Extensor-Coding

Cornelius Brand[a], Holger Dell[a], and Thore Husfeldt[b]

[a]Saarland University and Cluster of Excellence (MMCI), Saarbrücken, Germany,
{cbrand,hdell}@mmci.uni-saarland.de
[b]Lund University and Basic Algorithms Research Copenhagen, ITU Copenhagen, thore@itu.dk

April 26, 2018

We devise an algorithm that approximately computes the number of paths of length k in a given directed graph with n vertices up to a multiplicative error of $1 \pm \varepsilon$. Our algorithm runs in time $\varepsilon^{-2}4^k(n+m)\,\mathrm{poly}(k)$. The algorithm is based on associating with each vertex an element in the exterior (or, Grassmann) algebra, called an extensor, and then performing computations in this algebra. This connection to exterior algebra generalizes a number of previous approaches for the longest path problem and is of independent conceptual interest. Using this approach, we also obtain a deterministic $2^k \cdot \mathrm{poly}(n)$ time algorithm to find a k-path in a given directed graph that is promised to have few of them. Our results and techniques generalize to the subgraph isomorphism problem when the subgraphs we are looking for have bounded pathwidth. Finally, we also obtain a randomized algorithm to detect k-multilinear terms in a multivariate polynomial given as a general algebraic circuit. To the best of our knowledge, this was previously only known for algebraic circuits not involving negative constants.

1. Introduction

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, one of the benchmarks of progress in parameterized algorithms. Our approach generalizes and unifies previous techniques in a clean fashion, including the color-coding method of Alon, Yuster, and Zwick [4] and the vector-coding idea of Koutis [41]. Color-coding yields a randomized algorithm for approximately counting k-paths [1] that runs in time $(2e)^k\,\mathrm{poly}(n)$. We improve the running time to $4^k\,\mathrm{poly}(n)$, addressing an open problem in the survey article of Koutis and Williams [42]. 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," [54] 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.

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 [30]. We consider the decision version, the k-path problem, in which we wish to find a path of length k in a given graph G. It was proved fixed-parameter tractable *avant la lettre* [50], and a sequence of both iterative improvements and conceptual breakthroughs [11, 4, 7, 40, 16, 27, 63] have lead to the current state-of-the-art for undirected graphs: a randomized algorithm by Björklund *et al.* [9] in time $1.66^k \cdot \mathrm{poly}(n)$. For directed graphs, the fastest known randomized algorithm is by Koutis and Williams [43] in time $2^k \cdot \mathrm{poly}(n)$, whereas the fastest deterministic algorithm is due to Zehavi [66] 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 [19, 60]: 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 [49]. 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 applications in the study of gene transcription networks, neural networks, and social networks [49]. Consequently, there is a large body of work dedicated to algorithmic discovery of network motifs [32, 1, 52, 37, 57, 18, 38, 62, 55]. For example, Kibriya and Ramon [39, 53] use the ideas of Koutis and Williams [43] 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 [25, 26] was resolved by Bulatov [12, 13] almost a decade before proofs were announced for the decision version by Bulatov [14] and Zhuk [67]. 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 [64] (also cf. [44, 21, 20]), and it is #W[1]-hard whenever H is from a class of graphs with unbounded vertex cover number [21, 20], 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 [20] 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.

Our results. For finite directed or undirected graphs H and G, let $\mathrm{Sub}(H,G) \in \mathbf{N}$ be the number of (not necessarily induced) subgraphs of G that are isomorphic to H. The main algorithmic result in this paper is a randomized algorithm that computes an approximation to this number.

Theorem 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%*,*

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

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 path-width $\mathrm{pw}(H)$, 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 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 [5], who use a sampling method based on Karp and Luby [36] to obtain a version of Theorem 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 [2, 3] are able to combine the color-coding technique by Alon, Yuster, and Zwick [4] 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.* [1], in turn, use the color-coding technique to obtain the first singly-exponential time version of Theorem 1, in particular with an algorithm running in time $\varepsilon^{-2} \cdot (2e)^k \cdot n^{\mathrm{tw}(H)+O(1)}$. To the best of our knowledge, Theorem 1 is now the fastest known algorithm to approximately count subgraphs of small pathwidth.

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

Theorem 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 $C \in \mathbf{N}$ 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.* [28] detect subgraphs in randomized time $\tilde{O}(2^k n^{\mathrm{tw}(H)+1})$ and Fomin *et al.* [27] 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 [66], 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 3 (Detecting multilinear terms)**.** *Given an algebraic circuit C over $\mathbf{Z}[\zeta_1, \ldots, \zeta_n]$ 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 [41, 43] 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 3. To the best of our knowledge, this is the first fixed-parameter tractable algorithm for the problem of detecting a k-multilinear term in the polynomial computed by a general algebraic circuit. Our algorithm uses color-coding and performs the computation in the exterior algebra over $\mathbf{Q}^k$. 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 [33], 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 [34], the running time of the algorithm in Theorem 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 [15], 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 [35]. Moreover, the factor ε^{-2} in the running time stems from an application of Chebyshev's inequality and is unlikely to be avoidable.

1.1. Organization

In the body text of the present manuscript, we focus entirely on paths instead of general subgraphs H. Section 2 contains an elementary development of the exterior algebra, deliberately eschewing abstract algebra. Section 3 then presents a number of different extensor-codings and establishes Theorems 1 and 2 for the case where the pattern graph H is a k-path: Theorem 1 corresponds to Algorithm C and Theorem 8 in Section 3.6; Theorem 2 corresponds to Algorithm F and Theorem 11 in Section 3.7. Section 4 is mainly expository and connects our approach to previous work. The technical details needed to establish Theorems 1–3 in full generality are moved to the appendices.

1.2. Graphs and Walks

Let G be a directed graph with n vertices and m edges. The set of vertices is $V(G)$ and enumerated as $\{v_1, \dots, v_n\}$. The set of edges is $E(G)$, the edge from u to v is denoted by uv. A sequence of vertices $w_1, \dots, w_k$ in $V(G)$ such that $w_i w_{i+1} \in E$ holds for all $i \in \{1, \dots, k-1\}$ is called a *k-walk* in G. A walk of distinct vertices is called a *path*. The set of k-walks is denoted by $\mathscr{W}$ and the set of k-paths is denoted by $\mathscr{P}$. We write $\mathrm{poly}(n)$ for the set of polynomially bounded functions in n. Throughout the document, we silently assume $k \leq n$.

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

$$f(G;\xi) = \sum_{w_1 \dots w_k \in \mathscr{W}} \xi(w_1)\xi(w_1w_2)\xi(w_2)\cdots\xi(w_{k-1})\xi(w_{k-1}w_k)\xi(w_k)\,, \tag{2}$$

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:

$$f(G;\xi) = \Big(1 \dots 1\Big) \cdot A^{k-1} \cdot \begin{pmatrix} \xi(v_1) \\ \vdots \\ \xi(v_n) \end{pmatrix}, \tag{3}$$

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

$$a_{vw} = \begin{cases} \xi(v)\xi(vw), & \text{if } vw \in E(G); \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Note that the expression for $f(G;\xi)$ in (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 (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 (equivalently, we can view this process as a distributed algorithm that computes $(A^{k-1} \cdot (\xi(v_1)\dots\xi(v_n))^T)_v$ at each vertex v in $k-1$ rounds of synchronized communication). 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 $1 \in R$.

2. The Exterior Algebra

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 Section 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 $a_1, \ldots, a_k \in F$. We sometimes write a as the column vector $(a_1, \ldots, a_k)^T$. 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

$$\mathbf{e}_I \wedge \mathbf{e}_J = (-1)^{\operatorname{sgn}(I,J)} \mathbf{e}_{I \cup J}\,,$$

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.*, $(\mathbf{e}_1 \wedge \mathbf{e}_3 \wedge \mathbf{e}_6) \wedge (\mathbf{e}_2 \wedge \mathbf{e}_4) = -\mathbf{e}_1 \wedge \mathbf{e}_3 \wedge \mathbf{e}_2 \wedge \mathbf{e}_6 \wedge \mathbf{e}_4 = \mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3 \wedge \mathbf{e}_6 \wedge \mathbf{e}_4 = -\mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3 \wedge \mathbf{e}_4 \wedge \mathbf{e}_6$.

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 $x\wedge y = (x_1y_2-y_1x_2)\cdot \mathbf{e}_1\wedge\mathbf{e}_2$, 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 $(x_1,\dots,x_k)$ 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 $(\mathbf{e}_1,\dots,\mathbf{e}_k)$ 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 $x_1,\dots,x_k\in F^k$ and write

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

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

$$x_1\wedge\cdots\wedge x_k=\det\begin{pmatrix}x_{11}&\cdots&x_{1k}\\ \vdots&\ddots&\vdots\\ x_{k1}&\cdots&x_{kk}\end{pmatrix}\cdot\mathbf{e}_{[k]}\,, \tag{5}$$

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 $x\wedge x=(\mathbf{e}_1\wedge\mathbf{e}_3+\mathbf{e}_2)\wedge(\mathbf{e}_1\wedge\mathbf{e}_3+\mathbf{e}_2)=\mathbf{e}_1\wedge\mathbf{e}_3\wedge\mathbf{e}_1\wedge\mathbf{e}_3+\mathbf{e}_1\wedge\mathbf{e}_3\wedge\mathbf{e}_2+\mathbf{e}_2\wedge\mathbf{e}_1\wedge\mathbf{e}_3+\mathbf{e}_2\wedge\mathbf{e}_2=0-\mathbf{e}_1\wedge\mathbf{e}_2\wedge\mathbf{e}_3-\mathbf{e}_1\wedge\mathbf{e}_2\wedge\mathbf{e}_3+0=-2\cdot\mathbf{e}_1\wedge\mathbf{e}_2\wedge\mathbf{e}_3\neq 0$.

2.3. Representation and Computation

We represent an extensor $x\in\Lambda(F^k)$ by its coefficients in the expansion $x=\sum_{I\subseteq\{1,\dots,k\}}x_I\mathbf{e}_I$, 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

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

When y belongs to $\Lambda^j(F^k)$, we can restrict the summation to subsets J with $|J|=j$. Thus, $x\wedge y$ for $x\in\Lambda(F^k)$ and $y\in\Lambda^j(F^k)$ 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 (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*

$$z_I=\sum_{J\subseteq I}(-1)^{\operatorname{sgn}(J,I\setminus J)}x_Jy_{I\setminus J}\,, \tag{6}$$

we see that $x\wedge y$ can be computed in 3^k field operations. By following Leopardi [45] and the subsequent analysis of Włodarczyk [65], this bound can be improved to $O^*(2^{\omega\frac{k}{2}})$, where ω is the exponent for matrix multiplication. 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. (We never need this.)

Name		$v_i \mapsto$	$e \mapsto$	Algebra	Section
ϕ	Vandermonde	$(i^0, \dots, i^{k-1})^T$	1	$\Lambda(F^k)$	3.2, 3.3
$\overline{\phi}$	Lifted Vandermonde	$\overline{\phi(v_i)}$	1	$\Lambda(F^{2k})$	3.5
$\overline{\beta}$	Lifted Bernoulli	$\overline{(\pm 1, \dots, \pm 1)^T}$	1	$\Lambda(F^{2k})$	3.6
η	Edge-variable	$\phi(v_i)$	y_e	$\Lambda(F^k)[Y]$	3.7
ρ	Random edge-weight	$\phi(v_i)$	Random $r \in \{1, \dots, 100k\}$	$\Lambda(F^k)$	4.1
λ	Labeled walks	$(x_i^{(1)}, \dots, x_i^{(k)})^T$	y_e	$\Lambda(F^k)[X, Y]$	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

Table 1: Extensor-codings of graphs used in this paper.

3. Extensor-coding

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. *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:

$$f(G; \xi) = \sum_{W \in \mathscr{W}} \xi(W) = \sum_{P \in \mathscr{P}} \xi(P)\,. \tag{7}$$

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 (5), we have

$$\xi(w_1 \dots w_k) = d \cdot \mathbf{e}_{[k]}\,, \tag{8}$$

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.

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

$$\phi(v_i) = (1, i^1, i^2, \dots, i^{k-1})^T \text{ for all } i \in \{1, \dots, n\}\,. \tag{9}$$

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

$$\Phi_P = \begin{pmatrix} 1 & 1 & \dots & 1 \\ i_1 & i_2 & \dots & i_k \\ \vdots & \vdots & \ddots & \vdots \\ i_1^{k-1} & i_2^{k-1} & \dots & i_k^{k-1} \end{pmatrix}. \tag{10}$$

In particular,

$$d = \det \Phi_P = \prod_{\substack{i_a, i_b \\ a<b}} (i_a - i_b)\,. \tag{11}$$

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 (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 (7) has at most one term and cancellations cannot occur.

This allows us to establish Thm. 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 $F = \mathbf{Q}$. Let ϕ be the Vandermonde extensor-coding as in (9).
U2 (Compute the walk-sum) Compute $f(G;\phi)$ as in (4).
U3 (Decide.) If $f(G;\phi)$ is non-zero, then return 'yes.' Otherwise, return 'no.' □

Theorem 6. *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 (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 5 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. 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. □

3.4. Blades and Lifts

The reason that cancellations can occur in (7) is that the coefficients $d \in F$ in (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

$$\overline{x} = \Big(\sum_{i\in\{1,\dots,k\}} a_i \mathbf{e}_i\Big) \wedge \Big(\sum_{j\in\{1,\dots,k\}} a_j \mathbf{e}_{j+k}\Big)\,. \tag{12}$$

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 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

$$\Xi_P = \Big(\xi(w_1) \ldots \xi(w_k)\Big) \,.$$

From Property (W3), we get

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

and

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

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

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

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:

$$f(G; \overline{\xi}) = \pm \sum_{P \in \mathscr{P}} (\det \Xi_P)^2 \cdot \mathbf{e}_{[2k]} \,. \tag{13}$$

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 5. 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.* [17, 16], but does not come close to the record bound $2.5961^k \cdot \mathrm{poly}(n)$ of Zehavi [66].

Theorem 7 (Superseded by [66])**.** *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 (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. 2.3. Overall, this leads to the claimed running time. □

3.6. Bernoulli Vectors

We present our algorithm for approximate counting. Now instead of the Vandermonde extensor-coding as in Lemma 5, 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 $\beta(w_1)$, …, $\beta(w_k)$, 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 $i \in \{1, \dots, n\}$, 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 1, approximating $\mathrm{Sub}(H, G)$ when H is the k-path. In this case, $\mathrm{Sub}(H, G) = |\mathscr{P}|$.

Theorem 8. *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*

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

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 9 ([51])**.** *Let B be a $k \times k$ Bernoulli matrix. Then,*

$$\mathbf{E} \det B^2 = k! \tag{14}$$
$$\mathbf{E} \det B^4 \le (k!)^2 \cdot k^3 \,. \tag{15}$$

For completeness, we include a careful proof for a slightly different distribution in Appendix A.

Proof of Theorem 8. Run algorithm C with $t = 100k^3/\varepsilon^2$. Set $\mu = |\mathscr{P}|$. Recall from (13) that X_j can be written as

$$X_j = \pm(\det B_1^2 + \det B_2^2 + \cdots + \det B_\mu^2) \,, \tag{16}$$

where for $i \in \{1, \dots, \mu\}$, 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 9, we have $\mathbf{E} \det B_i^2 = k!$ for each $i \in \{1, \dots, \mu\}$, so by linearity of expectation,

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

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

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

where the last two inequalities uses $\mathrm{Var}\, Y \le \mathbf{E}\, Y^2$ with $Y = \det B_i^2$ and (15) in Theorem 9 with $B = B_i$. We obtain

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

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 $\mathbf{E}\,X = t\mu k!/(k!t) = \mu$. Recalling that $\mathrm{Var}\,(a \cdot X) = a^2 \cdot \mathrm{Var}\,(X)$ for a random variable X and a scalar a, by independence of the X_j, we have

$$\mathrm{Var}\,X = \mathrm{Var}\Big(\frac{1}{k!t}\sum_{j=1}^{t} X_j\Big) = \frac{1}{(k!t)^2}\sum_{j=1}^{t}\mathrm{Var}\,X_j \le \frac{1}{(k!t)^2}t\mu^2(k!)^2k^3 = \frac{\mu^2k^3}{t}\,.$$

Now Chebyshev's inequality gives

$$\Pr(|X-\mu| \ge \varepsilon\mu) \le \frac{\mathrm{Var}\,X}{\varepsilon^2\mu^2} \le \frac{\mu^2k^3}{\varepsilon^2\mu^2t} = \frac{1}{100}\,,$$

which implies the stated bound.

The claim on the running time follows from the discussion in Sec. 2.3 and the representation of the input as adjacency lists. □

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 10 (Theorem 2 in [10])**.** *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 $F = \mathbf{Z}$ 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 (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 $\mathbf{Z}[Y]$.] Create a circuit L with inputs from $\mathbf{Z}$ and Y as follows. Every gate g in K corresponds to 2^k gates g_I for $I \subseteq \{1,\ldots,k\}$ 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

$$\sum_{\substack{J\subseteq I\\ |J|\le 1}} (-1)^{\mathrm{sgn}\,(I\setminus J, J)} g'_{I\setminus J} g''_J\,.$$

(This is (6), noting $g''_J = 0$ for $|J| > 1$.) If g is the output gate of K then $g_{\{1,\ldots,k\}}$ 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 2 for the case where the pattern graph H is a path.

Theorem 11. *Algorithm F is a deterministic algorithm for the k-path problem when there are at most $C \in \mathbf{N}$ 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 (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 (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 10 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 $O(2^k(n+m)\operatorname{poly}(k))$ 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 10 is thus within the claimed running time bound. □

4. Connection to Previous Work

In this section, we show how our approach using exterior algebras specializes to the group algebra approach of Koutis [41] when the ground field has characteristic two. We also argue that the combinatorial approach of Björklund *et al.* [9] using *labeled walks* can be seen as an evaluation over an exterior algebra. Moreover, we show how color-coding [4] arises as a special case, and present the recent approach of representative paths due to Fomin et al. [27] in the language of exterior algebra.

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.

Theorem 12 ([41, 63])**.** *There is a randomized algorithm for the k-path problem with running time $2^k(n+m)\operatorname{poly}(k)$.*

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

$$\sum_{w_1 \cdots w_k \in \mathscr{P}} y_{w_1 w_2} \cdots y_{w_{k-1} w_k} \cdot \phi(w_1 \ldots w_k) \tag{17}$$

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 (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, $f(G;\rho)$ is nonzero with probability $\frac{1}{100}$. □

4.2. Group Algebras

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

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

Proposition 13. *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 $F[\mathbf{Z}_2^k]$ is isomorphic to $\Lambda(F^k)$.*

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

Remark. The previous proposition shows that over fields of characteristic two, our exterior algebras specialize exactly to the group algebras used by Koutis and Williams [41, 63], and therefore, the approach of using random edge-weights in the coding ρ from Section 4.1 specializes to Williams' algorithm [63] 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 $k \in \mathbf{N}$, 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.

4.3. Labeled Walks

The main goal of the labeled walk approach of Björklund *et al.* [9] was to give an algorithm for the *undirected* case running in time $1.66^k \cdot \mathrm{poly}(n)$. 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 $2^k \cdot \mathrm{poly}(n)$ runtime bound of Williams [63]. This is nicely laid out in the textbook by Cygan *et al.* [22, Section 10.4], and the following presentation is guided by theirs.

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. Consider the following polynomial in the y_e and $x_i^{(j)}$:

$$P(x,y) = \sum_{w_1 \cdots w_k \in \mathscr{W}} \sum_{\ell \in S_k} \prod_{i=1}^{k} y_{w_i w_{i+1}} \prod_{i=1}^{k} x_i^{\ell(i)}. \tag{18}$$

The crucial insight is that over characteristic 2, the sum can be restricted to paths instead of walks:

$$P(x,y) = \sum_{w_1 \cdots w_k \in \mathscr{P}} \sum_{\ell \in S_k} \prod_{i=1}^{k} y_{w_i w_{i+1}} \prod_{i=1}^{k} x_i^{\ell(i)}. \tag{19}$$

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. 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 11, 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 (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 Section 4.1 over characteristic two, and by the remark in Section 4.2, also the connection between the group-algebra approach, identifying the three techniques as one.

4.4. Color-coding

Let us see how the color-coding-technique by Alon, Yuster, and Zwick [4] arises as a special case of extensor-coding. Consider a coding with basis vectors of F^k taken uniformly and independently,

$$\chi(v) \in \{e_1, \ldots, e_k\}\,.$$

Readers familiar with [4] are encouraged to 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 $\overline{\chi}(P) = \{0 \cdot e_{[2k]}, 1 \cdot e_{[2k]}\}$, 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 [56].

Lemma 14. *$Z(F^k)$ 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.*, [61]), or more directly, fast subset convolution [8], 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 [4].

We apply these constructions in a similar setting in Appendix C.

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 [50]. Recent representative-sets algorithms [27, 66], 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 [46]. In fact, as pointed out by Marx [47, Proof of Lemma 4.2], the column vectors in the matroid representation are exactly Vandermonde extensors. Fomin *et al.* [27, 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.

We can give a relatively short and complete presentation. 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

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

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*

$$\phi(P) \in \langle \mathscr{R}_v^p \rangle \qquad \text{for each } P \in \mathscr{P}_v^p \tag{20}$$

and

$$|\mathscr{R}_v^p| \leq 2^k \,. \tag{21}$$

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

$$\mathscr{Q}_v^{p+1} = \{\, Rv : \; R \in \mathscr{R}_u^p \text{ and } uv \in E(G) \,\} \,. \tag{22}$$

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 15 (Theorem 1 in [27])**.** *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 (20) and (21) hold, and that the constructed sets $\mathscr{R}_v^p$ only contain paths from $\mathscr{P}_v^p$. For the size invariant (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 (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

$$\phi(Puv) = \phi(Pu) \wedge \phi(v) = \left(\sum_{j=1}^{d} a_i \phi(R_i) \right) \wedge \phi(v) = \sum_{j=1}^{d} a_i \phi(R_i) \wedge \phi(v) = \sum_{j=1}^{d} a_i \phi(R_i v) \,. \tag{23}$$

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 (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 (23), this shows that $\phi(Puv)$ belongs to $\langle \mathscr{R}_v^{p+1} \rangle$, so that the representation invariant (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 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 (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 $\exp(O(k)) \operatorname{poly}(n)$. □

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

Acknowledgments

We thank Markus Bläser, Radu Curticapean, Balagopal Komarath, Ioannis Koutis, Pascal Schweitzer, Karteek Sreenivasaiah and Meirav Zehavi for some valuable discussions and insights. Part of this work was done while the authors attended Dagstuhl seminar 17341, *Computational Counting*. Thore Husfeldt is supported by the Swedish Research Council grant VR-2016-03855 and the Villum Foundation grant 16582.

References

[1] Noga Alon, Phuong Dao, Iman Hajirasouliha, Fereydoun Hormozdiari, and S Cenk Sahinalp. Biomolecular network motif counting and discovery by color coding. *Bioinformatics*, 24(13):i241–i249, 2008. doi:10.1093/bioinformatics/btn163.

[2] Noga Alon and Shai Gutner. Balanced hashing, color coding and approximate counting. In Jianer Chen and Fedor V. Fomin, editors, *Parameterized and Exact Computation, 4th International Workshop, IWPEC 2009, Copenhagen, Denmark, September 10-11, 2009, Revised Selected Papers*, volume 5917 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2009. doi:10.1007/978-3-642-11269-0_1.

[3] Noga Alon and Shai Gutner. Balanced families of perfect hash functions and their applications. *ACM T. Algorithms*, 6(3):54:1–54:12, 2010. doi:10.1145/1798596.1798607.

[4] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995. doi:10.1145/210332.210337.

[5] Vikraman Arvind and Venkatesh Raman. Approximation algorithms for some parameterized counting problems. In Prosenjit Bose and Pat Morin, editors, *Algorithms and Computation, 13th International Symposium, ISAAC 2002 Vancouver, BC, Canada, November 21-23, 2002, Proceedings*, volume 2518 of *Lecture Notes in Computer Science*, pages 453–464. Springer, 2002. doi:10.1007/3-540-36136-7_40.

[6] László Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 684–697. ACM, 2016. doi:10.1145/2897518.2897542.

[7] Andreas Björklund. Determinant sums for undirected Hamiltonicity. *SIAM J. Comput.*, 43(1):280–299, 2014. doi:10.1137/110839229.

[8] 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*, pages 67–74, 2007. doi:10.1145/1250790.1250801.

[9] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Narrow sieves for parameterized paths and packings. *J. Comput. Syst. Sci.*, 87:119–139, 2017. doi:10.1016/j.jcss.2017.03.003.

[10] Markus Bläser, Moritz Hardt, Richard J. Lipton, and Nisheeth K. Vishnoi. Deterministically testing sparse polynomial identities of unbounded degree. *Inf. Process. Lett.*, 109(3):187–192, 2009. URL: https://doi.org/10.1016/j.ipl.2008.09.029, doi:10.1016/j.ipl.2008.09.029.

[11] Hans L. Bodlaender. On linear time minor tests with depth-first search. *J. Algorithms*, 14(1):1–23, 1993. doi:10.1006/jagm.1993.1001.

[12] Andrei A. Bulatov. The complexity of the counting constraint satisfaction problem. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *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*, volume 5125 of *Lecture Notes in Computer Science*, pages 646–661. Springer, 2008. doi:10.1007/978-3-540-70575-8_53.

[13] Andrei A. Bulatov. The complexity of the counting constraint satisfaction problem. *J. ACM*, 60(5):34:1–34:41, 2013. doi:10.1145/2528400.

[14] Andrei A. Bulatov. A dichotomy theorem for nonuniform CSPs, 2017. arXiv:1703.03021.

[15] 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. *Inform. Comput.*, 201(2):216–231, 2005. doi:10.1016/j.ic.2005.05.001.

[16] 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. *SIAM J. Comput.*, 38(6):2526–2547, 2009. doi:10.1137/080716475.

[17] 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*, pages 298–307, 2007. URL: http://dl.acm.org/citation.cfm?id=1283383.1283415.

[18] 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*, pages 106–115. ACM, 2006. doi:10.1145/1150402.1150418.

[19] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual Symposium on Theory of Computing (STOC)*, pages 151–158, 1971. doi:10.1145/800157.805047.

[20] Radu Curticapean, Holger Dell, and Dániel Marx. Homomorphisms are a good basis for counting small subgraphs. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 210–223. ACM, 2017. doi:10.1145/3055399.3055502.

[21] 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)*, pages 130–139, 2014. doi:10.1109/FOCS.2014.22.

[22] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

[23] Richard A. DeMillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Inform. Process. Lett.*, 7(4):193–195, 1978. doi:10.1016/0020-0190(78)90067-4.

[24] Josep Díaz, Maria J. Serna, and Dimitrios M. Thilikos. Counting h-colorings of partial k-trees. *Theor. Comput. Sci*, 281(1-2):291–309, 2002. doi:10.1016/S0304-3975(02)00017-8.

[25] Tomás Feder and Moshe Y. Vardi. Monotone monadic SNP and constraint satisfaction. In S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal, editors, *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 612–622. ACM, 1993. doi:10.1145/167088.167245.

[26] Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, 1998. doi:10.1137/S0097539794266766.

[27] Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient computation of representative families with applications in parameterized and exact algorithms. *J. ACM*, 63(4):29:1–29:60, 2016. doi:10.1145/2886094.

[28] Fedor V. Fomin, Daniel Lokshtanov, Venkatesh Raman, Saket Saurabh, and B. V. Raghavendra Rao. Faster algorithms for finding and counting subgraphs. *J. Comput. Syst. Sci.*, 78(3):698–706, 2012. doi:10.1016/j.jcss.2011.10.001.

[29] Fedor V. Fomin and Yngve Villanger. Treewidth computation and extremal combinatorics. *Combinatorica*, 32(3):289–308, 2012. doi:10.1007/s00493-012-2536-z.

[30] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[31] Vyacheslav L. Girko. *Theory of random determinants*, volume 45 of *Mathematics and its applications*. Springer, 1990. doi:10.1007/978-94-009-1858-0.

[32] 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*, pages 92–106. Springer, 2007. doi:10.1007/978-3-540-71681-5_7.

[33] Falk Hüffner, Sebastian Wernicke, and Thomas Zichner. Algorithm engineering for color-coding with applications to signaling pathway detection. *Algorithmica*, 52(2):114–132, 2008. doi:10.1007/s00453-007-9008-7.

[34] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

[35] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

[36] 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*, pages 56–64. IEEE Computer Society, 1983. doi:10.1109/SFCS.1983.35.

[37] 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. *BMC Bioinformatics*, 10(1):318, 2009. doi:10.1186/1471-2105-10-318.

[38] Nadav Kashtan, Shalev Itzkovitz, Ron Milo, and Uri Alon. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, 20(11):1746–1758, 2004. doi:10.1093/bioinformatics/bth163.

[39] Ashraf M. Kibriya and Jan Ramon. Nearly exact mining of frequent trees in large networks. *Data Min. Knowl. Disc.*, 27(3):478–504, 2013. doi:10.1007/s10618-013-0321-2.

[40] Joachim Kneis, Daniel Mölle, Stefan Richter, and Peter Rossmanith. Divide-and-color. In Fedor V. Fomin, editor, *Graph-Theoretic Concepts in Computer Science, 32nd International Workshop, WG 2006, Bergen, Norway, June 22-24, 2006, Revised Papers*, volume 4271 of *Lecture Notes in Computer Science*, pages 58–67. Springer, 2006. doi:10.1007/11917496_6.

[41] Ioannis Koutis. Faster algebraic algorithms for path and packing problems. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *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*, volume 5125 of *Lecture Notes in Computer Science*, pages 575–586. Springer, 2008. doi:10.1007/978-3-540-70575-8_47.

[42] Ioannis Koutis and Ryan Williams. Algebraic fingerprints for faster algorithms. *Commun. ACM*, 59(1):98–105, December 2015. doi:10.1145/2742544.

[43] Ioannis Koutis and Ryan Williams. Limits and applications of group algebras for parameterized problems. *ACM T. Algorithms*, 12(3):31:1–31:18, 2016. doi:10.1145/2885499.

[44] Miroslaw Kowaluk, Andrzej Lingas, and Eva-Marta Lundell. Counting and detecting small subgraphs via equations. *SIAM J. Discrete Math.*, 27(2):892–909, 2013. doi:10.1137/110859798.

[45] Paul Leopardi et al. A generalized fft for clifford algebras. *Bulletin of the Belgian Mathematical Society-Simon Stevin*, 11(5):663–688, 2005.

[46] László Lovász. Flats in matroids and geometric graphs. In *Combinatorial Surveys (Proc. Sixth British Combinatorial Conf., Royal Holloway Coll., Egham, 1977)*, pages 45–86. Academic Press, London, 1977.

[47] Dániel Marx. A parameterized view on matroid optimization problems. *Theor. Comput. Sci.*, 410(44):4471–4479, 2009. URL: https://doi.org/10.1016/j.tcs.2009.07.027, doi:10.1016/j.tcs.2009.07.027.

[48] Rudolf Mathon. A note on the graph isomorphism counting problem. *Inform. Process. Lett.*, 8(3):131–132, 1979. doi:10.1016/0020-0190(79)90004-8.

[49] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002. doi:10.1126/science.298.5594.824.

[50] Burkhard Monien. How to find long paths efficiently. In G. Ausiello and M. Lucertini, editors, *Analysis and Design of Algorithms for Combinatorial Problems*, volume 109 of *North-Holland Mathematics Studies*, pages 239 – 254. North-Holland, 1985. doi:10.1016/S0304-0208(08)73110-4.

[51] Harry Nyquist, Stephen O. Rice, and John F. Riordan. The distribution of random determinants. *Quart. Appl. Math.*, 12(2):97–104, 1954. URL: http://www.jstor.org/stable/43634123.

[52] Saeed Omidi, Falk Schreiber, and Ali Masoudi-Nejad. MODA: An efficient algorithm for network motif discovery in biological networks. *Genes Genet. Syst.*, 84(5):385–395, 2009. doi:10.1266/ggs.84.385.

[53] Jan Ramon, Constantin Comendant, Mostafa Haghir Chehreghani, and Yuyi Wang. Graph and network pattern mining. In Marie-Francine Moens, Juanzi Li, and Tat-Seng Chua, editors, *Mining User Generated Content.*, pages 97–126. Chapman and Hall/CRC, 2014.

[54] Gian-Carlo Rota. *Indiscrete thoughts*. Birkhäuser, 1997. doi:10.1007/978-0-8176-4781-0.

[55] 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)*, pages 53–67, 2015. doi:10.1007/978-3-319-21233-3_5.

[56] René Schott and G. Stacey Staples. Complexity of counting cycles using zeons. *Comput. Math. Appl.*, 62(4):1828–1837, 2011. doi:10.1016/j.camwa.2011.06.026.

[57] Falk Schreiber and Henning Schwöbbermeyer. Frequency concepts and pattern detection for the analysis of motifs in networks. In *Transactions on computational systems biology III*, pages 89–104. Springer, 2005. doi:10.1007/11599128_7.

[58] Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980. doi:10.1145/322217.322225.

[59] Richard P. Stanley. *Enumerative Combinatorics: Volume 2.* Number 62 in Cambridge studies in advanced mathematics. Cambridge University Press, New York, NY, USA, 1999.

[60] Julian R. Ullmann. An algorithm for subgraph isomorphism. *J. ACM*, 23(1):31–42, 1976. doi:10.1145/321921.321925.

[61] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra.* Cambridge University Press, 3rd edition, 2013. doi:10.1017/CBO9781139856065.

[62] Sebastian Wernicke. Efficient detection of network motifs. *IEEE ACM T. Comput. BI*, 3(4), 2006. doi:10.1109/TCBB.2006.51.

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

[64] Virginia Vassilevska Williams and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. *SIAM J. Comput.*, 42(3):831–854, 2013. doi:10.1137/09076619X.

[65] Michał Włodarczyk. Clifford algebras meet tree decompositions. In Jiong Guo and Danny Hermelin, editors, *11th International Symposium on Parameterized and Exact Computation, IPEC 2016, August 24-26, 2016, Aarhus, Denmark*, volume 63 of *LIPIcs*, pages 29:1–29:18. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.IPEC.2016.29.

[66] Meirav Zehavi. Mixing color coding-related techniques. In *Proceedings of the 23rd Annual European Symposium on Algorithms (ESA)*, volume 9294, pages 1037–1049. Springer, 2015. doi:10.1007/978-3-662-48350-3_86.

[67] Dmitriy Zhuk. The proof of CSP dichotomy conjecture, 2017. arXiv:1704.01914.

[68] 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*, pages 216–226, 1979. doi:10.1007/3-540-09519-5_73.

A. Random Determinants

Expressions for the higher moments of determinants of random matrices are available in the literature since the 1950s, see [51]. Such results are considered routine, and follow from exercise 5.64 in Stanley [59] or the general method laid out on page 45–46 in Girko's book [31], 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 $\mathbf{E}\, b_{ij} = 0$ and $\mathbf{E}\, b_{ij}^2 = \frac{1}{3}3 = 1$ and $\mathbf{E}\, b_{ij}^4 = \frac{1}{3}9 = 3$.

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

$$\mathbf{E} \det B^2 = k! \tag{24}$$

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

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

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,

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

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 $\mathbf{E} \det B^2$ as a function of the dimension k, which solves to (24).

To show

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

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

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

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

$$\sum_i (-1)^{4i}\, \mathbf{E}\, b_{1i}^4\, \mathbf{E} \det B_{1i}^4 = 3k\, \mathbf{E} \det B_{11}^4 = 3k f_{k-1}\,. \tag{25}$$

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

$$3 \sum_{i \neq j} (-1)^{2i+2j}\, \mathbf{E}\, b_{1i}^2\, \mathbf{E}\, b_{1j}^2\, \mathbf{E} \det(B_{1i}^2 B_{1j}^2) = 3k(k-1)\, \mathbf{E} \det(B_{11}^2 B_{12}^2)\,. \tag{26}$$

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

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

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

$$\Phi = \mathbf{E} \det(B^2_{\{1,i\},\{1,2\}} B^2_{\{1,j\},\{2,1\}}) \qquad (i \neq j)\,,$$

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

$$\mathbf{E} \det(B^2_{11} B^2_{12}) = (k-1)f_{k-2} + (k-1)(k-2)\Phi\,. \tag{27}$$

Combining (25), (26), and (27), we obtain

$$f_k = 3k f_{k-1} + 3k(k-1)^2\big(f_{k-2} + (k-2)\Phi\big)\,. \tag{28}$$

Using similar arguments from a different starting point, we obtain

$$f_{k-1} = \mathbf{E} \det B^4_{11} = 3(k-1)f_{k-2} + 3(k-1)(k-2)\Phi\,, \tag{29}$$

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 (28) and (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. We can use this distribution in Section 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 $\mathbf{Q}[\sqrt{3}]$, which can be handled just like complex numbers in spirit (after all, $\mathbf{C} = \mathbf{R}[\sqrt{-1}]$) by representing a number $a + b\sqrt{3}$ by the two rational coordinates a, b, and performing multiplication according to $(a + b\sqrt{3})(c + d\sqrt{3}) = ac + 3bd + (ad + bc)\sqrt{3}$.

B. Generalization to Subgraphs

In this section, we formally prove Theorem 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 Section 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:

$$\sum_{h \in \mathrm{Hom}(H \to G)} \prod_{v \in V(H)} \zeta_{h(v)}\,. \tag{30}$$

The variables are ζ_v for all $v \in V(G)$. We first show in §B.1 that this polynomial has small algebraic circuits when the pathwidth or the treewidth is bounded, and in §B.2 we prove Theorem 1.

B.1. Tree Decompositions

Fomin *et al.* [28, Lemma 1] construct an algebraic circuit that computes the homomorphism polynomial, based on a dynamic programming algorithm (*e.g.*, [24]). We reproduce a proof for completeness.

Lemma 16. *Let H and G be graphs with $V(H) = \{1, \ldots, k\}$ and $V(G) = \{1, \ldots, n\}$. 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 $\zeta_1, \ldots, \zeta_n$ such that C computes the homomorphism polynomial of H in G in the variables $\zeta_1, \ldots, \zeta_n$, that is, we have*

$$C(\zeta_1, \ldots, \zeta_n) = \sum_{h \in \mathrm{Hom}(H \to G)} \prod_{v \in V(H)} \zeta_{h(v)} \,. \tag{31}$$

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 $t \in V(T)$:

$$(\textit{separator at } t) \quad \mathfrak{s}(t) = \begin{cases} \emptyset \,, & t \text{ is the root of } T, \\ \mathfrak{b}(t) \cap \mathfrak{b}(s) \,, & s \text{ is the parent of } t \text{ in } T, \end{cases} \tag{32}$$

$$(\textit{cone at } t) \quad \mathfrak{c}(t) = \bigcup_{u \text{ is a descendant of } t} \mathfrak{b}(u), \tag{33}$$

$$(\textit{component at } t) \quad \mathfrak{a}(t) = \mathfrak{c}(t) \setminus \mathfrak{s}(t). \tag{34}$$

Proof of Lemma 16. 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 [29]. 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.

$$\hom(\, H \to G \mid \pi \,) = \sum_{\substack{h \in \mathrm{Hom}(H \to G) \\ h \supseteq \pi}} \prod_{v \in V(H)} \zeta_{h(v)} \,. \tag{35}$$

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

$$\hom(H \to G) = \sum_{\pi \in \mathrm{Hom}(H[S] \to G)} \hom(\, H \to G \mid \pi \,) \tag{36}$$

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

$$\hom(\, H \to G \mid \pi \,) = \prod_i \hom(\, H_i \to G \mid \pi \,) \,, \tag{37}$$

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:

$$C_v^\pi = \hom(\, H[\mathfrak{c}(v)] \to G \mid \pi \,) \,. \tag{38}$$

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 $\pi' = \pi \restriction_{\mathfrak{b}(w)}$:

$$C_v^\pi = C_w^{\pi'} \cdot \zeta_{\pi(x)} \,. \tag{39}$$

For the correctness, note that the right side of (41) is equal to

$$\hom(\, H[\mathfrak{c}(w)] \to G \mid \pi' \,) \cdot \zeta_{\pi(x)} \tag{40}$$

by the induction hypothesis (38). Since x is the unique vertex in $\mathfrak{c}(v) \setminus \mathfrak{c}(w)$ and π extends π' on x, the polynomial in (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:

$$C_v^\pi = \sum_{\pi'} C_w^{\pi'} \,. \tag{41}$$

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 (39) is equal to

$$\sum_{\pi'} \hom(\, H[\mathfrak{c}(w)] \to G \mid \pi' \,) \tag{42}$$

by the induction hypothesis (38). This is equal to $\hom(\, H[\mathfrak{c}(v)] \to G \mid \pi \,)$ due to the conditioning formula (36). For the size bound, note that x is the only vertex in $\mathfrak{b}(w) \setminus \mathfrak{b}(v)$. Thus, the sum in (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

$$C_v^\pi = C_w^\pi \cdot C_{w'}^\pi \,. \tag{43}$$

For the correctness, note that $\mathfrak{b}(v)$ is a separator for $H[\mathfrak{c}(v)]$, and so the induction hypothesis (38) together with the conditional independence (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 (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 $O(kn^{\mathrm{pw}(H)+1})$. □

B.2. Proof of Theorem 1 and 2

Theorem 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%,

$$(1-\varepsilon)\cdot \mathrm{Sub}(H,G) \leq \tilde{N} \leq (1+\varepsilon)\cdot \mathrm{Sub}(H,G)\,. \tag{44}$$

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 path-width $\mathrm{pw}(H)$, 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 16, 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:

$$C(\zeta_1,\ldots,\zeta_n) = \sum_{h\in \mathrm{Hom}(H\to G)} \prod_{v\in V(H)} \zeta_{h(v)}\,. \tag{45}$$

Following the setup of §3.6, we use the lifted Bernoulli extensor-coding $\overline{\beta}\colon V(G)\to F^{2k}$. We have

$$C(\overline{\beta}(v_1),\ldots,\overline{\beta}(v_n)) = \pm \sum_{h\in \mathrm{InjHom}(H\to G)} \prod_{v\in V(H)} \overline{\beta}(h(v))\,, \tag{46}$$

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 8. Note that this approach using (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 [48], which in turn can be computed in time $\exp(\mathrm{poly}\log k)\leq O(1.001^k)$ [6]. 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 2 in the case of paths is established through Theorem 11. For the more complicated case of general subgraphs, we can modify the polynomial (30) analogously so that it involves also the edge variables. By suitable modifications of the dynamic program that computes the corresponding circuit, Theorem 2 is established.

C. Proof of Theorem 3

In this section, we prove Theorem 3. This will follow by an application of our algebraic interpretation of color-coding from Section 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 14.

Proposition 17. *For any integer $t > 0$, an arithmetic circuit C over $\mathbf{Z}[\zeta_1,\ldots,\zeta_n]$ can be evaluated over $Z(\mathbf{Q}^t)$ in $2^t\cdot|C|\cdot\mathrm{poly}(n)$ operations over $\mathbf{Q}$.*

We are ready for the proof:

Proof of Theorem 3. We invoke Proposition 17 with Hüffner *et al.*'s [33] choice of $t = 1.3k$. One evaluation costs $2.4623^k\cdot|C|\cdot\mathrm{poly}(n)$ operations over $\mathbf{Q}$. 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 [23, 58, 68] the polynomial evaluates non-zero with constant probability of 99%. Following Hüffner *et al.* [33, Theorem 1], 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 $\mathbf{Q}$. Now, if the circuit can be evaluated over $\mathbf{Z}$ 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 $2^n \,\mathrm{poly}(n)$, and we can thus randomly pick a number from $\{1, \ldots, P\}$, where $P = 2^n \,\mathrm{poly}(n)$, until we find a prime (which can be tested in randomized polynomial time). Then we perform all the above calculations modulo p, and if the result doesn't vanish, the polynomial doesn't vanish over $\mathbf{Z}$. 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. □