$ edges. Lemma 10.2.4 allows us to efficiently compute the value $F(H; z)$ when we are given the value $F(H'; t)$ and $z = g_3(t)$ holds. Since g_k is a total function whenever k is a positive odd integer, and 3 is indeed odd, the value $g_3(t)$ is well-defined, and we set $z = g_3(t)$.

Set C'_ε large enough so that $E(H') \leq C'_\varepsilon \cdot n$ holds. Tracing back the reduction chain, we can use algorithm B_ε to compute $T(H'; x, 1)$ in time $2^{\varepsilon n}$ any $\varepsilon > 0$. Using (9.2), we get the value of $F(H'; t)$ since $x \neq 1$. This, in turn, yields the value of $F(H; z)$ since $(z+1)^k - z^k \neq 0$ and $g_3(t) = z$. We do this for each of the $2^{\varepsilon m}$ queries H that the reduction in Lemma 10.2.3 makes. Finally, the latter reduction outputs the coefficients of the bivariate forest polynomial of G', which contains the information on the number of perfect matchings of G.

To conclude, assuming the existence of the algorithm family $(B_\varepsilon)_{\varepsilon>0}$, we are able to count perfect matchings in time $\text{poly}(2^{\varepsilon m})$ for all $\varepsilon > 0$, which implies via Theorem 10.1.5 that #ETH is false. □

Note that the construction from the proof of Theorem 10.1.3 implies hardness of $T(G; x, 1)$ for tripartite G, and also in the bipartite case whenever $x \neq -1$.

11. The Parameterized Complexity of Counting Forests

11.1. Introduction

Parameterized counting complexity has produced results on the hardness of computing the number of paths, cliques and cycles with k edges in a given graph. One important step was the proof of #W[1]-hardness for computing the number of k-matchings in a simple graph [Cur13]. This line of research culminated in a classification theorem of Curticapean and Marx [CM14a] for the following problem: Given a graph H from a class of graphs $\mathcal{H}$ and an arbitrary graph G, compute the number of all subgraphs of G that are isomorphic to H, parameterized by $|V(H)|$. They proved that this problem is fixed-parameter tractable if the vertex cover number of all graphs in $\mathcal{H}$ is bounded by a constant[1], and #W[1]-hard otherwise.

This theorem does not cover the problem of counting all occurrences of *all* subgraphs of a certain size that are contained in a fixed class $\mathcal{H}$ of graphs. For example, using their theorem, we can classify the problem of counting all k-cliques in a graph as #W[1]-hard as follows: For the class $\mathcal{H}$, we take $\mathcal{H} = \{K_n \mid n \in \mathbb{N}\}$. As n goes to infinity, so does the vertex cover number of K_n, and the hardness follows. Of course, we might take $\mathcal{H}$ to be the set of all trees or all forests, but then the theorem speaks about the complexity of computing the number of *one*

[1] That is, $\sup\{\tau(H) \mid H \in \mathcal{H}\} < \infty$, where $\tau(H)$ is the size of a minimum vertex cover of H.

specific tree or forest in some given graph, instead of counting *all* trees or forests of a given size in one specific given graph. It is the complexity of the problem of counting all forests with a given number of edges in a given graph that we are concerned with in this chapter.[2]

Another problem that has yet escaped a parameterized analysis is the problem of counting bases in matroids. Matroids have been studied over decades and play a central role in numerous combinatorial applications (see e.g. [Oxl92]). Although they were treated in the parameterized world (see e.g. [Cyg+15a, Chap. 12] for an overview), the problem of computing the number of bases was only addressed from the classical point of view so far [Ver98; Sno12; Mau76]. It should be noted that this problem comprises also a generalization of counting forests in a graph, which gives the connection to the previously mentioned problems. Building on our results on counting forests, this gap in knowledge is one we address in the subsequent sections.

11.1.1. Related Work

It is known that computing the number of all (labeled) trees and computing the number of all forests are #P-hard problems, even on planar graphs [Jer94; VW92; GO09]. A general theorem of Eppstein implies their being fixed-parameter tractable on planar graphs [Epp02].

The problem of counting k-independent sets in a binary matroid is #P-hard. This follows from the well-known fact that the k-forests of a graph G correspond one-to-one to the k-independent sets of the binary matroid represented by the incidence matrix of G over GF(2) (see e.g. [Sno12]). Also, counting the bases of a binary matroid is #P-hard (see e.g. [Ver98]). On the other hand, the number of bases of a regular matroid can be computed in polynomial time [Mau76].

[2] Interestingly, the recent work of Roth [Rot17] covers a broad variety of questions of similar flavor, but cannot be applied to the case of counting forests with a given number of *edges*, either.

11.2. Parameterized Counting Complexity

We begin with basic definitions of parameterized counting complexity, following closely Chapt. 14 of the textbook [FG06], which we recommend to the interested reader for a more comprehensive overview of the topic. Our fundamental object of study is the following. A *parameterized counting problem* (F, k) consists of a function $F : \{0,1\}^* \to \mathbb{N}$ and a polynomial-time computable function $k : \{0,1\}^* \to \mathbb{N}$, called the *parameterization*.

A parameterized counting problem (F, k) is called *fixed-parameter tractable* if there is an algorithm A for computing F, a constant $c > 0$ and a computable function $f : \mathbb{N} \to \mathbb{N}$ such that A is running in time $f(k(x)) \cdot |x|^c$ for all $x \in \{0,1\}^*$. We say that such an algorithm runs in *fpt-time*. Let (F, k) and (F', k') be two parameterized counting problems. Then, a function $R : \{0,1\}^* \to \{0,1\}^*$ is called an *fpt parsimonious reduction from* (F, k) *to* (F', k') if

1. For all $x \in \{0,1\}^*$, $F(x) = F'(R(x))$.
2. R runs in fpt-time.
3. There is some computable $g : \mathbb{N} \to \mathbb{N}$ such that $k'(R(x)) \leq g(k(x))$ for all $x \in \{0,1\}^*$.

An algorithm A with oracle access to F' is called an *fpt Turing reduction from* (F, k) *to* (F', k') if

1. A computes F.
2. A runs in fpt-time.
3. There is some computable $g : \mathbb{N} \to \mathbb{N}$ such that for all $x \in \{0,1\}^*$ and for all instances y for which the oracle is queried during the execution of $A(x)$, $k'(y) \leq g(k(x))$.

The parameterized counting problem #k-Clique is defined as follows, and is parameterized by k: Given a graph G and an integer k, compute the number of cliques of size k in G. The class #W[1] is defined as the set of all parameterized counting problems (F, k) such that there is an

fpt parsimonious reduction from (F, k) to $\#k$-Clique. A parameterized counting problem (F, k) is called $\#\mathsf{W}[1]$-*hard* if there is an fpt Turing reduction from $\#k$-Clique to (F, k).

The following was established by Curticapean.

Theorem 11.2.1 ([Cur13])**.** *Given a graph G and a parameter k, it is $\#\mathsf{W}[1]$-hard to compute the number of matchings of size k in G.*

11.3. Matroids and Matrices

Given a graph G with vertices $v_1, \ldots, v_n$ and edges $e_1, \ldots, e_m$ we define the (unoriented) *incidence matrix* $M[G] \in \mathrm{Mat}(n \times m, \mathrm{GF}(2))$ of G by $M[G](i, j) := 1$ if $v_i \in e_j$ and 0 otherwise. A subset of columns of $M[G]$ is linearly independent (over $\mathrm{GF}(2)$) if and only if the corresponding edges form a k-forest in G.

A *matroid* is a pair $M = (E, \mathcal{I})$ consisting of a finite ground set E and a family $\mathcal{I} \neq \emptyset$ of subsets of E that satisfies the following axioms:

1. $\mathcal{I}$ is downward closed, *i.e.*, if $I \in \mathcal{I}$ and $I' \subset I$, then $I' \in \mathcal{I}$.
2. $\mathcal{I}$ has the exchange property, *i.e.*, if $I_1, I_2 \in \mathcal{I}$ and $|I_1| < |I_2|$, then there is some $e \in I_2 - I_1$ such that $I_1 \cup \{e\} \in \mathcal{I}$.

Note that this entails $\emptyset \in \mathcal{I}$. The elements of $\mathcal{I}$ are called *independent sets*, and an inclusion-wise maximal element of $\mathcal{I}$ is called a *basis* of M. The exchange property warrants that all bases have the same cardinality, and we call this cardinality the *rank* of M, written as $\mathrm{rk}\, M$. Furthermore we define $(|E| - \mathrm{rk}\, M)$ as the *nullity* of M. The pair $M_k = (E, \mathcal{I}_k)$ with $\mathcal{I}_k = \{I \in \mathcal{I} \mid |I| \leq k\}$ is again a matroid, called the *k-truncation* M_k of M.

For a field F, a *representation of M over F* is a mapping $\rho : E \to V$, where V is a vector space over F, such that for all $A \subseteq E$, A is independent if and only if $\rho(A)$ is linearly independent. M is called *representable* if it is representable over some field. If there is such a representation, we call M *representable over F*, or *F-linear*, and it holds that $\mathrm{rk}(\rho) = \mathrm{rk}(M)$. Conversely, every matrix over a field F induces an

F-linear matroid on its columns, where sets of columns are independent if and only if they are linearly independent. A *k-truncation* of a matrix is the matrix of a representation of the k-truncation of the corresponding linear matroid, possibly over a different field. Recently, Lokshtanov *et al.* proved that a k-truncation of a matrix can be computed in deterministic polynomial time (see [Lok+15], Theorem 3.23).[3] In the following we will write $\mathrm{GF}(q)$ for the field with q elements.

Given a matroid $M = (E, \mathcal{I})$, the *dual matroid* M^* of M is a matroid on the same ground set as M, and $B \subseteq E$ is a basis of M^* if and only if $E \setminus B$ is a basis of M. Given a representation of a matroid M, a representation of M^* in the same field can be found in polynomial time (see e.g. [Mar09b]).

In the following, *all* matroids will be assumed to be representable, and encoded using a representing matrix ρ and a suitable encoding for the ground field. Furthermore, we can, without loss of generality, always assume that ρ has $\mathrm{rk}(M)$ rows, because row operations (multiplying a row by a non-zero scalar, and adding such multiples to other rows) do not affect linear independence of the columns of ρ. Hence, any ρ' obtained from ρ through row operations is a representation of M. In particular, by Gaussian elimination, we may assume all but the first $\mathrm{rk}(M)$ rows of ρ to be zero.

11.4. Counting Forests is #W[1]-hard

In this section, we will prove that counting k-forests is #W[1]-hard. This result will be used to show hardness of counting matroid bases in fields of fixed characteristic.

For a forest A in G, let again $\mathcal{C}(A)$ be the family of all sets $T \subseteq V(G)$ such that $T \neq \emptyset$ and T is a maximal connected component in A. Adding an apex, that is, a new vertex that is connected to all other vertices, to a graph $G = (V, E)$ and labeling each of the new edges with a new variable z makes the univariate forest polynomial into a bivariate one,

[3] A slightly weaker version of this result with a simpler proof that still suffices for our application seems to follow along the lines of Snook [Sno12].

namely $F(G';x,z)$, where G' is the described graph with an added apex. In the following, G will always be the original graph, and G' will be the graph obtained in this way.

Note that $F(G';x,z) \in \mathbb{Z}[x,z] \cong (\mathbb{Z}[z])[x]$. In particular, the coefficient of x^k in $F(G';x,z)$ is an element of $\mathbb{Z}[z]$. To make this very clear in the following, we shall refer to this element of $\mathbb{Z}[z]$ as the *coefficient polynomial* of x^k in $F(G';x,z)$.

Lemma 11.4.1. *There is a polynomial-time Turing reduction from counting matchings of size k in a graph G to computing the coefficient polynomial of x^k of the bivariate forest polynomial $F(G';x,z)$ of the graph G', parameterized by k in both problems. In particular, this reduction retains the parameter k and is thus even an fpt Turing reduction.*

Proof. The coefficient polynomial $C_k(z) \in \mathbb{Z}[z]$ of x^k in $F(G';x,z)$ can be expressed in terms of G through

$$C_k(z) = \sum_{A \in \binom{E}{k} \text{ acyclic in } G} \prod_{T \in \mathcal{C}(A)} (1 + |T|z),$$

as follows immediately from Lemma 10.2.2 after specializing to x and z. Since a forest in G with k edges can cover at most $2k$ nodes of G, at least $n - 2k$ nodes of G are left uncovered by T, and are thus present in $\mathcal{C}(A)$ as components T with $|T| = 1$. This shows that the product $\prod_{T \in \mathcal{C}(A)}(1 + |T|z)$ of each summand (and hence also $C_k(z)$) is a multiple of $(1+z)^{n-2k}$. Thus, the polynomial quotient

$$Q_k(z) := C_k(z)/(1+z)^{n-2k}$$

is a well-defined element of $\mathbb{Z}[z]$.

Observe that it is precisely the k-matchings of G that will have $2k$ covered and $n - 2k$ uncovered nodes, and such a k-matching has k components with $|T| = 2$, and $n - 2k$ components with $|T| = 1$. Therefore, the summand corresponding to some A in $C_k(z)$ is of the form $(1+z)^{n-2k}(1+2z)^k$ if and only if A is a k-matching. Likewise, the number of k-matchings is precisely the number of such monomials in

$C_k(z)$. In all other monomials, the factor $(1+z)$ is hence present with degree at least $n-2k+1$. Denote with M_k the number of k-matchings in G. After substituting $z \mapsto y-1$ (hence, $y = z+1$), this can be stated as follows:

$$Q_k(y) = C_k(y)/y^{n-2k} = M_k \cdot (2y-1)^k + y \cdot R(y)$$

for some polynomial $R(y)$. We see that for $y = 0$, $y \cdot R(y) = 0$, and hence, keeping in mind that $z = y-1$, it follows

$$Q_k(y=0) = Q_k(z=-1) = M_k \cdot (-1)^k .$$

We now argue why this is an fpt Turing reduction. Note that in the coordinates y^i, the polynomial division is just a shift of coefficients. Therefore, an oracle to the k-th coefficient polynomial of $F(G';x,z)$, as provided in a Turing reduction, yields the polynomial $C_k(z)$. After a change of basis from z to $y-1$ and a corresponding shift of coefficients to perform the division by y^{n-2k}, we can evaluate the resulting polynomial $Q_k(y)$ at $y = 0$ and obtain $M_k \cdot (-1)^k$ and thus M_k. This can clearly be done in polynomial time in the size of G (and k, for that matter) once $C_k(z)$ was obtained, and the only oracle query involved does not alter the parameter and is hence valid for an fpt Turing reduction. □

Note that this implies Lemma 10.2.2 as a special case, where $k = n/2$.

This proves that the coefficient polynomial of x^k in the bivariate polynomial $F(G';x,z)$ is hard to compute. We now want to show that this implies that the k-th coefficient (which is a natural number, not a polynomial) of the univariate polynomial is hard to compute. We do this by reducing the compution the coefficient polynomial of x^k in $F(G';x,z)$ to computing the k-th coefficient in a suitable univariate forest polynomial.

We first show that, although the degree of the coefficient polynomial $C_k(z)$ (in the bivariate case) is not bounded by $f(k)$, but $\Omega(n)$, it suffices to know $O(k)$ coefficients of the coefficient polynomial $C_k(z)$ in order to reconstruct the whole coefficient polynomial. This is an easy application of the Chinese Remainder Theorem for polynomials.

Lemma 11.4.2. *There is an fpt Turing reduction from computing the coefficient polynomial $C_k(z)$ of x^k in $F(G'; x, z)$ to computing the first k coefficients of univariate forest polynomials on multigraphs.*

Combining the above proves:

Theorem 11.4.3. *Given a graph G and a number k, it is $\#\mathsf{W}[1]$-hard to compute the number of acyclic subsets of edges of size k in G, parameterized by k.*

Proof. Combining Theorem 11.2.1 with Lemma 11.4.1 and Lemma 11.4.2 yields that computing the first k coefficients of the univariate forest polynomial of multigraphs is $\#\mathsf{W}[1]$-hard. Using Lemma 10.2.4 allows to express the forest polynomial of the multigraph G as $F(G; x) = p(x) \cdot F(G'; g(x))$, where $p, g : \mathbb{R} \to \mathbb{R}$ are functions such that g is invertible and G' is a simple graph. Now, observe that computing the mapping $G \mapsto F_k(G; x)$ is $\#\mathsf{W}[1]$-hard for each fixed x, where $F_k(G; x)$ is the forest polynomial $F(G; x)$ evaluated at x only over the first k coefficients: It is easy to see that $F_k(G; ax) = F_k(G(a); x)$, where G is the graph obtained from G by replacing each edge with a copies of weight x. By using k different values for a, this would allow polynomial interpolation of the first k coefficients of $F(G; x)$. Employing now the equation $F_k(G; x) = p(x) \cdot F_k(G'; g(x))$ and the properties of g, this shows that computing the mapping $G' \mapsto F_k(G'; x)$ on *simple graphs* G' is $\#\mathsf{W}[1]$-hard for all x, and hence also for $x = 1$, where it coincides with counting forests.

□

11.5. Counting Matroid Bases is $\#\mathsf{W}[1]$-hard

Definition 11.5.1. The problem of computing the number of bases of a matroid parameterized by its rank (nullity) is denoted as #Rank-Bases (#Nullity-Bases).

Lemma 11.5.2. *The problem of counting k-forests in a simple graph is fpt Turing reducible to the problem* #Rank-Bases, *even when the matroid is restricted to be representable over a field of characteristic* 2.

Proof. Given a graph $G = (V, E)$ with $|V| = n$ and $|E| = m$ and a natural number k, we want to count the k-forests of G. Therefore we first construct the incidence matrix $M[G] \in \mathrm{Mat}(n \times m, \mathrm{GF}(2))$ of G. Recall that the linearly independent k-subsets of columns of $M[G]$ correspond one-to-one to k-forests in G. In the next step, we compute the reduced row echelon form of $M[G]$ by applying elementary row operations. As stated in the beginning, these operations do not change the linear dependency of the column vectors. Then, we delete the zero rows which also does not change the linear dependency of the columns. We denote the resulting matrix as $M^{\mathrm{red}}[G]$. Now, let r be the rank of $M^{\mathrm{red}}[G]$, which equals the rank of $M[G]$. Note that $M^{\mathrm{red}}[G] \in \mathrm{Mat}(r \times m, \mathrm{GF}(2))$. If $r < k$, then we output 0, as G does not have any k-forests in this case. Otherwise, we k-truncate $M^{\mathrm{red}}[G]$ in polynomial time by the deterministic algorithm of Lokshtanov *et al.* [Lok+15] and end up with the matrix $M^{(k)}[G] \in \mathrm{Mat}(k \times m, \mathrm{GF}(2^{rk}))$. Observe that the linear dependency of the column vectors is preserved, *i.e.*, whenever columns $c_1, \ldots, c_k$ are linearly independent in $M[G]$, they are also linearly independent in $M^{(k)}[G]$ and vice versa. Therefore, the rank of $M^{(k)}[G]$ is at least k, since $M[G]$ has rank greater or equal k. As $M^{(k)}[G]$ has only k rows, it follows that the rank is *exactly* k, *i.e.*, $M^{(k)}[G]$ has full rank. Furthermore, the number of linearly independent k-subsets of columns of $M^{(k)}[G]$ equals the number of k-forests in G. As the rank of $M^{(k)}[G]$ is full, we conclude that the number of bases of the matroid that is represented by $M^{(k)}[G]$ equals the number of k-forests in G. Finally, this matroid is representable over $\mathrm{GF}(2^{rk})$—a field of characteristic 2—by construction. □

Lemma 11.5.3. *The problem of counting k-forests in a simple graph is fpt Turing-reducible to the problem* #Nullity-Bases, *even when the matroid is restricted to be representable over a field of characteristic* 2.

Proof. We proceed as in the proof of Lemma 11.5.2. Having $M^{(k)}[G]$,

we construct its dual matroid $M^*[G]$, which can be done in polynomial time (see e.g. [Mar09b]). It holds that the number of bases of $M^*[G]$ equals the number of bases of $M^{(k)}[G]$. Furthermore, the rank of $M^*[G]$ is $n - k$, *i.e.*, its nullity is k, which concludes the proof. □

Theorem 11.5.4. #Rank-Bases *and* #Nullity-Bases *are* #W[1]*-hard, even when restricted to matroids representable over a field of characteristic* 2.

Proof. Follows from Lemma 11.5.2, Lemma 11.5.3 and Theorem 11.4.3. □

One might ask whether the same is true for matroids that are representable over a fixed finite field. Due to Vertigan [Ver98], it is known that the classical problem of counting bases in binary matroids is #P-hard. However, it is fixed-parameter tractable for each fixed finite field.

Theorem 11.5.5. *For every fixed finite field F, the problems* #Rank-Bases *and* #Nullity-Bases *are fixed parameter tractable for matroids given in a linear representation over F.*

of Theorem 11.5.5. We give an fpt algorithm for #Rank-Bases. For #Nullity-Bases, an algorithm follows by computing the dual matroid as in the proof of Lemma 11.5.3.

Let s be the size of the finite field, M be the representation of the given matroid and let k be its rank. We can assume that M only has k rows. For otherwise, we can compute the reduced row echelon form and delete zero rows, which does not change the linear dependencies of the column vectors. If M has only k rows, then there are at most s^k different column vectors. Therefore, we remember the muliplicity of every column vector and delete multiple occurences afterwards. We end up with a matrix with at most s^k columns. Then, we can check for every k-subset of columns whether they are linearly independent. If this is the case, we just multiply the multiplicities of the columns and in the end, we output the sum of all those terms. The running time of this procedure is bounded by $\binom{s^k}{k} \cdot \text{poly}(n)$, where n is the number of columns of the matrix. □

Bibliography

[AG09] Noga Alon and Shai Gutner. "Balanced Hashing, Color Coding and Approximate Counting". In: *Parameterized and Exact Computation, 4th International Workshop, IWPEC 2009, Copenhagen, Denmark, September 10-11, 2009, Revised Selected Papers*. Ed. by Jianer Chen and Fedor V. Fomin. Vol. 5917. Lecture Notes in Computer Science. Springer, 2009, pp. 1–16. DOI: 10.1007/978-3-642-11269-0_1.

[AG10] Noga Alon and Shai Gutner. "Balanced families of perfect hash functions and their applications". In: *ACM T. Algorithms* 6.3 (2010), 54:1–54:12. DOI: 10.1145/1798596.1798607.

[AKK18] Per Austrin, Petteri Kaski, and Kaie Kubjas. "Tensor Network Complexity of Multilinear Maps". In: *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*. Ed. by Avrim Blum. Vol. 124. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 7:1–7:21. ISBN: 978-3-95977-095-8. DOI: 10.4230/LIPIcs.ITCS.2019.7. URL: http://drops.dagstuhl.de/opus/volltexte/2018/10100.

[Alo+08] Noga Alon, Phuong Dao, Iman Hajirasouliha, Fereydoun Hormozdiari, and S Cenk Sahinalp. "Biomolecular network motif counting and discovery by color coding". In: *Bioinformatics* 24.13 (2008), pp. i241–i249. DOI: 10.1093/bioinformatics/btn163.

[AR02] Vikraman Arvind and Venkatesh Raman. "Approximation Algorithms for Some Parameterized Counting Problems". In: *Algorithms and Computation, 13th International Symposium, ISAAC 2002 Vancouver, BC, Canada, November 21-23, 2002, Proceedings*. Ed. by Prosenjit Bose and Pat Morin. Vol. 2518. Lecture Notes in Computer Science. Springer, 2002, pp. 453–464. DOI: 10.1007/3-540-36136-7_40.

[Arv+18a] Vikraman Arvind, Abhranil Chatterjee, Rajit Datta, and Partha Mukhopadhyay. "Fast Exact Algorithms Using Hadamard Product of Polynomials". In: *CoRR* abs/1807.04496 (2018). arXiv: 1807.04496. URL: http://arxiv.org/abs/1807.04496.

[Arv+18b] Vikraman Arvind, Abhranil Chatterjee, Rajit Datta, and Partha Mukhopadhyay. "Univariate Ideal Membership Parameterized by Rank, Degree, and Number of Generators". In: *38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2018, December 11-13, 2018, Ahmedabad, India*. 2018, 7:1–7:18. DOI: 10.4230/LIPIcs.FSTTCS.2018.7. URL: https://doi.org/10.4230/LIPIcs.FSTTCS.2018.7.

[AYZ94] Noga Alon, Raphael Yuster, and Uri Zwick. "Color-coding: a new method for finding simple paths, cycles and other small subgraphs within large graphs". In: *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*. 1994, pp. 326–335. DOI: 10.1145/195058.195179. URL: https://doi.org/10.1145/195058.195179.

[AYZ95] Noga Alon, Raphael Yuster, and Uri Zwick. "Color-Coding". In: *J. ACM* 42.4 (1995), pp. 844–856. DOI: 10.1145/210332.210337.

[Bab16] László Babai. "Graph isomorphism in quasipolynomial time [extended abstract]". In: *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*. Ed. by Daniel Wichs and Yishay Mansour. ACM, 2016, pp. 684–697. DOI: 10.1145/2897518.2897542.

[BCS97] Peter Bürgisser, Michael Clausen, and Mohammad Amin Shokrollahi. *Algebraic complexity theory*. Vol. 315. Grundlehren der mathematischen Wissenschaften. Springer, 1997. ISBN: 3-540-60582-7.

[BDH18] Cornelius Brand, Holger Dell, and Thore Husfeldt. "Extensor-coding". In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2018. Los Angeles, CA, USA: ACM, 2018, pp. 151–164. ISBN: 978-1-4503-5559-9. DOI: 10.1145/3188745.3188902. URL: http://doi.acm.org/10.1145/3188745.3188902.

[BDR16] Cornelius Brand, Holger Dell, and Marc Roth. "Fine-Grained Dichotomies for the Tutte Plane and Boolean #CSP". In: *11th International Symposium on Parameterized and Exact Computation, IPEC 2016, August 24-26, 2016, Aarhus, Denmark*. 2016, 9:1–9:14. DOI: 10.4230/LIPIcs.IPEC.2016.9. URL: https://doi.org/10.4230/LIPIcs.IPEC.2016.9.

[BDR19] Cornelius Brand, Holger Dell, and Marc Roth. "Fine-Grained Dichotomies for the Tutte Plane and Boolean #CSP". In: *Algorithmica* 81.2 (2019), pp. 541–556. DOI: 10.1007/s00453-018-0472-z. URL: https://doi.org/10.1007/s00453-018-0472-z.

[Bjö+07] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. "Fourier meets Möbius: Fast subset convolution". In: *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*. 2007, pp. 67–74. DOI: 10.1145/1250790.1250801.

[Bjö+08] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. "Computing the Tutte polynomial in vertex-exponential time". In: *Proceedings of the 47th annual IEEE Symposium on Foundations of Computer Science, FOCS 2008*. 2008, pp. 677–686. DOI: 10.1109/FOCS.2008.40.

[Bjö+10] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. "Trimmed Moebius Inversion and Graphs of Bounded Degree". In: *Theory Comput. Syst.* 47.3 (2010), pp. 637–654. DOI: 10.1007/s00224-009-9185-7. URL: https://doi.org/10.1007/s00224-009-9185-7.

[Bjö+17a] 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.

[Bjö+17b] Andreas Björklund, Vikram Kamat, Lukasz Kowalik, and Meirav Zehavi. "Spotting Trees with Few Leaves". In: *SIAM J. Discrete Math.* 31.2 (2017), pp. 687–713. DOI: 10.1137/15M1048975. URL: https://doi.org/10.1137/15M1048975.

[Bjö+19] Andreas Björklund, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. "Approximate Counting of k-Paths: Deterministic and in Polynomial Space". In: *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*. Ed. by Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi. Vol. 132. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, 24:1–24:15. ISBN: 978-3-95977-109-2. DOI: 10.4230/LIPIcs.ICALP.2019.24. URL: http://drops.dagstuhl.de/opus/volltexte/2019/10600.

[Bjö12] Andreas Björklund. "Counting perfect matchings as fast as Ryser". In: *Proceedings of the 23rd Symposium on Discrete Algorithms, SODA 2012*. 2012, pp. 914–921. DOI: 10.1137/1.9781611973099.73.

[Bjö14] Andreas Björklund. "Determinant Sums for Undirected Hamiltonicity". In: *SIAM J. Comput.* 43.1 (2014), pp. 280–299. DOI: 10.1137/110839229.

[BK95] Manuel Blum and Sampath Kannan. “Designing Programs That Check Their Work”. In: *J. ACM* 42.1 (Jan. 1995), pp. 269–291. ISSN: 0004-5411. DOI: 10.1145/200836.200880. URL: http://doi.acm.org/10.1145/200836.200880.

[BKK17] Andreas Björklund, Petteri Kaski, and Ioannis Koutis. “Directed Hamiltonicity and Out-Branchings via Generalized Laplacians”. In: *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*. 2017, 91:1–91:14. DOI: 10.4230/LIPIcs.ICALP.2017.91. URL: https://doi.org/10.4230/LIPIcs.ICALP.2017.91.

[Blä+09] Markus Bläser, Moritz Hardt, Richard J. Lipton, and Nisheeth K. Vishnoi. “Deterministically testing sparse polynomial identities of unbounded degree”. In: *Inf. Process. Lett.* 109.3 (2009), pp. 187–192. DOI: 10.1016/j.ipl.2008.09.029. URL: https://doi.org/10.1016/j.ipl.2008.09.029.

[Bod93] Hans L. Bodlaender. “On Linear Time Minor Tests with Depth-First Search”. In: *J. Algorithms* 14.1 (1993), pp. 1–23. DOI: 10.1006/jagm.1993.1001.

[BR17] Cornelius Brand and Marc Roth. “Parameterized Counting of Trees, Forests and Matroid Bases”. In: *Computer Science - Theory and Applications - 12th International Computer Science Symposium in Russia, CSR 2017, Kazan, Russia, June 8-12, 2017, Proceedings*. 2017, pp. 85–98. DOI: 10.1007/978-3-319-58747-9_10. URL: https://doi.org/10.1007/978-3-319-58747-9%5C_10.

[Bra] Cornelius Brand. “Apolarity for Determinants of Generic Hankel Matrices”. In preparation.

[Bra19] Cornelius Brand. “Patching Colors with Tensors”. In: *27th Annual European Symposium on Algorithms, ESA 2019, Munich, Germany*. 2019. Forthcoming.

[BS16] Cornelius Brand and Michael Sagraloff. “On the Complexity of Solving Zero-Dimensional Polynomial Systems via Projection”. In: *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC 2016, Waterloo, ON, Canada, July 19-22, 2016*. 2016, pp. 151–158. DOI: 10.1145/2930889.2930934. URL: https://doi.org/10.1145/2930889.2930934.

[Bul08] Andrei A. Bulatov. “The Complexity of the Counting Constraint Satisfaction Problem”. 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. 646–661. DOI: 10.1007/978-3-540-70575-8_53.

[Bul13] Andrei A. Bulatov. "The complexity of the counting constraint satisfaction problem". In: *J. ACM* 60.5 (2013), 34:1–34:41. DOI: 10.1145/2528400.

[Bul17] Andrei A. Bulatov. *A dichotomy theorem for nonuniform CSPs*. 2017. eprint: 1703.03021.

[CDM17] Radu Curticapean, Holger Dell, and Dániel Marx. "Homomorphisms are a good basis for counting small subgraphs". In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*. Ed. by Hamed Hatami, Pierre McKenzie, and Valerie King. ACM, 2017, pp. 210–223. DOI: 10.1145/3055399.3055502.

[Che+05] Jianer Chen, Benny Chor, Mike Fellows, Xiuzhen Huang, David W. Juedes, Iyad A. Kanj, and Ge Xia. "Tight lower bounds for certain parameterized NP-hard problems". In: *Inform. Comput.* 201.2 (2005), pp. 216–231. DOI: 10.1016/j.ic.2005.05.001.

[Che+06] Jin Chen, Wynne Hsu, Mong Li Lee, and See-Kiong Ng. "NeMoFinder: Dissecting genome-wide protein-protein interactions with meso-scale network motifs". In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2006, pp. 106–115. DOI: 10.1145/1150402.1150418.

[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*. 2007, pp. 298–307. URL: http://dl.acm.org/citation.cfm?id=1283383.1283415.

[Che+09] Jianer Chen, Joachim Kneis, Songjian Lu, Daniel Mölle, Stefan Richter, Peter Rossmanith, Sing-Hoi Sze, and Fenghui Zhang. "Randomized Divide-and-Conquer: Improved Path, Matching, and Packing Algorithms". In: *SIAM J. Comput.* 38.6 (2009), pp. 2526–2547. DOI: 10.1137/080716475.

[CIP06] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. "A Duality Between Clause Width and Clause Density for SAT". In: *Proceedings of the 21st Annual IEEE Conference on Computational Complexity*. CCC '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 252–260. ISBN: 0-7695-2596-2. DOI: 10.1109/CCC.2006.6. URL: http://dx.doi.org/10.1109/CCC.2006.6.

[CM14a] Radu Curticapean and Dániel Marx. "Complexity of Counting Subgraphs: Only the Boundedness of the Vertex-Cover Number Counts". In: *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*. 2014, pp. 130–139. DOI: 10.1109/FOCS.2014.22. URL: http://dx.doi.org/10.1109/FOCS.2014.22.

[CM14b] Radu Curticapean and Dániel Marx. "Complexity of counting subgraphs: Only the boundedness of the vertex-cover number counts". In: *Proceedings of the 55th Annual Symposium on Foundations of Computer Science (FOCS)*. 2014, pp. 130–139. DOI: 10.1109/FOCS.2014.22.

[Coh+10] Nathann Cohen, Fedor V. Fomin, Gregory Z. Gutin, Eun Jung Kim, Saket Saurabh, and Anders Yeo. "Algorithm for finding k-vertex out-trees and its application to k-internal out-branching problem". In: *J. Comput. Syst. Sci.* 76.7 (2010), pp. 650–662. DOI: 10.1016/j.jcss.2010.01.001. URL: https://doi.org/10.1016/j.jcss.2010.01.001.

[Con98] Aldo Conca. "Straightening Law and Powers of Determinantal Ideals of Hankel Matrices". In: *Advances in Mathematics* 138.2 (1998), pp. 263–292. ISSN: 0001-8708. DOI: https://doi.org/10.1006/aima.1998.1740. URL: http://www.sciencedirect.com/science/article/pii/S0001870898917406.

[Coo71] Stephen A. Cook. "The Complexity of Theorem-Proving Procedures". In: *Proceedings of the 3rd Annual Symposium on Theory of Computing (STOC)*. 1971, pp. 151–158. DOI: 10.1145/800157.805047.

[Cor+09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009. ISBN: 978-0-262-03384-8. URL: http://mitpress.mit.edu/books/introduction-algorithms.

[Cur13] Radu Curticapean. "Counting Matchings of Size k Is #W[1]-Hard". In: *Automata, Languages, and Programming: 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 352–363. ISBN: 978-3-642-39206-1. DOI: 10.1007/978-3-642-39206-1_30. URL: http://dx.doi.org/10.1007/978-3-642-39206-1_30.

[Cur15] Radu Curticapean. "Block Interpolation: A Framework for Tight Exponential-Time Counting Complexity". In: *Proceedings of the 42nd International Colloquium on Automata, Languages and Programming, ICALP 2015*. Springer, 2015, pp. 380–392. DOI: 10.1007/978-3-662-47672-7_31.

[Cur18] Radu Curticapean. “Block interpolation: A framework for tight exponential-time counting complexity”. In: *Inf. Comput.* 261.Part (2018), pp. 265–280. DOI: 10.1016/j.ic.2018.02.008. URL: https://doi.org/10.1016/j.ic.2018.02.008.

[Cyg+15a] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. ISBN: 978-3-319-21274-6. DOI: 10.1007/978-3-319-21275-3. URL: http://dx.doi.org/10.1007/978-3-319-21275-3.

[Cyg+15b] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. ISBN: 978-3-319-21274-6. DOI: 10.1007/978-3-319-21275-3.

[Dal11] Jean Daligault. “Techniques combinatoires pour les algorithmes paramétrés et les noyaux, avec applications aux problèmes de multicoupe. (Combinatorial Techniques for Parameterized Algorithms and Kernels, with Applications to Multicut.)” PhD thesis. Montpellier 2 University, France, 2011. URL: https://tel.archives-ouvertes.fr/tel-00804206.

[Del+14] Holger Dell, Thore Husfeldt, Dániel Marx, Nina Taslaman, and Martin Wahlén. “Exponential Time Complexity of the Permanent and the Tutte Polynomial”. In: *ACM T. Algorithms* 10.4 (2014), 21:1–21:32. DOI: 10.1145/2635812.

[DL78] Richard A. DeMillo and Richard J. Lipton. “A Probabilistic Remark on Algebraic Program Testing”. In: *Inform. Process. Lett.* 7.4 (1978), pp. 193–195. DOI: 10.1016/0020-0190(78)90067-4.

[DPV08] Sanjoy Dasgupta, Christos H. Papadimitriou, and Umesh V. Vazirani. *Algorithms*. McGraw-Hill, 2008. ISBN: 978-0-07-352340-8.

[DST02] Josep Díaz, Maria J. Serna, and Dimitrios M. Thilikos. “Counting H-colorings of partial k-trees”. In: *Theor. Comput. Sci* 281.1-2 (2002), pp. 291–309. DOI: 10.1016/S0304-3975(02)00017-8.

[EMon] Joanna Ellis-Monaghan and Iain Moffatt. *CRC handbook on the Tutte polynomial and related topics*. CRC press, in preparation.

[Epp02] David Eppstein. “Subgraph Isomorphism in Planar Graphs and Related Problems”. In: *Graph Algorithms and Applications* (2002), p. 283.

[FG06] J. Flum and M. Grohe. *Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. ISBN: 3540299521.

[FGR09] Henning Fernau, Serge Gaspers, and Daniel Raible. "Exact and parameterized algorithms for max internal spanning tree". In: *International Workshop on Graph-Theoretic Concepts in Computer Science*. Springer. 2009, pp. 100–111.

[FKL07] Uffe Flarup, Pascal Koiran, and Laurent Lyaudet. "On the Expressive Power of Planar Perfect Matching and Permanents of Bounded Treewidth Matrices". In: *Algorithms and Computation, 18th International Symposium, ISAAC 2007, Sendai, Japan, December 17-19, 2007, Proceedings*. 2007, pp. 124–136. DOI: 10.1007/978-3-540-77120-3_13. URL: https://doi.org/10.1007/978-3-540-77120-3%5C_13.

[Fom+12a] Fedor V. Fomin, Fabrizio Grandoni, Daniel Lokshtanov, and Saket Saurabh. "Sharp Separation and Applications to Exact and Parameterized Algorithms". In: *Algorithmica* 63.3 (2012), pp. 692–706. DOI: 10.1007/s00453-011-9555-9. URL: https://doi.org/10.1007/s00453-011-9555-9.

[Fom+12b] Fedor V. Fomin, Daniel Lokshtanov, Venkatesh Raman, Saket Saurabh, and B. V. Raghavendra Rao. "Faster algorithms for finding and counting subgraphs". In: *J. Comput. Syst. Sci.* 78.3 (2012), pp. 698–706. DOI: 10.1016/j.jcss.2011.10.001.

[Fom+13] Fedor V Fomin, Serge Gaspers, Saket Saurabh, and Stéphan Thomassé. "A linear vertex kernel for maximum internal spanning tree". In: *Journal of Computer and System Sciences* 79.1 (2013), pp. 1–6.

[Fom+14] Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. "Representative Sets of Product Families". In: *Algorithms - ESA 2014 - 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*. Ed. by Andreas S. Schulz and Dorothea Wagner. Vol. 8737. Lecture Notes in Computer Science. Springer, 2014, pp. 443–454. DOI: 10.1007/978-3-662-44777-2_37.

[Fom+16] Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. "Efficient Computation of Representative Families with Applications in Parameterized and Exact Algorithms". In: *J. ACM* 63.4 (2016), 29:1–29:60. DOI: 10.1145/2886094.

[FV12] Fedor V. Fomin and Yngve Villanger. "Treewidth computation and extremal combinatorics". In: *Combinatorica* 32.3 (2012), pp. 289–308. DOI: 10.1007/s00493-012-2536-z.

[FV93] Tomás Feder and Moshe Y. Vardi. "Monotone monadic SNP and constraint satisfaction". In: *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*. Ed. by S. Rao Kosaraju, David S. Johnson, and

Alok Aggarwal. ACM, 1993, pp. 612–622. DOI: 10.1145/167088.167245.

[FV98] Tomás Feder and Moshe Y. Vardi. "The Computational Structure of Monotone Monadic SNP and Constraint Satisfaction: A Study through Datalog and Group Theory". In: *SIAM J. Comput.* 28.1 (1998), pp. 57–104. DOI: 10.1137/S0097539794266766.

[Gal14] François Le Gall. "Powers of tensors and fast matrix multiplication". In: *International Symposium on Symbolic and Algebraic Computation, ISSAC '14, Kobe, Japan, July 23-25, 2014*. 2014, pp. 296–303. DOI: 10.1145/2608628.2608664. URL: https://doi.org/10.1145/2608628.2608664.

[GG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. 3rd. Cambridge University Press, 2013. ISBN: 978-1-107-03903-2. DOI: 10.1017/CBO9781139856065.

[Gir90] Vyacheslav L. Girko. *Theory of random determinants*. Vol. 45. Mathematics and its applications. Springer, 1990. DOI: 10.1007/978-94-009-1858-0.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979. ISBN: 0716710447.

[GK07] Joshua A Grochow and Manolis Kellis. "Network motif discovery using subgraph enumeration and symmetry-breaking". In: *Annual International Conference on Research in Computational Molecular Biology*. Springer. 2007, pp. 92–106. DOI: 10.1007/978-3-540-71681-5_7.

[GM07] Geoffrey Grimmett and Colin McDiarmid, eds. *Combinatorics, Complexity, and Chance: A Tribute to Dominic Welsh*. Oxford Lecture Series in Mathematics and Its Applications (Book 34). Oxford University Press, 2007. ISBN: 978-0198571278.

[GO09] Heidi Gebauer and Yoshio Okamoto. "Fast Exponential-Time Algorithms for the Forest Counting and the Tutte Polynomial Computation in Graph Classes". In: *Int. J. Found. Comput. Sci.* 20.1 (2009), pp. 25–44. DOI: 10.1142/S0129054109006437. URL: http://dx.doi.org/10.1142/S0129054109006437.

[GRK09] Gregory Gutin, Igor Razgon, and Eun Jung Kim. "Minimum leaf out-branching and related problems". In: *Theoretical Computer Science* 410.45 (2009), pp. 4571–4579.

[GS95] Ira M. Gessel and Richard P. Stanley. "Handbook of Combinatorics (Vol. 2)". In: ed. by R. L. Graham, M. Grötschel, and L. Lovász. Cambridge, MA, USA: MIT Press, 1995. Chap. Algebraic Enumeration, pp. 1021–1061. ISBN: 0-262-07171-1. URL: http://dl.acm.org/citation.cfm?id=233228.233231.

[Gut+18] Gregory Z. Gutin, Felix Reidl, Magnus Wahlström, and Meirav Zehavi. "Designing deterministic polynomial-space algorithms by color-coding multivariate polynomials". In: *J. Comput. Syst. Sci.* 95 (2018), pp. 69–85. DOI: 10.1016/j.jcss.2018.01.004. URL: https://doi.org/10.1016/j.jcss.2018.01.004.

[HWZ08] Falk Hüffner, Sebastian Wernicke, and Thomas Zichner. "Algorithm Engineering for Color-Coding with Applications to Signaling Pathway Detection". In: *Algorithmica* 52.2 (2008), pp. 114–132. DOI: 10.1007/s00453-007-9008-7.

[IP01] Russell Impagliazzo and Ramamohan Paturi. "On the complexity of k-SAT". In: *J. Comput. Syst. Sci.* 62.2 (2001), pp. 367–375. DOI: 10.1006/jcss.2000.1727.

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. "Which problems have strongly exponential complexity?" In: *J. Comput. Syst. Sci.* 63.4 (2001), pp. 512–530. DOI: 10.1006/jcss.2001.1774.

[Jer94] Mark Jerrum. "Counting Trees in a Graph is #P-Complete". In: *Inf. Process. Lett.* 51.3 (1994), pp. 111–116. DOI: 10.1016/0020-0190(94)00085-9. URL: http://dx.doi.org/10.1016/0020-0190(94)00085-9.

[JS82] Mark Jerrum and Marc Snir. "Some Exact Complexity Results for Straight-Line Computations over Semirings". In: *J. ACM* 29.3 (1982), pp. 874–897. DOI: 10.1145/322326.322341. URL: http://doi.acm.org/10.1145/322326.322341.

[JVW90] François Jaeger, Dirk L. Vertigan, and Dominic J.A. Welsh. "On the computational complexity of the Jones and Tutte polynomials". In: *Mathematical proceedings of the Cambridge Philosophical Society* 108.1 (1990), pp. 35–53. DOI: 10.1017/S0305004100068936.

[Kas+04] Nadav Kashtan, Shalev Itzkovitz, Ron Milo, and Uri Alon. "Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs". In: *Bioinformatics* 20.11 (2004), pp. 1746–1758. DOI: 10.1093/bioinformatics/bth163.

[Kas+09] Zahra Razaghi Moghadam Kashani, Hayedeh Ahrabian, Elahe Elahi, Abbas Nowzari-Dalini, Elnaz Saberi Ansari, Sahar Asadi, Shahin Mohammadi, Falk Schreiber, and Ali Masoudi-Nejad. "Kavosh: A new algorithm for finding network motifs". In: *BMC Bioinformatics* 10.1 (2009), p. 318. DOI: 10.1186/1471-2105-10-318.

[KL83] Richard M. Karp and Michael Luby. "Monte-Carlo Algorithms for Enumeration and Reliability Problems". In: *24th Annual Symposium on Foundations of Computer Science, Tucson, Arizona, USA, 7-9 November 1983*. IEEE Computer Society, 1983, pp. 56–64. DOI: 10.1109/SFCS.1983.35.

[KLL13] Miroslaw Kowaluk, Andrzej Lingas, and Eva-Marta Lundell. "Counting and Detecting Small Subgraphs via Equations". In: *SIAM J. Discrete Math.* 27.2 (2013), pp. 892–909. DOI: 10.1137/110859798.

[Kne+06] Joachim Kneis, Daniel Mölle, Stefan Richter, and Peter Rossmanith. "Divide-and-Color". In: *Graph-Theoretic Concepts in Computer Science, 32nd International Workshop, WG 2006, Bergen, Norway, June 22-24, 2006, Revised Papers*. Ed. by Fedor V. Fomin. Vol. 4271. Lecture Notes in Computer Science. Springer, 2006, pp. 58–67. DOI: 10.1007/11917496_6.

[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.

[KR08] Martin Kreuzer and Lorenzo Robbiano. *Computational Commutative Algebra 1*. Springer Publishing Company, Incorporated, 2008. ISBN: 354067733X, 9783540677338.

[KR13] Ashraf M. Kibriya and Jan Ramon. "Nearly exact mining of frequent trees in large networks". In: *Data Min. Knowl. Disc.* 27.3 (2013), pp. 478–504. DOI: 10.1007/s10618-013-0321-2.

[KT06] Jon M. Kleinberg and Éva Tardos. *Algorithm design*. Addison-Wesley, 2006. ISBN: 978-0-321-37291-8.

[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.

[KW16] Ioannis Koutis and Ryan Williams. "Limits and Applications of Group Algebras for Parameterized Problems". In: *ACM T. Algorithms* 12.3 (2016), 31:1–31:18. DOI: 10.1145/2885499.

[Leo+05] Paul Leopardi et al. "A generalized FFT for Clifford algebras". In: *Bulletin of the Belgian Mathematical Society-Simon Stevin* 11.5 (2005), pp. 663–688.

[Li+17] Wenjun Li, Yixin Cao, Jianer Chen, and Jianxin Wang. "Deeper local search for parameterized and approximation algorithms for maximum internal spanning tree". In: *Inf. Comput.* 252 (2017), pp. 187–200. DOI: 10.1016/j.ic.2016.11.003. URL: https://doi.org/10.1016/j.ic.2016.11.003.

[Lok+15] Daniel Lokshtanov, Pranabendu Misra, Fahad Panolan, and Saket Saurabh. "Deterministic Truncation of Linear Matroids". In: *Automata, Languages, and Programming: 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*. Ed. by Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 922–934. ISBN: 978-3-662-47672-7. DOI: 10.1007/978-3-662-47672-7_75. URL: http://dx.doi.org/10.1007/978-3-662-47672-7_75.

[Lov77] László Lovász. "Flats in matroids and geometric graphs". In: *Combinatorial Surveys (Proc. Sixth British Combinatorial Conf., Royal Holloway Coll., Egham, 1977)*. Academic Press, London, 1977, pp. 45–86.

[Mac94] Francis Sowerby Macaulay. *The algebraic theory of modular systems*. Vol. 19. Cambridge University Press, 1994.

[Mar09a] Dániel Marx. "A parameterized view on matroid optimization problems". In: *Theor. Comput. Sci.* 410.44 (2009), pp. 4471–4479. DOI: 10.1016/j.tcs.2009.07.027. URL: https://doi.org/10.1016/j.tcs.2009.07.027.

[Mar09b] Dániel Marx. "A parameterized view on matroid optimization problems". In: *Theoretical Computer Science* 410.44 (2009), pp. 4471–4479.

[Mat79] Rudolf Mathon. "A Note on the Graph Isomorphism counting Problem". In: *Inform. Process. Lett.* 8.3 (1979), pp. 131–132. DOI: 10.1016/0020-0190(79)90004-8.

[Mau76] Stephen B. Maurer. "Matrix Generalizations of Some Theorems on Trees, Cycles and Cocycles in Graphs". In: *SIAM Journal on Applied Mathematics* 30.1 (1976), pp. 143–148. DOI: 10.1137/0130017.

[Mil+02] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. "Network motifs: Simple building blocks of complex networks". In: *Science* 298.5594 (2002), pp. 824–827. ISSN: 0036-8075. DOI: 10.1126/science.298.5594.824.

[Mon85] Burkhard Monien. “How to Find Long Paths Efficiently”. In: *Analysis and Design of Algorithms for Combinatorial Problems*. Ed. by G. Ausiello and M. Lucertini. Vol. 109. North-Holland Mathematics Studies. North-Holland, 1985, pp. 239–254. DOI: 10.1016/S0304-0208(08)73110-4.

[MS08] Kurt Mehlhorn and Peter Sanders. *Algorithms and Data Structures: The Basic Toolbox*. Springer, 2008. ISBN: 978-3-540-77977-3. DOI: 10.1007/978-3-540-77978-0. URL: https://doi.org/10.1007/978-3-540-77978-0.

[NRR54] Harry Nyquist, Stephen O. Rice, and John F. Riordan. “The distribution of random determinants”. In: *Quart. Appl. Math.* 12.2 (1954), pp. 97–104. ISSN: 0033569X, 15524485. URL: http://www.jstor.org/stable/43634123.

[OSM09] Saeed Omidi, Falk Schreiber, and Ali Masoudi-Nejad. “MODA: An efficient algorithm for network motif discovery in biological networks”. In: *Genes Genet. Syst.* 84.5 (2009), pp. 385–395. DOI: 10.1266/ggs.84.385.

[Oxl92] James G. Oxley. *Matroid Theory*. Oxford University Press, 1992. ISBN: 978-0-19-853563-8.

[Por81] Ian R. Porteous. “CLIFFORD ALGEBRAS”. In: *Topological Geometry*. Cambridge University Press, 1981, pp. 240–276. DOI: 10.1017/CBO9780511623943.015.

[Pra18] Kevin Pratt. “Faster Algorithms via Waring Decompositions”. In: *CoRR* abs/1807.06194 (2018). arXiv: 1807.06194. URL: http://arxiv.org/abs/1807.06194.

[Pra19] Kevin Pratt. “Waring Rank, Parameterized and Exact Algorithms”. In: *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, MD, USA, November 9-12, 2019*. 2019. Forthcoming.

[PS05] Elena Prieto and Christian Sloper. “Reducing to independent set structure: the case of k-internal spanning tree”. In: *Nordic Journal of Computing* 12.3 (2005), pp. 308–318.

[PY93] Christos H. Papadimitriou and Mihalis Yannakakis. “On Limited Nondeterminism and the Complexity of the V.C Dimension (Extended Abstract)”. In: *Proceedings of the Eigth Annual Structure in Complexity Theory Conference, San Diego, CA, USA, May 18-21, 1993*. 1993, pp. 12–18. DOI: 10.1109/SCT.1993.336545. URL: https://doi.org/10.1109/SCT.1993.336545.

[PY96] Christos H. Papadimitriou and Mihalis Yannakakis. "On Limited Nondeterminism and the Complexity of the V-C Dimension". In: *J. Comput. Syst. Sci.* 53.2 (1996), pp. 161–170. DOI: 10.1006/jcss.1996.0058. URL: https://doi.org/10.1006/jcss.1996.0058.

[Ram+14] Jan Ramon, Constantin Comendant, Mostafa Haghir Chehreghani, and Yuyi Wang. "Graph and Network Pattern Mining". In: *Mining User Generated Content.* Ed. by Marie-Francine Moens, Juanzi Li, and Tat-Seng Chua. Chapman and Hall/CRC, 2014, pp. 97–126.

[Rob55] Herbert Robbins. "A remark on Stirling's formula". In: *Amer. Math. Monthly* 62 (1955), pp. 26–29. DOI: 10.2307/2308012. URL: https://doi.org/10.2307/2308012.

[Rot17] Marc Roth. "Counting Restricted Homomorphisms via Möbius Inversion over Matroid Lattices". In: *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria.* 2017, 63:1–63:14. DOI: 10.4230/LIPIcs.ESA.2017.63. URL: https://doi.org/10.4230/LIPIcs.ESA.2017.63.

[Rot97] Gian-Carlo Rota. *Indiscrete thoughts.* Birkhäuser, 1997. DOI: 10.1007/978-0-8176-4781-0.

[Sch+15] Benjamin Schiller, Sven Jager, Kay Hamacher, and Thorsten Strufe. "StreaM - A Stream-Based Algorithm for Counting Motifs in Dynamic Graphs". In: *Proceedings of the 2nd International Conference on Algorithms for Computational Biology (AlCoB).* 2015, pp. 53–67. DOI: 10.1007/978-3-319-21233-3_5.

[Sch80] Jacob T. Schwartz. "Fast Probabilistic Algorithms for Verification of Polynomial Identities". In: *J. ACM* 27.4 (1980), pp. 701–717. DOI: 10.1145/322217.322225.

[Sha13] Masoumeh Sepideh Shafiei. "Apolarity for determinants and permanents of generic symmetric matrices". In: *arXiv preprint arXiv:1303.1860* (2013).

[Sha15] Sepideh Masoumeh Shafiei. "Apolarity for determinants and permanents of generic matrices". In: *J. Commut. Algebra* 7.1 (Mar. 2015), pp. 89–123. DOI: 10.1216/JCA-2015-7-1-89. URL: https://doi.org/10.1216/JCA-2015-7-1-89.

[Sho88] Victor Shoup. "New Algorithms for Finding Irreducible Polynomials over Finite Fields". In: *29th Annual Symposium on Foundations of Computer Science, White Plains, New York, USA, 24-26 October 1988.* 1988, pp. 283–290. DOI: 10.1109/SFCS.1988.21944. URL: https://doi.org/10.1109/SFCS.1988.21944.

[Sho94] Victor Shoup. “Fast Construction of Irreducible Polynomials over Finite Fields”. In: *J. Symb. Comput.* 17.5 (1994), pp. 371–391. DOI: 10.1006/jsco.1994.1025. URL: https://doi.org/10.1006/jsco.1994.1025.

[Sno12] Michael Snook. “Counting Bases of Representable Matroids”. In: *Electr. J. Comb.* 19.4 (2012), P41. URL: http://www.combinatorics.org/ojs/index.php/eljc/article/view/v19i4p41.

[Sok05] Alan D. Sokal. “The multivariate Tutte polynomial (alias Potts model) for graphs and matroids”. In: *Surveys in Combinatorics*. Vol. 327. London Mathematical Society Lecture Note Series. 2005, pp. 173–226.

[SS05] Falk Schreiber and Henning Schwöbbermeyer. “Frequency concepts and pattern detection for the analysis of motifs in networks”. In: *Transactions on computational systems biology III*. Springer, 2005, pp. 89–104. DOI: 10.1007/11599128_7.

[SS11] René Schott and G. Stacey Staples. “Complexity of counting cycles using zeons”. In: *Comput. Math. Appl.* 62.4 (2011), pp. 1828–1837. DOI: 10.1016/j.camwa.2011.06.026.

[Sta99] Richard P. Stanley. *Enumerative Combinatorics: Volume 2*. Cambridge studies in advanced mathematics 62. New York, NY, USA: Cambridge University Press, 1999.

[Sto10] Andrew James Stothers. “On the complexity of matrix multiplication”. PhD thesis. The University of Edinburgh, 2010.

[Tas13] Nina Sofia Taslaman. “Exponential-Time Algorithms and Complexity of NP-Hard Graph Problems”. In: (2013).

[Tod91] Seinosuke Toda. “PP is as hard as the polynomial-time hierarchy”. In: *SIAM Journal on Computing* 20.5 (1991), pp. 865–877. DOI: 10.1137/0220053.

[Tod92] Seinosuke Toda. “Classes of arithmetic circuits capturing the complexity of computing the determinant”. In: *IEICE Transactions on Information and Systems* 75.1 (1992), pp. 116–124.

[Ull76] Julian R. Ullmann. “An algorithm for subgraph isomorphism”. In: *J. ACM* 23.1 (1976), pp. 31–42. DOI: 10.1145/321921.321925.

[Uma19] Chris Umans. “Fast generalized DFTs for all finite groups”. In: *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, MD, USA, November 9-12, 2019*. 2019. Forthcoming.

[Val79] Leslie G. Valiant. “The complexity of computing the permanent”. In: *Theoretical Computer Science* 8.2 (1979), pp. 189–201. DOI: 10.1016/0304-3975(79)90044-6.

[Ver91] Dirk Llewellyn Vertigan. "On the computational complexity of Tutte, Jones, Homfly and Kauffman invariants." PhD thesis. University of Oxford, 1991.

[Ver98] Dirk Vertigan. "Bicycle dimension and special points of the Tutte polynomial". In: *Journal of Combinatorial Theory, Series B* 74.2 (1998), pp. 378–396.

[VW92] Dirk Vertigan and Dominic J. A. Welsh. "The Compunational Complexity of the Tutte Plane: the Bipartite Case". In: *Combinatorics, Probability & Computing* 1 (1992), pp. 181–187. DOI: 10.1017/S0963548300000195. URL: http://dx.doi.org/10.1017/S0963548300000195.

[Wer06] Sebastian Wernicke. "Efficient detection of network motifs". In: *IEEE ACM T. Comput. BI* 3.4 (2006). DOI: 10.1109/TCBB.2006.51.

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

[Wil12] Virginia Vassilevska Williams. "Multiplying matrices faster than coppersmith-winograd". In: *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*. 2012, pp. 887–898. DOI: 10.1145/2213977.2214056. URL: http://doi.acm.org/10.1145/2213977.2214056.

[Wło16] Michał Włodarczyk. "Clifford Algebras Meet Tree Decompositions". In: *11th International Symposium on Parameterized and Exact Computation, IPEC 2016, August 24-26, 2016, Aarhus, Denmark*. Ed. by Jiong Guo and Danny Hermelin. Vol. 63. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016, 29:1–29:18. DOI: 10.4230/LIPIcs.IPEC.2016.29.

[WW13] Virginia Vassilevska Williams and Ryan Williams. "Finding, minimizing, and counting weighted subgraphs". In: *SIAM J. Comput.* 42.3 (2013), pp. 831–854. DOI: 10.1137/09076619X.

[Zeh15] Meirav Zehavi. "Mixing Color Coding-Related Techniques". In: *Proceedings of the 23rd Annual European Symposium on Algorithms (ESA)*. Vol. 9294. Springer, 2015, pp. 1037–1049. DOI: 10.1007/978-3-662-48350-3_86.

[Zeh17] Meirav Zehavi. "Algorithms for k-Internal Out-Branching and k-Tree in Bounded Degree Graphs". In: *Algorithmica* 78.1 (May 2017), pp. 319–341. ISSN: 1432-0541. DOI: 10.1007/s00453-016-0166-3. URL: https://doi.org/10.1007/s00453-016-0166-3.

[Zhu17] Dmitriy Zhuk. *The Proof of CSP Dichotomy Conjecture*. 2017. eprint: 1704.01914.

[Zip79] Richard Zippel. “Probabilistic algorithms for sparse polynomials”. In: *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation, Marseille, France, June 1979, Proceedings*. 1979, pp. 216–226. DOI: 10.1007/3-540-09519-5_73.