Extensor-coding

Cornelius Brand[1], Holger Dell[1], and Thore Husfeldt[2]

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

February 17, 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 $\varepsilon^{-2}4^k \cdot O(n+m) \cdot \mathrm{poly}(k)$. The algorithm is based on associating with each vertex an element in the exterior (or, Grassmann) algebra, called an extensor. Computations are then performed in the exterior algebra and its tensor square. This connection to exterior algebra generalises 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, assuming that there is at most one k-path. 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 general, given algebraic circuit. To the best of our knowledge, this was previously only known for algebraic circuits not involving negative constants.

*Part of this work was done at the Dagstuhl Seminar 17341 on "Computational Counting" in August 2017. The third author is partially supported by the project VR 2016-03855 "Algebraic Graph Algorithms" of the Swedish Research Council.

1. Introduction

A path is just a walk that does not vanish in the exterior algebra. This observation leads us to a framework for algebraic graph algorithms for the k-path problem, one of the benchmarks of progress in parameterized algorithms. Our framework generalises and unifies previous approaches in a clean fashion, including the color-coding method of Alon, Yuster, and Zwick [4] and the vector-coding idea of Koutis and Williams [41, 61, 43]. We demonstrably *extend* these approaches developing a simple algorithm for approximately *counting* k-paths that improves the best known running time [1] from $(2e)^k \operatorname{poly}(n)$ to $4^k \operatorname{poly}(n)$, an open problem in the survey article of Koutis and Williams [42]. Our approach generalizes from paths to other subgraphs of bounded pathwidth.

H: Wort "extend" vermeiden. Warum emph?

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," [52] 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 path in a given graph. Clearly, this problem generalizes the NP-hard Hamiltonian path problem [29]. 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* [48], and a sequence of both iterative improvements and conceptual breakthroughs [10, 4, 7, 40, 15, 26, 61] have lead to the current state-of-the-art for undirected graphs: a randomized algorithm by Björklund *et al.* [9] in time $1.66^k \cdot \operatorname{poly}(n)$. For directed graphs, the fastest known randomized algorithm is by Koutis and Williams [43] in time $2^k \cdot \operatorname{poly}(n)$, whereas the fastest deterministic algorithm is due to Zehavi [64] in time $2.5961^k \cdot \operatorname{poly}(n)$.

Subgraph isomorphism. The subgraph isomorphism problem generalizes the k-path problem and is one of the most fundamental graph problems [18, 58]: 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 [47]. 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 [47]. Consequently, there is a large body of work dedicated to algorithmic discovery of network motifs [31, 1, 50, 37, 55, 17, 38, 60, 53]. For example, Kibriya and Ramon [39, 51] 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 [24, 25] was resolved by Bulatov [11, 12] almost a decade before proofs were announced for the decision version by Bulatov [13] and Zhuk [65]. 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 [62] (also cf. [44, 20, 19]), and it is #W[1]-hard whenever H is from a class of graphs with unbounded vertex cover number [20, 19], 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 [19] 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 $\mathrm{Sub}(H,G) \in \mathbb{N}$ 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 the number of vertices of H is k and the number of vertices of G is n.

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 $\exp(O(k\log k))\cdot n^{t+O(1)}$. 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, namely with an algorithm running in time $\varepsilon^{-2}\cdot(2e)^k\cdot n^{t+O(1)}$. They also implemented this algorithm, which outperformed some previously existing motif discovery methods. To the best of our knowledge, Theorem 1 is now the fastest known general-purpose algorithm to approximately count subgraphs.

Jerrum and Meeks [35, 46] consider the problem of approximately counting induced subgraphs that have some graph property Φ. They prove that, if all subgraph-minimal elements of Φ have bounded treewidth, then the problem has an FPT-RAS. By applying our algorithm in Theorem 1 to these subgraph-minimal elements and summing the results, our work implies an improved algorithm for this problem as well. We also obtain deterministic algorithms for the case where G contains at most one given subgraph.

Theorem 2 (Detecting subgraphs unambiguously)**.** *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 one such subgraph. This algorithm runs in time $\tilde{O}(2^k n^{\mathrm{pw}(H)+1})$, where the number of vertices of H is k and the number of vertices of G is n.*

Fomin *et al.* [27] can detect subgraphs without the unambiguity promise in randomized time $\tilde{O}(2^k n^{\mathrm{tw}(H)+1})$, and Fomin *et al.* [26] can detect them in deterministic time $\tilde{O}(2.851^k n^{O(\mathrm{tw}(H))})$. For the interesting special case of paths, which have pathwidth 1, the running time of the fastest deterministic algorithm for k-paths (without unambiguity promise) is $2.5961^k \cdot \mathrm{poly}(n)$ by Zehavi [64]. We conjecture that using our method it is possible to match the running time of the latter algorithm also without the unambiguity promise.

Theorem 3 (Detecting multilinear terms)**.** *Given an algebraic circuit C over $\mathbb{Z}[\zeta_1, \ldots, \zeta_n]$ and a number k, we can detect whether the polynomial $C(\zeta_1, \ldots, \zeta_n)$ has a degree-k multilinear term in randomized time $4.32^k \cdot |C| \cdot \mathrm{poly}(n)$.*

Using algebraic fingerprinting with elements from a group algebra, Koutis and Williams [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. Since we are able to work over an algebra whose ground field has 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 $\mathbb{Q}^k$. To reduce the running time from $2^k e^k \cdot \mathrm{poly}(n)$ to $4.32^k \cdot \mathrm{poly}(n)$, we use an idea of Hüffner, Wernicke, and Zichner [32], who improved color-coding by using $1.3 \cdot k$ instead of only k different colors.

Hardness results. Under the exponential-time hypothesis (ETH) by Impagliazzo and Paturi [33], 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 [14], 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 [34]. Moreover, the factor ε^{-2} in the running time stems from an application of Chebychev'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. These concepts suffice to demonstrate our main algebraic idea and its algorithmic applications. Section 3 then moves to the tensor square of the exterior algebra and establishes Theorem 8, which is the special case of Theorem 1: an FPT-RAS in time $4^k \,\mathrm{poly}(n)$ for k-paths. Section 4 is mainly expository and explains how previous approaches fit into our framework. The technical details needed to establish Theorems 1–3 in full generality are moved to the appendices.

Notation. Let G be a directed graph with n vertices. The set of vertices is $V(G)$ and enumerated as $\{v_1, \ldots, v_n\}$. The set of edges is $E(G)$, the edge from u to v is denoted uv. A sequence of vertices $w_1, \ldots, w_k$ in $V(G)$ such that $w_i w_{i+1} \in E$ holds for all $i \in \{1, \ldots, k-1\}$ is called a *k-walk* in G; we denote W by $w_1 \cdots w_k$. A walk of distinct vertices is called a *path*. The set of k-walks is denoted $\mathscr{W}(G)$ and the set of k-paths is denoted $\mathscr{P}(G)$.

We write $\mathrm{poly}(n)$ for the set of polynomially bounded functions in n. Since we may always assume $k \leq n$, we have $\mathrm{poly}(n+k) = \mathrm{poly}(n)$, and we can hence always omit reference to k.

2. All-Walks Extensor and Baseline Algorithm for Unambiguous Path

This section is primarily expository and contains no new algorithmic results. We begin with 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. We then present our main idea, namely a correspondence between paths in a graph and elements in the exterior algebra, and make the resulting algorithm explicit.

2.1. Concrete Definition and Properties of the Exterior Algebra

Consider the unit vectors $e_1, \ldots, e_k$ and a field F. Together, they define the usual k-dimensional vector space, which we call F^k. Every element a of F^k is a linear combination $a_1e_1 + \cdots + a_ke_k$ of unit vectors by field elements $a_1, \ldots, a_k \in F$. We sometimes write a as the column vector $(a_1, \ldots, a_k)$. 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 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 $\sum_{I \subseteq \{1,\ldots,k\}} a_I e_I$ 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 $x = 3e_{\{1,2\}} - 7e_{\{3\}}$ and $y = e_{\{1\}} + 2e_{\{3\}}$ and we have $x + 2y = 3e_{\{1,3\}} + 2e_{\{1\}} - 3e_{\{3\}}$. By confusing e_i with $e_{\{i\}}$ for $i \in \{1, \ldots, k\}$, we can view F^k as a subspace of $\Lambda(F^k)$ spanned by the singleton basis vectors.

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

$$(x \wedge y) \wedge z = x \wedge (y \wedge z), \qquad \text{for all } x, y, z \in \Lambda(F^k)$$

and bilinear

$$x \wedge (a \cdot y + z) = a \cdot x \wedge y + x \wedge z\,,\ (x + a \cdot y) \wedge z = x \wedge z + a \cdot y \wedge z\,, \qquad \text{for all } a, b \in F \text{ and } x, y, z \in \Lambda(F^k)\,,$$

so it suffices to define how $\wedge$ behaves on a pair of basis vectors e_I and e_J. If I and J contain a common element, then we set $e_I \wedge e_J = 0$. Otherwise, we set $e_I \wedge e_J = \pm e_{I \cup J}$; 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

$$e_I \cup e_J = (-1)^{\mathrm{sgn}\,(\pi)} e_{I \cup J}\,,$$

where π is the permutation that brings the sequence $i_1, \ldots, i_r, j_1, \ldots, j_s$ into increasing order.

For instance, if $\max I < \min J$ there is nothing to permute, so $e_1 \wedge e_2 = e_{\{1,2\}}$. Consequently, we now abandon the set-indexed notation $e_{\{i_1,\ldots,i_r\}}$ (where $i_1 < \cdots < i_r$) and just write $e_{i_1} \wedge \cdots \wedge e_{i_r}$ instead. It is also immediate that $e_1 \wedge e_2 = -e_2 \wedge e_1$. In general, we can multiply basis vectors using pairwise transpositions and associativity, *e.g.*,

$$(e_1 \wedge e_3 \wedge e_6) \wedge (e_2 \wedge e_4) = -e_1 \wedge e_3 \wedge e_2 \wedge e_6 \wedge e_4 = e_1 \wedge e_2 \wedge e_3 \wedge e_6 \wedge e_4 = -e_1 \wedge e_2 \wedge e_3 \wedge e_4 \wedge e_6\,.$$

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

1. *Anti-commutative on F^k.* For $x, y \in F^k$, elementary calculations show that $x \wedge y = -y \wedge x$. For $x \in F^k$, the requirement $x \wedge x = -x \wedge x$ implies $x \wedge x = 0$, so all squares in F^k vanish. This extends to products of elements from F^k: If $x_1 \wedge \cdots \wedge x_r$ contains the same element twice, then the whole expression vanishes.

2. *Wedge product on F^k is a determinant.* For $k = 2$ write $x, y \in F^2$ as column vectors (x_1, x_2) and (y_1, y_2). Elementary calculations show $x \wedge y = (x_1 y_2 - y_1 x_2) \cdot e_1 \wedge e_2$, and we recognise the determinant of the 2×2-matrix whose columns are x and y. By induction, we arrive at a fundamental property of the exterior algebra: Let

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

then

$$x_1 \wedge \cdots \wedge x_k = \det \begin{pmatrix} x_{11} & \cdots & x_{1k} \\ \vdots & \ddots & \vdots \\ x_{k1} & \cdots & x_{kk} \end{pmatrix} \cdot e_1 \wedge \cdots \wedge e_k \,. \tag{2}$$

To avoid a misunderstanding, neither of these two properties extends to all of $\Lambda(F^k)$. For instance, if $x = e_1 \wedge e_3 + e_2$ then $x \wedge x = (e_1 \wedge e_3 + e_2) \wedge (e_1 \wedge e_3 + e_2) = e_1 \wedge e_3 \wedge e_1 \wedge e_3 + e_1 \wedge e_3 \wedge e_2 + e_2 \wedge e_1 \wedge e_3 + e_2 \wedge e_2 = 0 - e_1 \wedge e_2 \wedge e_3 - e_1 \wedge e_2 \wedge e_3 + 0 = -2 \cdot e_1 \wedge e_2 \wedge e_3 \neq 0$.

2.2. Walk Extensors

We introduce the shorthand

$$\psi = e_1 \wedge \cdots \wedge e_k \,.$$

An *extensor coding* is a mapping $\xi \colon V(G) \to \Lambda(F^k)$ associating an extensor with every vertex of G. The mapping extends to the walks in G in the natural fashion: If W is a walk given by $w_1 \cdots w_k$, we set

$$\xi(W) = \xi(w_1) \wedge \cdots \wedge \xi(w_k) \,,$$

and finally define the *all walks extensor* $\xi(\mathscr{W}(G))$ as

$$\xi(\mathscr{W}(G)) = \sum_{W \in \mathscr{W}(G)} \xi(W) \,.$$

The extensor $\xi(\mathscr{W}(G))$ is a single element in $\Lambda(G)$. We are ready for our main lemma, which shows that if $V(G)$ is coded with extensors from F^k (rather than $\Lambda(F^k)$) then the all-walks-extensor resolves to a determinant sum over all *paths*.

Lemma 4. *If $\xi(v) \in F^k$ for all $v \in V(G)$ then*

$$\xi(\mathscr{W}(G)) = \sum_{w_1 \cdots w_k \in \mathscr{P}(G)} \det \Xi \cdot \psi \,,$$

where Ξ is the $k \times k$ matrix with columns $\xi(w_1), \ldots, \xi(w_k)$.

Proof. Using property (2), we obtain

$$\xi(w_1 \cdots w_k) = \xi(w_1) \wedge \cdots \wedge \xi(w_k) = \det \Xi \cdot \psi \,,$$

Now, if the walk is not a path, then there is a pair $i, j \in \{1, \ldots, k\}$ with $i \neq j$ but $\xi(w_i) = \xi(w_i)$, so Ξ is singular and therefore its contribution vanishes. □

2.3. Baseline Algorithm

The only algorithmic observation we now need is that $\xi(\mathscr{W}(G))$ can be computed in poly(n) many algebraic operations over $\Lambda(F^k)$. But this is just the well-known connection between the kth power of the adjacency matrix and k-walks, sometimes expressed as dynamic programming. To recall this argument and fix notation, construct the $n \times n$ matrix A with ijth entry given by

$$a_{ij} = \begin{cases} \xi(v_i)\,, & \text{if } v_i v_j \in E\,; \\ 0\,, & \text{otherwise}\,. \end{cases} \tag{3}$$

Then an easy induction argument shows that for $r \in \{1, \ldots, k\}$, the ijth entry of A^r, evaluated in $\Lambda(F^k)$, contains the value

$$\sum \xi(w_1) \wedge \cdots \wedge \xi(w_{r-1})\,,$$

where the sum is over all $(r-1)$-walks $w_1 \cdots w_{r-1}$ with $w_1 = v_i$ and $w_{r-1} = v_j$. In particular,

$$\xi(\mathscr{W}(G)) = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}^T A^k \begin{pmatrix} \xi(v_1) \\ \vdots \\ \xi(v_n) \end{pmatrix}. \tag{4}$$

Now, from Lemma 4 is clear that $\xi(\mathscr{W}(G)) = 0$ if there are no k-paths in G. On the other hand, if G has exactly one k-path, then Lemma 4 show that $\xi(\mathscr{W}(G))$ simplifies to the single term $(\det \Xi) \cdot \psi$. Thus $\xi(\mathscr{W}(G)) \neq 0$, provided Ξ is nonsingular. We can guarantee the latter by letting ξ map to Vandermonde vectors.

We have arrived at fast algorithm for unambiguous k-path detection.

Algorithm U (*Detect an 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 ξ.) For each $i \in \{1, \ldots, n\}$, let $\xi(v_i) = i^0 e_1 + \cdots + i^{j-1} e_j + \cdots + i^{k-1} e_k$. [So $\xi(v_i)$ belongs to F^k and can be viewed as the column vector $(1, i, i^2, \ldots, i^{k-1})$.]

U2 (Set up A.) Define the $n \times n$ matrix A as in (3).

U3 (Compute all-walks extensor.) Compute $\xi(\mathscr{W}(G))$ as in (4).

U4 (Decide.) If $\xi(\mathscr{W}(G))$ is nonzero, then return 'yes.' Otherwise, return 'no.' □

Theorem 5. *Algorithm U is a deterministic algorithm for the unambiguous k-path problem with running time 2^k poly(n).*

Proof. Consider the extensor x computed in Step U3. If G contains no k-path then x vanishes. Otherwise, write the unambiguous k-path in G as $v_{i_1}, \ldots, v_{i_k}$. By our choice of ξ in U1, Lemma 4 shows that x has the form $d \cdot \psi$ with

$$d = \det \begin{pmatrix} 1 & 1 & \ldots & 1 \\ i_1 & i_2 & \ldots & i_k \\ i_1^2 & i_2^2 & \ldots & i_k^2 \\ \vdots & \vdots & \ddots & \vdots \\ i_1^{k-1} & i_2^{k-1} & \ldots & i_k^{k-1} \end{pmatrix} = \prod_{1 \leq j < l \leq k} (i_l - i_j)\,,$$

where the factorisation is a fundamental property of Vandermonde vectors. In particular, the matrix is nonsingular, so its determinant is nonzero and U4 returns 'yes.'

The running time of Algorithm U is clearly dominated by the calculation (4). At this point, it is not immediately clear that the operations in $\Lambda(F^k)$ can be performed using only 2^k field operations. A thorough (but entirely pedestrian) argument for this fact is made later, in Section D.1. Accepting this for a moment, the running time of Algorithm U is $2^k \operatorname{poly}(n)$. □

3. Tensor squares and Approximative Counting

In this section, we introduce the algebra $\Lambda(F^k)^{\otimes 2}$, and arrive at an FPT-RAS for k-path counting.

3.1. The tensor square of the exterior algebra

The F-algebra $\Lambda(F^k)^{\otimes 2}$ is the *tensor square* of the exterior algebra. Its basis is given by formal expressions $e_I \otimes e_J$, where e_I and e_J are any two of the 2^k basis vectors of $\Lambda(F^k)$, and the formal product $\otimes$ is bilinear in both operands. Multiplication of basis vectors is performed component-wise, *i.e.*, $(e_I \otimes e_J) \boxtimes (e_{I'} \otimes e_{J'}) = (e_I \wedge e_{I'}) \otimes (e_J \wedge e_{J'})$. This is extended bilinearly.

For elements of the form $x \otimes y$ with $x, y \in \Lambda(F^k)$, multiplication works as

$$(x \otimes y) \boxtimes (x' \otimes y') = (x \wedge x') \otimes (y \wedge y') \,.$$

We write the square $(x \otimes x)$ as $x^{\otimes 2}$.

In summary, three different algebraic objects are in play, with different names for their objects. Two of these spaces are algebras with a multiplication operator, using different symbols:

Notation	Name	Objects	Basis	dim	Product
F^k	Vector space	Vectors	$\{ e_i \mid 1 \le i \le k \}$	k	None
$\Lambda(F^k)$	Exterior algebra of F^k	Extensors	$\{ e_I \mid I \subseteq [k] \}$	2^k	$\wedge$
$\Lambda(F^k)^{\otimes 2}$	Tensor-square of $\Lambda(F^k)$	Tensor products	$\{ e_I \otimes e_J \mid I, J \subseteq [k] \}$	4^k	$\boxtimes$

3.2. Main Lemma

As before, let $\xi \colon V(G) \to \Lambda(F^k)$ be an extensor-coding of $V(G)$. Unlike before, define a mapping $\xi^{\otimes 2}$ from $\mathscr{W}(G)$ to $\Lambda(F^k)^{\otimes 2}$ (rather than $\Lambda(F^k)$), as follows. For a walk W given by the vertex sequence $w_1 \cdots w_k$, set

$$\xi^{\otimes 2}(W) = \xi(w_1)^{\otimes 2} \boxtimes \cdots \boxtimes \xi(w_k)^{\otimes 2} \,,$$

and extend this mapping to all of $\mathscr{W}(G)$ in the natural fashion:

$$\xi^{\otimes 2}(\mathscr{W}(G)) = \sum_{W \in \mathscr{W}(G)} \xi^{\otimes 2}(W) \,.$$

Analogously to Sec. 2.2, the value $\xi^{\otimes 2}(\mathscr{W}(G))$ is a single element in $\Lambda(F^k)^{\otimes 2}$, and its value depends entirely on $\mathscr{P}(G)$, the set of *paths*.

Lemma 6. *Let ξ denote an extensor-coding with $\xi(v) \in F^k$ for all $v \in V(G)$. Then,*

$$\xi^{\otimes 2}(\mathscr{W}(G)) = \sum_{w_1 \cdots w_k \in \mathscr{P}(G)} \det \Xi^2 \cdot \psi^{\otimes 2} \,, \tag{5}$$

where Ξ denotes the $k \times k$ matrix with columns $\xi(w_1), \ldots, \xi(w_k)$.

Proof. Expanding the definition of $\boxtimes$ and using property (2) of the wedge product, we obtain

$$\xi^{\otimes 2}(w_1 \cdots w_k) = \xi(w_1)^{\otimes 2} \boxtimes \cdots \boxtimes \xi(w_k)^{\otimes 2} = (\xi(w_1) \wedge \cdots \wedge \xi(w_k))^{\otimes 2} = (\det \Xi \cdot \psi)^{\otimes 2},$$

which equals $\det \Xi^2 \cdot \psi^{\otimes 2}$ because $\otimes$ is bilinear. As before, whenever $w_1 \cdots w_k$ contains a repeated vertex, then Ξ is singular and the contribution vanishes. □

Note that the coefficient to $\psi^{\otimes 2}$ is a square and therefore nonnegative.

3.3. Deterministic Algorithm for Path Detection

We quickly imitate Algorithm U to arrive at a deterministic algorithm for k-path detection in the ambiguous case. The matrix A from (3) is now defined as

$$a_{ij} = \begin{cases} \xi(v_i)^{\otimes 2}, & \text{if } v_i v_j \in E; \\ 0, & \text{otherwise}. \end{cases} \tag{6}$$

and the matrix computation becomes

$$\xi^{\otimes 2}(\mathscr{W}(G)) = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}^T A^k \begin{pmatrix} \xi(v_1)^{\otimes 2} \\ \vdots \\ \xi(v_n)^{\otimes 2} \end{pmatrix}. \tag{7}$$

We have arrived at a deterministic algorithm for k-path, even slightly improving upon the time bound of $4^{k+o(k)} \cdot \mathrm{poly}(n)$ of Chen *et al.* [16, 15], but not coming close to the record bound $2.5961^k \cdot \mathrm{poly}(n)$ of Zehavi [64].

Theorem 7 (Superseded by [64])**.** *There is a deterministic algorithm that, given a directed graph G, checks if G has a path of length k in time $4^k \cdot \mathrm{poly}(n)$.*

Proof. The algorithm is just Algorithm U, replacing the construction (3) in Step U2 by (6) and the computation (4) in Step U3 by (7). The running time has a factor 4^k because the arithmetic operations in $\Lambda(F^k)^{\otimes 2}$ can be performed in that time, as spelt out in Appendix D. □

3.4. Randomized Approximate Counting of Paths

We present our algorithm for approximated counting. The idea is to let the matrices Ξ appearing in Lemma 10 be random matrices so that the random variables $\det \Xi^2$ all have the same mean. In expectation, the sum of their squared determinants then depends on the size of $\mathscr{P}(G)$. Our technical challenge is to bound their variance.

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

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

C2 (Set up jth trial.) For each $i \in \{1, \ldots, n\}$, let $\xi(v_i)$ be a column vector of k values chosen from ± 1 independently and uniformly at random. Define the $n \times n$ matrix A as in (6).

C3 (Compute scaled approximate mean X_j.) Compute $x = \xi^{\otimes 2}(\mathscr{W}(G))$ from A and ξ using (7). Let X_j be the coefficient of $\psi^{\otimes 2}$ in x.

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

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

Theorem 8. *For any $\varepsilon > 0$, Algorithm C produces in time $(4^k/\varepsilon^2) \cdot \mathrm{poly}(n)$ a value X such that with probability at least 99%, we have*

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

The proof consists of a bound on the fourth moment of a random determinant, see Appendix A.

4. Connection to Previous Work

In this section, we show how our approach using exterior algebras specializes to the group algebra approach by Koutis and Williams [41, 43, 61] 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.

4.1. 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 $R[M]$. Letting 1_M be the unit of M, we call the coefficient of 1_M the *constant part* of 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 $+$. This is justified in the following way: As an R-algebra, $R[M]$ is also a ring. One can see that M embeds as a submonoid of the multiplicative monoid of the ring $R[M]$ by the inclusion $\iota_M : M \hookrightarrow R[M]$ with $m \mapsto 1_R \cdot m$ that sends group elements to their corresponding unit vectors. Indeed, $\iota_M(M) \cong M$ as monoids. Similarly, R lies in $R[M]$ as a subring, witnessed by the inclusion $\iota_R : R \hookrightarrow R[M]$ with $r \mapsto r \cdot 1_M$ that sends ring elements r to the monoid algebra element with constant part r. Again we have $\iota_R(R) \cong R$ as rings. We may thus understand M and R as a submonoid and a subring of $R[G]$, respectively. Note that $\iota_R(1_R) = \iota_M(1_M) = 1_R \cdot 1_M$, which is the identity of $R[M]$. In this sense, both 1_R and 1_M can be considered as the multiplicative identity of $R[M]$, and we can thus identify $1_R = 1_M = 1_{R[M]}$, and simply write 1.

Proposition 9. *Let F be of characteristic two and F^k the free vector space of dimension k with basis $\{e_1, \ldots, e_k\}$. Then, the group algebra $F[\mathbb{Z}_2^k]$ is isomorphic to $\Lambda(F^k)$.*

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

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, 61].

Exterior Algebras as Group Algebras

We can shed a little light from the perspective of group algebras on the construction of an exterior algebra (even in the general setting of characteristic 0) as follows: For a natural number k, consider the free monoid E^* over the generators $E := \{e_1, \ldots, e_k, \mu, \theta\}$, foreshadowing a little bit by naming the generators just like the standard basis of F^k. We impose now the following relations on this monoid: The element θ shall act as a zero, *i.e.*, $\theta x = x\theta = \theta$ for all $x \in E^*$, and the element μ shall be central, *i.e.*, $\mu x = x\mu$ for all $x \in E^*$. Furthermore, we shall have for all i that $e_i^2 = \theta$. Additionally, we demand that $e_i e_j = \mu e_j e_i$ and $\mu^2 = 1_E$. Write $\sim$ for the congruence generated by these equalities, and let $S := E^*/\sim$ be the resulting quotient monoid. Consider now the monoid algebra $F[S]$, and let I_S be the ideal generated by θ and $\mu + 1$. In $F[S]/I_S$, we then have the identities $\theta = 0$ and $\mu = -1$, which in particular entails that $e_i^2 = 0$ and $e_i e_j = -e_j e_i$ in $R[S]/I_s$. In other words, $F[S]/I_S$ is *precisely* the exterior algebra over F^k. Hence, we do not have to resort to understanding the above-mentioned group algebra as a special case of an exterior algebra, but can as well understand any exterior algebra as the homomorphic image of some monoid algebra.

4.2. 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 labelling 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 [61]. This is nicely laid out in the textbook by Cygan *et al.* [21, Section 10.4], and the following presentation is guided by theirs.

Let x_e be variables associated with each directed edge $e \in E(G)$, and let $y_i^{(j)}$ with $j \in \{1, \ldots, k\}$ be variables 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 x_e and $y_i^{(j)}$:

$$P(x,y) = \sum_{w_1 \cdots w_k \in \mathscr{W}(G)} \sum_{\ell \in S_k} \prod_{i=1}^{k} x_{w_i w_{i+1}} \prod_{i=1}^{k} y_i^{\ell(i)}. \tag{8}$$

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

$$P(x,y) = \sum_{w_1 \cdots w_k \in \mathscr{P}(G)} \sum_{\ell \in S_k} \prod_{i=1}^{k} x_{w_i w_{i+1}} \prod_{i=1}^{k} y_i^{\ell(i)}. \tag{9}$$

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 $Y(w_1 \cdots w_k) := (y_{w_j}^{(i)})_{i,j}$ indexed by pairs of numbers and vertices, and we can write

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

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 (2), where the variables $y_i^{(j)}$ form the entries of the vectors $\xi(w_i) \in F^k$ that are used for the extensor coding in Algorithm U, of course with additional weights to bring the monomials in general position with high probability and avoiding cancellations. Employing these weights is equivalent to the algebraic fingerprinting of Koutis [41] and Williams [61].

4.3. Color-Coding

For completeness, we will briefly discuss how the color-coding-technique by Alon, Yuster, and Zwick [4] can be seen as a special case of extensor-coding: When designing our deterministic algorithm in Section 3.3, we evaluate at tensor squares of extensors, $\xi(v_i)^{\otimes 2}$, chosen just as in Algorithm U. Now, if we let $\xi(v_i)$ not be *arbitrary* general position elements from F^k, but restrict the choice for the coding of v_i to the standard basis $\{e_i \mid 1 \leq i \leq k\}$ of unit vectors in F^k, then we arrive at a much simpler, commutative subalgebra of $\Lambda(F^k)^{\otimes 2}$ where we can perform arithmetic operations in time $2^k \cdot \mathrm{poly}(n)$ instead of $4^k \cdot \mathrm{poly}(n)$. However, the probability of success (*i.e.*, not making non-zero products vanish) is not 1 anymore as with our choice of $\xi(v_i)$ being in general linear position, but again drops to the well-known e^{-k}. Combining this with the cost of arithmetic in the algebra, we reproduce the classic $(2e)^k \cdot \mathrm{poly}(n)$. This is detailed in a similar setting in Appendix C.

Acknowledgments. We thank Markus Bläser, Radu Curticapean, Balagopal Komarath, Ioannis Koutis, Pascal Schweitzer, Karteek Sreenivasaiah and Meirav Zehavi for some valuable discussions and insights.

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] 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.

[11] 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.

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

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

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

[29] 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.

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

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] Mark Jerrum and Kitty Meeks. The parameterised complexity of counting connected subgraphs and graph motifs. *J. Comput. Syst. Sci.*, 81(4):702–716, 2015. doi:10.1016/j.jcss.2014.11.015.

[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] 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.

[46] Kitty Meeks. The challenges of unbounded treewidth in parameterised subgraph counting problems. *Discrete Appl. Math.*, 198:170–194, 2016. doi:10.1016/j.dam.2015.06.019.

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

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

[53] 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.

[54] 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.

[55] 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.

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

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

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

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

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

[61] 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.

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

[63] 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.

[64] 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.

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

[66] 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. Proof of Theorem 8

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 10 ([49])**.** *Let B be a Bernoulli matrix of dimension $k \times k$. Then,*

$$\mathbf{E} \det B^2 = k! \tag{10}$$

$$\mathbf{E} \det B^4 \leq (k!)^2 \cdot k^3 \,. \tag{11}$$

For completeness, we include a careful proof for a slightly different distribution in Appendix A.1.

Proof of Theorem 8. Run algorithm C with $t = 100k^3/\varepsilon^2$. Set $\mu = |\mathscr{P}(G)|$. Recall from Lemma 6 that X_j can be written as

$$X_j = \det \Xi_1^2 + \det \Xi_2^2 + \cdots + \det \Xi_\mu^2 \,, \tag{12}$$

where for $i \in \{1, \ldots, \mu\}$, each Ξ_i is a submatrix of of the $k \times n$ matrix with columns $\xi(v_1)$, $\xi(v_2)$,$\cdots$, $\xi(v_n)$. By our choice of ξ in Step C2, each Ξ_i is therefore a Bernoulli matrix, but they are not independent.

By Theorem 10, we have $\mathbf{E} \det \Xi_i^2 = k!$ for each $i \in \{1, \ldots, \mu\}$, so by linearity of expectation,

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

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

$$\operatorname{Cov}(\det \Xi_i^2, \det \Xi_l^2) = \sqrt{(\operatorname{Var} \det \Xi_i^2) \cdot (\operatorname{Var} \det \Xi_l^2)} = \sqrt{(\operatorname{Var} \det \Xi_i^2)^2} = \operatorname{Var} \det \Xi_i^2 \leq \mathbf{E} \det \Xi_i^4 \leq (k!)^2 k^3 \,,$$

where the last two inequalities uses $\operatorname{Var} Y \leq \mathbf{E}\, Y^2$ with $Y = \det \Xi_i^2$ and (11) in Theorem 10 with $B = \Xi_i$. We obtain

$$\operatorname{Var} X_j = \operatorname{Cov}(X_j, X_j) = \operatorname{Cov}\Big(\sum_{i=1}^{\mu} \det \Xi_i^2, \sum_{l=1}^{\mu} \det \Xi_l^2\Big) = \sum_{i,l=1}^{\mu} \operatorname{Cov}(\det \Xi_i^2, \det \Xi_l^2) \leq \mu^2 \cdot (k!)^2 \cdot k^3 \,.$$

Now consider X 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 $\operatorname{Var}(a \cdot X) = a^2 \cdot \operatorname{Var}(X)$ for a random variable X and a scalar a, by independence of the X_j, we have

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

Now Chebychev's inequality gives

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

which implies the stated bound. □

A.1. Higher Moments of the Determinant of a Random Matrix

Expressions for the higher moments of determinants of random matrices are available in the literature since the 1950s, see [49] and there references therein. Such results are considered routine, and follow from solving exercise 5.64 in Stanley [57] or following the general method laid out on page 45–46 in Girko's book [30], 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 here.

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 linearity of the determinant, we can write $(\det B)^r = \det B^r$.

To see

$$\mathbf{E} \det B^2 = k! \tag{13}$$

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 (13).

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

$$\sum_i (-1)^{4i}\, \mathbf{E}\, b_{1i}^4\, \mathbf{E} \det B_{1i}^4 = 3k\, \mathbf{E} \det B_{11}^4 = 3k f_{k-1}\,. \tag{14}$$

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

$$3 \sum_{i \neq j} (-1)^{2i+2j}\, \mathbf{E}\, b_{1i}^2\, \mathbf{E}\, b_{1j}^2\, \mathbf{E} \det(B_{1i}^2 B_{1j}^2) = 3k(k-1)\, \mathbf{E} \det(B_{11}^2 B_{12}^2)\,. \tag{15}$$

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 deleting from B 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

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

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

$$\mathbf{E}\det(B_{11}^2 B_{12}^2) = (k-1)f_{k-2} + (k-1)(k-2)\Phi\,. \tag{16}$$

Combining (14), (15), and (16), we obtain

$$f_k = 3kf_{k-1} + 3k(k-1)^2\big(f_{k-2} + (k-2)\Phi\big)\,. \tag{17}$$

Using similar arguments from a different starting point, we obtain

$$f_{k-1} = \mathbf{E}\det B_{11}^4 = 3(k-1)f_{k-2} + 3(k-1)(k-2)\Phi\,, \tag{18}$$

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 (17) and (18) 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 now readily employ the distribution we discussed in this section in the FPT-RAS from Section 3.4. The only thing we have to keep in mind is that we still have to be able to perform arithmetic operations in the field. This is clear in the case of the distribution in Section 3.4 with the values $+1$ and -1, but our use of irrational numbers here might create some confusion. This confusion is quickly resolved, however, when noting that we only have to calculate with values coming from the field extension $\mathbb{Q}[\sqrt{3}]$, which can be handled just like complex numbers in spirit (after all, $\mathbb{C} = \mathbb{R}[\sqrt{-1}]$) by representing a number $a + b\sqrt{3}$ by the two rational coordinates a, b, and performing multiplication according to $(a + b\sqrt{3})(c + d\sqrt{3}) = ac + 3bd + (ad + bc)\sqrt{3}$.

B. Generalization to subgraph counting and detection via the homomorphism polynomial

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 3.4. 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:

$$\sum_{h \in \mathrm{Hom}(H \to G)} \prod_{v \in V(H)} \zeta_{h(v)} \,. \tag{19}$$

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. Dynamic programming on tree decompositions

Fomin *et al.* [27, Lemma 1] construct an algebraic circuit that computes the homomorphism polynomial, based on a standard dynamic programming algorithm (see, *e.g.*, [23]). We reproduce a proof here for completeness.

Lemma 11. *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*

$$C(\zeta_1, \ldots, \zeta_n) = \sum_{h \in \mathrm{Hom}(H \to G)} \prod_{v \in V(H)} \zeta_{h(v)} \,. \tag{20}$$

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

Before we prove this lemma, we formalize some preliminaries on tree decompositions. A *tree decomposition* of a graph G is a pair (T, β), where T is a tree and β is a mapping from $V(T)$ to $2^{V(G)}$ such that, for all vertices $v \in V(G)$, the set $\{\, t \in V(T) : \; v \in \beta(t) \,\}$ is nonempty and connected in T, and for all edges $e \in E(G)$, there is some node $t \in V(T)$ such that $e \subseteq \beta(t)$. The set $\beta(t)$ is the *bag at t*. The *width* of (T, β) is the integer $\max\{\, |\beta(t)| - 1 : \; t \in V(T) \,\}$, and the *treewidth* $\mathrm{tw}(G)$ of a graph 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 $\sigma, \gamma, \alpha : V(T) \to 2^{V(G)}$ for all $t \in V(T)$:

$$\text{(the \textit{separator at t})} \qquad \sigma(t) = \begin{cases} \emptyset & \text{if } t \text{ is the root of } T, \\ \beta(t) \cap \beta(s) & \text{if } s \text{ is the parent of } t \text{ in } T, \end{cases} \tag{21}$$

$$\text{(the \textit{cone at t})} \qquad \gamma(t) = \bigcup_{u \text{ is a descendant of } t} \beta(u), \tag{22}$$

$$\text{(the \textit{component at t})} \qquad \alpha(t) = \gamma(t) \setminus \sigma(t). \tag{23}$$

Proof of Lemma 11. We first compute a minimum-width tree decomposition (T, β) of H, for example using the $O(1.76^k)$ time algorithm by Fomin and Villanger [28]. Without loss of generality, we assume it to be a *nice* tree decomposition, in which each node has at most two children; the leaves satisfy $\beta(v) = \emptyset$, the nodes with two children w, w' satisfy $\beta(v) = \beta(w) = \beta(w')$ and are called *join* nodes, the nodes with one child w satisfy $\beta(v) = \beta(w) \cup \{x\}$ and are called *introduce* nodes, or $\beta(v) \cup \{x\} = \beta(w)$ and are called *forget* nodes.

Recall that $\gamma(v)$ 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.

$$\hom(\, H \to G \mid \pi \,) = \sum_{\substack{h \in \mathrm{Hom}(H \to G) \\ h \supseteq \pi}} \prod_{v \in V(H)} \zeta_{h(v)} \,. \tag{24}$$

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

$$\hom(H \to G) = \sum_{\pi \in \mathrm{Hom}(H[S] \to G)} \hom(\, H \to G \mid \pi \,) \tag{25}$$

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

$$\hom(\, H \to G \mid \pi \,) = \prod_i \hom(\, H_i \to G \mid \pi \,), \tag{26}$$

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 $\pi \in \mathrm{Hom}(\beta(v) \to G)$ such that the following holds:

$$C_v^\pi = \hom(\, H[\gamma(v)] \to G \mid \pi \,). \tag{27}$$

Note already here that there are at most $n^{|\beta(v)|} \leq n^{\mathrm{tw}(H)+1}$ 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. In this case, we have $\gamma(v) = \emptyset$, resulting in the trivial circuit $C_v^\pi = 1$ for the unique empty function $\pi : \emptyset \to V(G)$. Indeed, since $H[\gamma(v)]$ 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, $\beta(w) \cup \{x\} = \beta(v)$. Let $\pi \in \mathrm{Hom}(H[\beta(v)] \to G)$ be a partial homomorphism at the bag of v. Then we define C_v^π using the circuit $C_w^{\pi'}$ where $\pi' = \pi \restriction_{\beta(w)}$:

$$C_v^\pi = C_w^{\pi'} \cdot \zeta_{\pi(x)}. \tag{28}$$

For the correctness, note that the right side of (30) is equal to

$$\hom(\, H[\gamma(w)] \to G \mid \pi' \,) \cdot \zeta_{\pi(x)} \tag{29}$$

by the induction hypothesis (27). Since x is the unique vertex in $\gamma(v) \setminus \gamma(w)$ and π extends π' on x, the polynomial in (29) is equal to $\hom(\, H[\gamma(v)] \to G \mid \pi \,)$.

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, $\beta(w) \setminus \{x\} = \beta(v)$. Then the neighborhood of x is contained in $\gamma(w)$. Let $\pi \in \mathrm{Hom}(H[\beta(v)] \to G)$. We define C_v^π using the circuits $C_w^{\pi'}$ as follows:

$$C_v^\pi = \sum_{\pi'} C_w^{\pi'}. \tag{30}$$

The sum is over all $\pi' \in \mathrm{Hom}(H[\beta(w)] \to H)$ that agree with π on the intersection $\beta(v) \cap \beta(w)$ of their domains. Since v is a forget node, this intersection is equal to $\beta(v)$. For the correctness, note that the right side of (28) is equal to

$$\sum_{\pi'} \hom(\, H[\gamma(w)] \to G \mid \pi' \,) \tag{31}$$

by the induction hypothesis (27). This is equal to $\hom(\, H[\gamma(v)] \to G \mid \pi \,)$ due to the conditioning formula (25). For the size bound, note that x is the only vertex in $\beta(w) \setminus \beta(v)$. Thus the sum in (30) 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, that is, it has exactly two children w and w' with $\beta(v) = \beta(w) = \beta(w')$. Let $\pi \in \mathrm{Hom}(H[\beta(v)] \to G)$. Then we define the circuit simply as

$$C_v^\pi = C_w^\pi \cdot C_{w'}^\pi \,. \tag{32}$$

For the correctness, note that $\beta(v)$ is a separator for $H[\gamma(v)]$, and so the induction hypothesis (27) together with the conditional independence (26) 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 $\beta(r) = \emptyset$. 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 (28) 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%,*

$$(1-\varepsilon) \cdot \mathrm{Sub}(H, G) \leq \tilde{N} \leq (1+\varepsilon) \cdot \mathrm{Sub}(H, G) \,. \tag{33}$$

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

Proof sketch. Let H, G, and $\varepsilon > 0$ be given as input. Let n be the number of vertices of G. By Lemma 11, 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:

$$C(\zeta_1, \ldots, \zeta_n) = \sum_{h \in \mathrm{Hom}(H \to G)} \prod_{v \in V(H)} \zeta_{h(v)} \,. \tag{34}$$

Following exactly the setup of §3.4, we define an extensor-coding $\xi : V(G) \to \Lambda(F^k)$ of $V(G)$. In complete analogy to Lemma 6, we notice that

$$C(\xi(v_1)^{\otimes 2}, \ldots, \xi(v_n)^{\otimes 2}) = \sum_{h \in \mathrm{InjHom}(H \to G)} \prod_{v \in V(H)} \xi(h(v))^{\otimes 2} \,, \tag{35}$$

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 $\xi^{\otimes 2}(\mathscr{W}(G))$ with $C(\xi(v_1)^{\otimes 2}, \ldots, \xi(v_n)^{\otimes 2})$. The rest goes through as in Theorem 8. Note that this approach using (35) 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 [45], 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 $\Lambda(F^k)^{\otimes 2}$ takes time $O(4^k)$. Thus the overall running time is $O(\varepsilon^{-2} 4^k |C|)$. □

A proof of Theorem 2 follows precisely along the same lines as the proofs of Theorem 1 and Theorem 5: Instead of evaluating the circuit computing the homomorphism polynomial over $\Lambda(F^k)^{\otimes 2}$ as in the proof of Theorem 1, we evaluate it over $\Lambda(F^k)$. The same arguments used to prove Theorem 5 then suffice to prove the time bounds and correctness claim of Theorem 2.

C. Proof of Theorem 3

In this section, we formally prove Theorem 3. This will follow by an application of color-coding. For consistency, we will formulate it in the framework of exterior algebras.

Definition 12. The subalgebra of $\Lambda(F^k)^{\otimes 2}$ generated by the pure squares of the basis vectors of F^k, $\{e_{\{i\}} \otimes e_{\{i\}} \mid 1 \leq i \leq k\}$ is denoted by $Z(F^k)$. Sometimes, this is called the *Zeon-algebra*.

This algebra previously played a role in an algebraic algorithm for cycle detection by Schott and Staples [54].

Lemma 13. *$Z(F^k)$ is commutative and of dimension 2^k. Furthermore, addition and multiplication can be performed in $2^k \cdot \mathrm{poly}(n)$ field operations.*

Proof. Again, we confuse singletons $\{i\}$ with elements i. Then,

$$(e_i \otimes e_i) \boxtimes (e_j \otimes e_j) = (e_i \wedge e_j \otimes e_i \wedge e_j) = -(e_j \wedge e_i \otimes e_i \wedge e_j) = \\ -\left(-(e_j \wedge e_i \otimes e_j \wedge e_i)\right) = (e_j \wedge e_i \otimes e_j \wedge e_i) = (e_j \otimes e_j) \boxtimes (e_i \otimes e_i)\,,$$

and therefore $Z(F^k)$ is commutative. Furthermore, it is readily verified that the elements $e_I \otimes e_I$ with $I \subseteq [k]$ form a basis of $Z(F^k)$. By renaming $e_i \otimes e_i$ 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.*, [59]), 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. □

The next proposition is now a trivial consequence.

Proposition 14. *An arithmetic circuit C over $\mathbb{Z}[\zeta_1, \ldots, \zeta_n]$ and a number can be evaluated over $Z(\mathbb{Q}^t)$ in $2^t \cdot |C| \cdot \mathrm{poly}(n)$ operations over $\mathbb{Q}$.*

We are now ready to state:

Proof sketch of Theorem 3. We will invoke Proposition 14 with Hüffner *et al.*'s [32] choice of $t = 1.3k$. One evaluation will then cost $2.4623^k \cdot |C| \cdot \mathrm{poly}(n)$ operations over $\mathbb{Q}$. The classical color-coding approach would now correspond to evaluation C at the generators $e_i \otimes e_i$, where $1 \leq i \leq k$ is chosen uniformly at random. In this way, all non-multilinear terms will vanish, but of course, distinct monomials might cancel when being mapped to the same product of generators. To avoid this, we scale each generator by a random weight, and plug in $\alpha_i \cdot e_j \otimes e_j$ at the ith input of the circuit, for random $\alpha_i \in \{0, 1, \ldots, 100 \cdot k\}$ and random $1 \leq j \leq k$. The circuit C then evaluates to some multiple of $\psi^{\otimes 2}$, and the coefficient of $\psi^{\otimes 2}$ in the result is a multilinear polynomial in the $\alpha_i, 1 \leq i \leq n$. By the DeMillo–Lipton–Schwartz–Zippel-Lemma [22, 56, 66] the polynomial will evaluate non-zero with constant probability of 99%. Following Hüffner *et al.* [32, Theorem 1], the probability that some multilinear term is mapped to a multiple of the k distinct generators X_i is at least $\Omega(1.752^{-k})$, and thence we derive the total running time of $4.32^k \cdot |C| \cdot \mathrm{poly}(n)$ operations in $\mathbb{Q}$. Now, if the circuit can be evaluated over $\mathbb{Z}$ in polynomial time (*i.e.*, all numbers stay of appropriate size), then this will also only cost a polynomial overhead when evaluating over $Z(\mathbb{Q}^k)$. Therefore, the claim follows. However, by repeated squaring, the circuit C may well generate numbers of value up to 2^{2^n}, so we need to resort to calculating modulo some randomly chosen prime number. Of course, numbers of bitlength $O(2^n)$ may have up to roughly $O(2^n)$ prime factors, so choosing a random

prime p from the first $\Omega(n2^n)$ primes, we will succeed in finding a value for p such that the resulting coefficient doesn't vanish modulo p with probability $1 - o(1)$. By the prime number theorem, the first $n2^n$ primes are of magnitude $2^n \operatorname{poly}(n)$, and we can thus just randomly pick a number from the set $\{1, \dots, P\}$, where $P = 2^n \operatorname{poly}(n)$, until we find a prime number (which can be easily tested in randomized polynomial time). Then we perform all the above calculations modulo p, and if the result doesn't vanish, we know for sure it wouldn't have vanished over $\mathbb{Z}$. On the other hand, if it vanishes, we might have had bad luck, but as argued before, this only happens with probability $1/n$, which is fine for our purposes. □

D. Complexity of Arithmetic and Algebraic Circuits

D.1. Complexity of arithmetic operations

In computations involving elements of the F-algebra $\Lambda(F^k)^{\otimes 2}$, we assume they are given in the standard basis representation over $\Lambda(F^k)^{\otimes 2}$, that is, as a 4^k-dimensional vector with entries from F. In particular, for every element $a \in \Lambda(F^k)^{\otimes 2}$, we store an entry $a_{S,T} \in F$ for all $S, T \subseteq \{1, \dots, k\}$. The following lemma gives upper bounds on the cost of arithmetic operations in the algebra, expressed in terms of operations in the underlying field F.

Lemma 15. *Given two elements $u, v \in \Lambda(F^k)^{\otimes 2}$, we can compute*

1. *(Sum) their sum $u \boxplus v$ with $4^k \cdot \operatorname{poly}(n)$ arithmetic operations in F.*
2. *(Skew product) their product $u \boxtimes v$ with $4^k \cdot \operatorname{poly}(n)$ arithmetic operations in F if either u or v is equal to a pure tensor $p \otimes p$ for some $p \in F^k \subseteq \Lambda(F^k)$.*

Proof. The sum $\boxplus$ can be computed simply by the standard point-wise sum of the vector representations of u and v, taking one addition operation in F for each of the 4^k coordinates. This shows the first claims.

The skew product $u' = u \boxtimes (p \otimes p)$ can be computed more quickly because $p \otimes p$ is represented as a sparse vector in F^{4^k} that has at most k^2 non-zero entries. Thus the naïve multiplication using the distributive law has at most k^2 terms for each entry of u', so u' can be computed using $4^k \cdot k^2$ arithmetic operations over F. □

Remark. We conjecture that $O^*(2^{\omega k})$ operations are possible for performing general products in $\Lambda(F^k)^{\otimes 2}$. This could follow using a similar approach as that of Włodarczyk [63] by making use of an embedding of $\Lambda(F^k)^{\otimes 2}$ into a matrix algebra of dimension $2^{k/2} \times 2^{k/2}$ via the Wedderburn–Artin-Theorem.

For completeness, we also state the version for the mere exterior algebra $\Lambda(F^k)$.

Lemma 16. *Given two elements $u, v \in \Lambda(F^k)$, we can compute*

1. *(Sum) their sum $u + v$ with $2^k \cdot \operatorname{poly}(n)$ arithmetic operations in F.*
2. *(Skew product) their product $u \wedge v$ with $2^k \cdot \operatorname{poly}(n)$ arithmetic operations in F if either u or v is equal to a vector p for some $p \in F^k \subseteq \Lambda(F^k)$.*

Proof. The proof goes precisely along the same lines as the proof of Lemma 15, in particular, we employ a similar sparseness argument for skew multiplication. □

D.2. Algebraic circuits

We define algebraic circuits and some useful properties.

Definition 17. Let $\mathcal{A}$ be an algebra with addition $+$ and multiplication $\cdot$. An *algebraic circuit C over $\mathcal{A}$* with input variables $\zeta_1, \ldots, \zeta_n$ is a directed acyclic graph with vertex set $V(C)$, called *gates*, and arc set $E(C)$, called *wires*. The unique gate with out-degree one is called the *output gate*. Gates with in-degree zero are called *input gates*, and they are labeled either with a variable from $\{\zeta_1, \ldots, \zeta_n\}$ or a constant $\tau \in \mathcal{A}$. Every other gate has in-degree two, and is either labeled as a *plus gate* or a *product gate*. We write $C(\zeta_1, \ldots, \zeta_n)$ to denote the polynomial computed at the output gate. We define the *size of C* as the number $|V(C)| + |E(C)|$ of gates and wires. The circuit C is called *skew* if at least one child of every multiplication gate is an input gate.

Clearly, any circuit C can be evaluated at a point $(\tau_1, \ldots, \tau_n) \in \mathcal{A}^n$ to yield the element $C(\tau_1, \ldots, \tau_n)$ by using $|C|$ arithmetic operations over $\mathcal{A}$. In the case of the F-algebra $\Lambda(F^k)^{\otimes 2}$, this gives rise to the following corollary to Lemma 15.

Lemma 18. *Let $C(\zeta_1, \ldots, \zeta_n)$ be a skew circuit over the F-algebra $\Lambda(F^k)^{\otimes 2}$. Given pure tensors $\tau_1, \ldots, \tau_n \in F^k \otimes F^k \subseteq \Lambda(F^k)^{\otimes 2}$, then the only arithmetic operations used are addition and skew multiplication in $\mathcal{A}$, we can compute $C(\tau_1, \ldots, \tau_n) \in \Lambda(F^k)^{\otimes 2}$ using at most $4^k|C|$ arithmetic operations in F. Finally, $F = \mathbb{Q}$ holds, the bitlength of $C(\tau_1, \ldots, \tau_n) \in \mathbb{Q} \cdot \psi$ and the intermediate results at each gate are bounded by $O(|C| \cdot \max_i |\tau_i|)$, where $|\tau_i|$ denotes the length of the binary encoding of the element τ_i*

We omit the simple proof. However, let us briefly discuss the claim on the bitlength of the intermediate results in skew-circuits over $\mathbb{Q}$: For addition, the sum $a + b$ of two rationals has at most one bit more than the maximum of the bitlengths of a and b. At skew multiplication gates, we encounter products ab where a has polynomial length and b has length $\max_i |\tau_i|$ at most, so the bitlength of the product increases at most by $O(\max_i |\tau_i|)$. Since this occurs at most a linear number of times, the total maximum bitlength of each rational is $O(|C| \cdot \max_i |\tau_i|)$ To see now that Lemmas 15 and 16 enable us to perform the operations in (4) resp. (7) in the claimed time bound, observe that the computation of the iterated matrix powers only ever involved skew multiplications, since by the choice of the $\xi(v_i)$ and $\xi(v_i)^{\otimes 2}$, the matrix A only contains vectors from F^k and pure tensors from $F^k \otimes F^k$, respectively. This can be implemented by setting up and evaluating an arithmetic circuit over the respective algebra that is of size $O(k \cdot (n + m))$.