Limits and Applications of Group Algebras for Parameterized Problems*

Ioannis Koutis
Computer Science Department
U. of Puerto Rico, Rio Piedras
ioannis.koutis@upr.edu

Ryan Williams
Computer Science Department
Stanford University
rrw@cs.stanford.edu

Abstract

The fastest known randomized algorithms for several parameterized problems are based on reductions to the k-MLD problem: detection of multilinear monomials of degree k in polynomials presented as circuits. The algorithm for k-MLD is purely algebraic and uses only oracle accesses to an arithmetic circuit, i.e. simple evaluations of the circuit on elements from a suitable algebra. In this paper we use communication complexity to show that the aforementioned algorithm is essentially optimal within this evaluation oracle framework. On the positive side, we give new applications of the method: finding a copy of a given tree on k nodes, a minimum set of nodes that dominate at least t nodes, and an m-dimensional k-matching. In each case we achieve a faster algorithm than what was known before. We also apply the algebraic method to problems in exact counting. Among other results, we show that a variation of it can break the trivial upper bounds for the disjoint summation problem.

1 Introduction

The central topic of this article is the parameterized multilinear monomial detection problem.

k-MLD: Given an arithmetic circuit C representing a polynomial $P(X)$ over $\mathbb{Z}$, decide whether $P(X)$ construed as a sum of monomials contains a multilinear monomial of degree k.

Here, an arithmetic circuit is a directed acyclic graph with nodes corresponding to addition and multiplication gates, sources (with indegree zero) corresponding to variables, and one terminal (with outdegree zero) corresponding to the output gate. The k-MLD problem is arguably a fundamental parameterized problem. Several parameterized problems are reducible to it, and the fastest known algorithms for many of them are based on the reduction. Notable examples are the k-path problem on directed graphs and packing k sets of size m [16, 23].

The fastest known algorithm for k-MLD is purely algebraic [16, 23]. It first constructs an 'extended' version $\tilde{C}$ of C and labels certain edges of $\tilde{C}$ with random multipliers from a field $\mathbb{F}$. It then chooses a particular commutative algebra $\mathcal{A}$ and an appropriately defined randomized assignment $X \rightarrow \mathcal{A}$ such that (i) squares (and by commutativity, non-multilinear monomials) evaluate to the zero element of $\mathcal{A}$, and (ii) some multilinear monomial does not evaluate to zero, with good probability. Finally it evaluates $\tilde{C}$ on the assignment. By construction the output is non-zero with good probability if and only if the input instance of k-MLD is positive.[1]

*A preliminary version of this article appeared in ICALP'09 [17].

[1] **Note:** The proof of [23] was given in the context of the k-path problem and omitted the construction of $\tilde{C}$. We give the missing details in Section 2.

This algebraic framework yields an $O^*(2^k)$ time algorithm for k-MLD, by choosing $\mathbb{F}$ to be $GF(2^{3+\log_2 k})$ and $\mathcal{A}$ to be the group algebra $\mathbb{F}[\mathbb{Z}_2^k]$ (for the definition of a group algebra, see the Appendix).[2] The algorithm can be implemented in polynomial space.

Applications. As a rule of thumb, the k-MLD framework improves upon all parameterized decision problems that previously solvable using the color coding method [4] and the divide-and-color method [14, 10]. However, depending on the problem, the details of reducing the problem to k-MLD can be tricky and sometimes requires extensions to the basic framework. To support this claim, Section 3 gives faster algorithms for: (i) finding a copy of a given tree on k nodes, (ii) finding a minimum set of nodes that dominate at least t nodes in a graph. We also present a faster algorithm for finding an m-dimensional k-matching, by presenting a tighter reduction to k-MLD.

Limits of group algebras. A faster algorithm for k-MLD would have tremendous implications, not only in parameterized but also exact in algorithms, as it would imply faster algorithms for problems where progress has stagnated for nearly 50 years; an example is the Hamiltonian Path problem on directed graphs. Thus, an intriguing and natural question is whether k-MLD can be solved faster, by evaluating circuits over a more exotic $\mathcal{A}$ supporting faster operations over the circuit.[3]

The answer is, unfortunately, negative. In Section 4 we use communication complexity to show that for any commutative algebra $\mathcal{A}$ used to evaluate the circuit C, the lengths of elements in $\mathcal{A}$ must be at least $\Omega(2^k/k)$ in order to solve k-MLD.[4] Thus the $O^*(2^k)$ algorithm for k-monomial detection is optimal in a certain rigorous sense, and further progress on the relevant parameterized problems or k-MLD itself will require different kinds of operations altogether.

Applications in counting. Although group algebras are powerful for k-MLD, their potential has not been explored in the context of the related *counting* problem:

(k,n)-MLC: Given a commutative arithmetic circuit C describing an n-variate polynomial $P(X)$, compute the sum of the coefficients of the degree-k multilinear monomials in $P(X)$.

Unlike the k-MLD problem, in the counting problem we specify the number of variables n too. This is because the problem is #W[1]-hard and so an $O^*(f(k))$ time algorithm is unlikely to exist for it [11]. The hardness of (k,n)-MLC follows from the parsimonious reduction of the k-path problem to it [16] in combination with the hardness of the k-path problem [12].

There is an $O^*(\binom{n}{k})$ time algorithm for the (k,n)-MLC problem, but breaking below this barrier is a significant open problem. Alon and Gutner [3] showed that the there is no $o^*(n^{k/2})$ time algorithm for (k,n)-MLC based on color-coding; however this theoretical limit hasn't been met. In Section 5.1 we present partial progress showing how to solve (k,n)-MLC modulo 2, in $O^*(n^{k/2})$ time.

Next, we consider special versions of counting, starting with disjoint summation in Section 5.2:

Disjoint Summation: Given two n-variate polynomials P and Q each being a sum of multilinear monomials of total degree $k/2$, with coefficients from a ring $\mathbb{R}$, find the sum of the coefficient of the multilinear monomials in the product PQ.

The disjoint summation problem is largely motivated by the problem of counting exactly the number of k-paths. Before our result it was known how to count $2k$-paths in $O^*\left(\binom{n}{k}\right)$ time [7]. For the disjoint summation problem we present an algorithm that performs $O^*(n^{k/2})$ operations. As a result we get an $O^*(n^{\lceil mk/2 \rceil})$ time algorithm for counting k-packings of m-sets. Our result was later improved in [8], but our techniques are different and may have the potential to solve more difficult problems.

[2] Following standard conventions, the $O^*()$ notation hides polynomial factors in the instance size.

[3] Recently, progress *has* been made on solving Hamiltonian Path in undirected graphs by Björklund [6]; his techniques are inspired by those in this work.

[4] We state our result for commutative algebras, for the sake of clarity and coherence with our algorithmic results that all use commutativity. However, our lower bound holds in the non-commutative setting as well, under the appropriate definitions.

We finally consider the (k,k)-MLC problem, which can be used as a subroutine in a number of recent algorithms for approximate parameterized counting [5, 19, 2, 3]. We show that (k,k)-MLC can be solved in $O^*(2^k)$ time and polynomial space, a fact that seems to have been missed in the literature. We therefore reduce the space complexity of all the aforementioned approximate counting algorithms.

We extend our techniques for (k,k)-MLC to the computation of the $k \times n$ matrix permanent. We derive an $O^*(2^k)$ time polynomial space algorithm for the $k \times n$ permanent of matrices over rings, and an $O^*(2^k)$ time and space algorithm of matrices over commutative semirings. To the best of our knowledge the previously[5] fastest algorithms use $O^*(2^k)$ space and run in $O^*(3^k)$ time over rings [13], and in $O^*(4^k)$ time over commutative semirings [22]. The faster permanent algorithms imply speedups in algorithms for counting weighted subgraphs [22]. The algebraic perspective behind our algorithm allows us to derive an alternative formula for the $n \times n$ permanent, which to the best of our knowledge is new.

2 Background Results

2.1 Generalizing the k-path algorithm to k-MLD

Here we briefly describe how to generalize the multilinear monomial detection algorithm of [23] to arbitrary arithmetic circuits over $\mathbb{Z}$. The algorithm in [23] multiplies each edge e coming out of a gate in C with a new variable z_e which is drawn from an additional set of variables Z. The proof assumes that the coefficient of each multilinear monomial in the resulting polynomial $P(X,Z)$ is 1.

This assumption is true for the k-path circuit, but doesn't hold in general.[6] Nevertheless, it is relatively easy to massage any given circuit into one in which the assumption does hold (while preserving the multilinearity of the monomials in the circuit). In particular we can efficiently convert any circuit into one with the following properties:

(i) Addition gates and multiplication gates alternate, i.e. the inputs of each multiplication gate are outputs of addition gates and vice-versa (here, source nodes labeled by variables can be considered as addition gates).

(ii) The fan-out of each addition gate is 1.

(iii) Every scalar in the circuit is either 0 or 1.

To see why these properties of C imply the desired property for $P(X,A)$, observe that multilinear terms in $P(X)$ are in a one-to-one correspondence with "valid" connected subcircuits of C. (Here 'valid' means that in the subcircuit, each addition gate has fan-in 1, and each multiplication gate has full fan-in.) Each valid subcircuit C_s is completely specified by the set E_s of edges incoming to its addition gates. This is because the fan-in of the addition gates in C_s is 1, so the input wires specify the addition gates; in turn, the fan-out of each edge in E_s is 1, which specifies all the multiplication gates in C_s. After introducing the variables z_e on all the edges of the circuit, the multilinear term corresponding to C_s in $P(X)$ appears in $P(X,Z)$ multiplied by the variables on the edges of E_s. Now consider two different multilinear terms of C corresponding to two different valid subcircuits C_s and C'_s of C. Since C_s and C'_s are different and the multiplication gates have full fan-in, the sets E_s and E'_s must differ in at least one edge, so the corresponding multilinear terms in $P(X,Z)$ are different (one contains a variable z_e that the other does not). Therefore every multilinear monomial in $P(X,Z)$ has coefficient 1 (or 0, if not present).

[5] Similar results were derived using inclusion-exclusion techniques in [9] which cites the conference version of our results [17].

[6] For example, consider the polynomial $P(X) = (x+y)^2$, implemented in the usual way (one addition gate that takes x and y, and one multiplication gate that takes the output of the addition gate twice as input). If each edge out of a gate is multiplied by a new variable z_e, we obtain the polynomial $z_{e''}(z_e x + z_{e'} y) \cdot z_{e'''}(z_e x + z_{e'} y) = z_{e''} z_{e'''}(z_e^2 x^2 + 2xy z_e z_{e'} + y^2 z_{e'}^2)$ and the coefficient of $xy z_e z_{e'} z_{e''} z_{e'''}$ is not 1, but 2.

We can transform any circuit C over $\mathbb{Z}$ to an equivalent circuit $\tilde{C}$ that has the three properties above. First, we can set every non-zero scalar in C to 1; then our C trivially meets property (iii) above, without changing the multilinearity of the monomials in the circuit. The next step is to construct a circuit C' where addition and multiplication gates alternate, to meet property (i). This can be easily done in polynomial time: for each addition gate g find the set S_g of all multiplication gates and terminals that are reachable from g via paths that use only addition gates in C. Then replace g with a multiplication gate $\tilde{g}$ with inputs coming directly from the multiplication gates in S_g. After finishing with addition gates, we perform the analogous operation to multiplication gates, with the roles of addition and multiplication gates reversed. It is not hard to see that each new gate $\tilde{g}$ represents the same polynomial as g. Finally, to satisfy property (ii), we construct $\tilde{C}$ by replacing each addition gate g with a number of addition gates $g_1, \ldots, g_k$ equal to the fan-out k of g, where each gate g_i has the same inputs as g, and each g_i exactly one output (namely, g_i sends its output to one of the k gates that g sends its output).

Observe that the size of $\tilde{C}$ is at most quadratic in the size of C. The multilinear monomial detection algorithm of [23] applies then to $\tilde{C}$, yielding the same time and space upper bounds.

2.2 An extension for k-MLD

The main tool for obtaining faster randomized parameterized algorithms is the following slight generalization of our results in [16, 23].

Lemma 2.1. *Let $P(X,z)$ be a polynomial represented by a commutative arithmetic circuit C. The existence of a term of the form $z^t Q(X)$ in $P(X,z)$, where $Q(X)$ is a multilinear monomial of degree at most k, can be decided in time $O^*(2^k t \log t)$ and space $O^*(t)$.*

Proof. (Sketch). The results in [16, 23] are stated without the special variable z. It is fairly easy to modify the algorithm in order to handle the extra variable z. At a high level the algorithm works by summing the outputs of 2^k evaluations of C over the ring of bivariate polynomials $\mathbb{Z}[z,u]$, modulo z^{t+1}. The answer is obtained from the coefficient of z^t in the sum. By letting $z = 1$ in the final sum we obtain exactly the output of the k-MLD algorithm of [23] so the modification amounts to just keeping z unevaluated and handling it symbolically. The $O^*(t)$ space is needed to represent the aforementioned bivariate polynomials, and the $O(t \log t)$ factor in the running time comes from using fast Fourier multiplication of univariate polynomials of degree at most t. We will omit further details here. □

3 Faster Parameterized Algorithms

Using the k-MLD framework we obtain faster randomized algorithms for the following problems:

k-Tree. Given a tree T on k nodes and a graph G on n nodes, decide if there is a (not necessarily induced) copy of T in G.

t-Dominating Set. Given a graph $G = (V,E)$, find a minimum set of nodes S that dominate at least t nodes in the graph. That is, $|S \cup N(S)| \geq t$ where $N(S) = \{v \mid (u,v) \in E, u \in S\}$.

m-Dimensional k-Matching. Given mutually disjoint sets U_i, for $i = 1, \ldots, m$, and a collection $\mathcal{C}$ of m-tuples from $U_1 \times \ldots \times U_m$, decide whether $\mathcal{C}$ contains a sub-collection of k mutually disjoint m-tuples.

Each of these can be solved by formulating them as k-MLD instances in some way. To the best of our knowledge, the only other algorithm we know for k-tree follows from the color-coding method [4], and runs in $O^*((2e)^k)$ time. The best known (randomized) algorithm for t-Dominating Set runs in time $O^*((4+\varepsilon)^t)$ [15]. The best known (randomized) algorithm for the k m-Dimensional Matching Problem

runs in time $O^*(2^{mk})$ [16]. In addition, in all these problems, given an algorithm for the decision version of the problem, standard reductions can be used to recover an algorithm for the search version, with the same exponential dependence on the parameter.

Theorem 3.1. *The k-Tree problem can be solved in $O^*(2^k)$ time.*

Proof. Suppose T is a k-node tree we wish to find in G. Let the nodes of T be $\{1,\ldots,k\}$, and let the nodes of G be $\{1,\ldots,n\}$. We define an arithmetic circuit $C_{T,i,j}(x_1,\ldots,x_n)$ inductively, for all $i \in [k]$ and $j \in [n]$.

- If $|V(T)| = 1$, simply define $C_{T,i,j} := x_j$.
- If $|V(T)| > 1$, let $T_{i,1},\ldots,T_{i,\ell}$ be the connected subtrees of T remaining after node i is removed from T. For all $t = 1,\ldots,\ell$, let $n_{i,t} \in [k]$ be the (unique) node in $T_{i,t}$ that is a neighbor of i in T. Define

$$C_{T,i,j} := \prod_{t=1}^{\ell} \left(\sum_{j':(j,j')\in E(G)} x_j \cdot C_{T_{i,t},n_{i,t},j'} \right).$$

Observe that the size of $C_{T,i,j}(x_1,\ldots,x_n)$ is at most $O(|V(T)| \cdot |E(G)|)$. The polynomial $C_{T,i,j}$ enumerates those homomorphisms that map nodes of T into nodes of G, such that node i in T is mapped to node j in G. Each monomial represents the range of some homomorphism. More precisely, the monomial $x_{j_1}\cdots x_{j_k}$ is present in the polynomial if and only if there is a homomorphism where the nodes of T are mapped to the vertices $\{j_1,\ldots,j_k\} \subseteq V$ of G. These mappings are all homomorphisms, since $i' \in V(T)$ can only be mapped to $j' \in V(G)$ if $(i,i') \in E(T)$, $(j,j') \in E(G)$, and $i \in V(T)$ is already mapped to $j \in V(G)$.

Now consider the sum-product expansion of $Q = \sum_{j\in V(G), i\in V(T)} C_{T,i,j}$. Q is a sum over all connected subgraphs S of G on k vertices, where each monomial corresponds to the range of some homomorphism from T into S. Note Q also includes homomorphisms that map distinct nodes of T to a common node in G. However, those homomorphisms correspond precisely to monomials with *squares* in them. That is, the multilinear monomials of Q correspond to homomorphisms that map all nodes in T to distinct nodes in G, i.e. an isomorphic copy of T in G. Therefore, by calling k-MLD on the arithmetic circuit defined by Q, we can detect whether G contains a copy of T in $O^*(2^k)$ time. □

Theorem 3.2. *The t-Dominating Set problem can be solved in $O^*(2^t)$ time.*

Proof. Let the vertex set be $V = \{1,\ldots,n\}$ and $X = \{x_1,\ldots,x_n\}$, where x_i corresponds to the vertex i. Consider the polynomial

$$P(X,z) = \left(\sum_{i\in V} \left((1 + z\cdot x_i) \cdot \prod_{j\,:\,(i,j)\in E} (1 + z\cdot x_j) \right) \right)^k,$$

where z is an extra indeterminate. P is a sum of monomials in which each monomial of the form $z^t x_{i_1}\cdots x_{i_t}$ for *distinct* i_j appears if and only if the t nodes $\{i_1,\ldots,i_t\}$ are dominated by a set of at most k nodes. In addition, every other term of the form $z^t Q(X)$ contains a square since $Q(X)$ has total degree t. Then, the proof follows from Lemma 2.1 and trials with increasing values of k. □

Theorem 3.3. *The m-Dimensional k-Matching problem can be solved in $O^*(2^{(m-1)k})$ time.*

Proof. Encode each element u in $U = \bigcup_{i=2}^{m} U_i$ by a variable $x_u \in X$. Encode each m-tuple $t = (u_1, \ldots, u_m) \in \mathcal{C} \subseteq U_1 \times \cdots \times U_m$ by the monomial $M_t = \prod_{i=2}^{m} x_{u_i}$. Assume $U_1 = \{u_{1,1}, \ldots, u_{1,n}\}$, and let $T_j \subseteq \mathcal{C}$ denote the subset of m-tuples whose first coordinate is $u_{1,j}$. Consider the polynomial

$$P(X, z) = \prod_{j=1}^{n} \left(1 + \sum_{t \in T_j} (z \cdot M_t) \right),$$

where z is an extra indeterminate. The coefficient of z^k is a polynomial $Q(X)$. There is a one-to-one correspondence between the terms of $Q(X)$ and collections of k m-tuples, where the m-tuples in each k-collection have mutually different first coordinates. Each term is the product of the $(m-1)k$ variables corresponding to the elements in the other $(m-1)$ coordinates of the m-tuples of each k-collection. Hence, each k-collection contributes a multilinear term if and only if the m-tuples in it form a k-matching. Therefore, the coefficient of z^k contains a multilinear $((m-1)k)$-term if and only if $\mathcal{C}$ contains a k-matching. The proof follows from Lemma 2.1. □

4 Lower Bound for Multilinear Detection

We now investigate whether the algebraic approach to k-MLD in [16, 23] can be improved upon further. One basic open question is whether is it possible to determine if an arbitrary arithmetic circuit C has a multilinear k-monomial in much less time than ${}^*O(2^k)$, by evaluating C over a more exotic algebraic structure. We shall prove that this is *not* the case. For any commutative $\mathcal{A}$, we show that if it can be used to detect multilinear monomials in an arithmetic circuit (even randomly), then $|\mathcal{A}| \geq 2^{\Omega(2^k/\sqrt{k})}$. This implies a lower bound of $\Omega^*(2^k/\sqrt{k})$. That is, the group algebras used in [16, 23] (which have $|\mathcal{A}| \leq 2^{O(2^k \log k)}$) are essentially optimal for their purpose.

At a high level, our proof uses hypothetical fast multilinear monomial detection in order to design communication protocols that are too efficient to exist. Let us informally recall some notions from communication complexity. Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$. Suppose two parties wish to compute $f(x, y)$, but one party is given x (the x-party), the other is given y (the y-party). The parties must communicate bits about their inputs in order to compute f. A *simultaneous public-coin protocol for f on n-bits* is specified by a pair of functions (g_1, g_2), and works as follows on all n-bit strings. Initially, the x-party and y-party see a common string z chosen at random from a distribution independent of x and y (to maximize the parties' capabilities, we assume the distribution is uniform). The x-party computes $g_1(x, z)$, the y-party computes $g_2(y, z)$, and both send their answers to a third party. We require that the third party holding the two messages can compute f (with high probability), on all x and y of length n. The *randomized public-coin simultaneous communication complexity of* (g_1, g_2) is the maximum length $L(n)$ of a message sent in the protocol (g_1, g_2) over all n-bit strings. The corresponding communication complexity of f is the minimum $L(n)$ achieved by any n-bit protocol (g_1, g_2) for f.

The set disjointness function $DISJ_n(x, y)$ on $x, y \in \{0,1\}^n$ is defined to be $\vee_{i=1}^{n}(x_i \wedge y_i)$. We utilize the following fact:

Theorem 4.1. *[[18],p.79] The randomized public-coin communication complexity of $DISJ_n$ is $\Omega(n)$.*

Theorem 4.2. *Let $\mathcal{A}$ be a commutative semiring. Suppose there is a distribution $\mathcal{D}$ on elements from $\mathcal{A}$, an element $e \in \mathcal{A}$, and constants $d_1, d_2 \in [0,1]$, $d_1 < d_2$ such that for every circuit $C(x_1, \ldots, x_n)$ representing a degree-k polynomial:*

- *C has a multilinear monomial $\Longrightarrow \Pr_{(e_1,\ldots,e_n)\in\mathcal{D}^n}[C(e_1, \ldots, e_n) = e] \leq d_1$*

- *C does not have a multilinear monomial* $\Longrightarrow \Pr_{(e_1,\ldots,e_n)\in\mathcal{D}^n}[C(e_1,\ldots,e_n)=e] \geq d_2$.

Then $|\mathcal{A}| \geq 2^{\Omega(2^k/\sqrt{k})}$. Furthermore, for every k there is a circuit C with $n = k$ for which this lower bound holds.

In other words, the group algebra utilized in [23] is optimal within poly(k) factors: any other algebra could only yield a slightly better asymptotic upper bound. As a consequence it is not possible to solve Hamilton Path much faster than $O(2^n)$ by solving k-MLD over a more interesting algebra.

Proof. Using multilinear detection, we design a protocol for $DISJ_N$. Let $N > 0$ and $k = n = \log N + \frac{1}{2}\log\log N + c$ where $c > 0$ is sufficiently large. Without loss of generality, assume k is even. Let $\mathcal{S} = \{S_1,\ldots,S_\ell\}$ be an intersecting set system over $[k]$ such that $|S_i| = k/2$, for all i, and $\ell \geq N$. That is, we have $S_i \cap S_j \neq \varnothing$ for all i, j. For example, we may take $\mathcal{S} = \{S \subseteq [k] \mid |S| = k/2 \;\&\; 1 \in S\}$. Observe that for this collection, when c is large enough we have

$$|\mathcal{S}| = \binom{k-1}{k/2} \geq \Omega\left(\frac{2^k}{\sqrt{k}}\right) = \Omega\left(\frac{2^{\log N+\frac{1}{2}\log\log N+c}}{\sqrt{\log N+\frac{1}{2}\log\log N+c}}\right) \geq N$$

by Stirling's inequality. Hence $\ell \geq N$.

We now give a protocol for set disjointness, assuming the existence of a good $\mathcal{A}$. Let $a, b \in \{0,1\}^N$. For all $i = 1,\ldots,\ell$, define the monomials

$$P_i = \prod_{j\in S_i} x_j \quad \text{and} \quad Q_i = \prod_{j\notin S_i} x_j.$$

Now define an arithmetic circuit

$$C(x_1,\ldots,x_k) = \left(\sum_{i=1}^{n} a_i P_i\right)\cdot\left(\sum_{i=1}^{n} b_i Q_i\right).$$

Note that C represents a homogeneous polynomial of degree k. We claim that C has a square-free monomial if and only if $DISJ_N(a,b) = 1$. This follows from the fact that the monomial $P_i \cdot Q_j$ has a square if and only if $i \neq j$.[7]

Let $C_a = \sum_{i=1}^n a_i P_i$ and $C_b = \sum_{i=1}^n b_i Q_i$. To get a communication protocol for set disjointness, the x-party uses public randomness to obtain $e_i \in \mathcal{A}$ for each x_i, computes $v = C_a(e_1,\ldots,e_k) \in \mathcal{A}$, and sends an $O(\log|\mathcal{A}|)$-bit string corresponding to v. Similarly, the y-party obtains all $e_i \in \mathcal{A}$, evaluates $w = C_b(e_1,\ldots,e_k)$, and sends w. The third party outputs *disjoint* if and only if $e = v \cdot w$ (where $e \in \mathcal{A}$ has the properties of the theorem's hypothesis).

Under the hypotheses of the theorem (and repeating the protocol $O(1)$ times to obtain a good estimate of $\Pr_{(e_1,\ldots,e_n)\in\mathcal{D}^n}[C(e_1,\ldots,e_n)=e]$), the above is a correct public-coin protocol for $DISJ_n$. It follows from Theorem 4.1 that $\log|\mathcal{A}| \geq cN$, for some constant $c > 0$. Finally, note that $N \geq \Omega\left(\frac{2^k}{\sqrt{k}}\right)$. □

5 Applications in Counting

In this section we give algorithms for exact counting of multilinear monomials. We will be making use of facts about group algebras presented in the Appendix. We also need the following (properly adapted)

[7] Note this is the portion of the argument where commutativity of multiplication is used. In the non-commutative setting, the condition is that the monomial $P_i \cdot Q_j$ contains two instances of the same variable in its product. This condition is precisely the same as that required for all the algorithmic applications.

theorem from the theory of error correcting codes [20][Chap. 5, Theorem 8] also used in the deterministic construction of k-wise probability spaces [1][Proposition 6.5].

Theorem 5.1. *There is a 0-1 matrix M with n columns and m rows with $2^m \leq 2(n+1)^{\lfloor k/2 \rfloor}$, such that every k columns of M are linearly independent over $\mathbb{Z}_2$. Moreover, the matrix can be constructed in $O(n^{k/2+1})$ time.*

5.1 The (k,n)-MLC mod 2 problem

Using the above matrices, we can readily give a deterministic algorithm computing a version of the multilinear monomial detection problem:

Theorem 5.2. *Let C be a circuit representing a polynomial $P(X)$ over $\mathbb{Z}$. The parity of the sum of the coefficients of the multilinear monomials of degree k in $P(X)$ can be computed in deterministic $O^*(n^{\lceil k/2 \rceil})$ time and polynomial space.*

Proof. Let m be the number of rows of the matrix M in Theorem 5.1. Note that $m \leq \frac{k}{2}\log n + 1$. Let z be an indeterminate. We define an assignment A as follows: $x_i \mapsto z(\mathbf{v_0} + M_i)$, where $M_i \in \mathbb{Z}_2^m$ is the i^{th} column of M, and $\mathbf{v_0} \in \mathbb{Z}_2^m$ is the zero vector. $P(A)$ is an element of the group algebra $\mathbb{R}[\mathbb{Z}_2^m]$ where $\mathbb{R}$ is the ring of univariate polynomials $\mathbb{Z}_2[z]$. So, the coefficient of v_0 in $P(A)$ is a univariate polynomial $Q(z)$. We claim that the coefficient of z^k in $Q(z)$ is equal modulo 2 to the desired quantity.

To see why we consider the properties of the assignment A, which was also used in [16, 23] with the only difference being the multiplication by z. By the results of [16], $\bar{X}$ has two properties. (i) Any non-multilinear term of $P(X)$ evaluates to $0 \ mod\ 2$. (i) Any multilinear monomial $t(X)$ of $P(X)$ with total degree d evaluates to an element of the form $z^d t(A)$ where $t(A)$ is an element of the group algebra $\mathbb{Z}_2[\mathbb{Z}_2^m]$. The coefficient of v_0 in $t(A)$ is 1. By the associativity and commutativity of addition in $\mathbb{R}[\mathbb{Z}_2^m]$ the monomial $t(X)$ contributes one copy of z^d to the coefficient of v_0 in $P(A)$.

This gives a correct algorithm: evaluate $Q(z)$ and return the coefficient of z^k in it. The proof for the running time and space claims is analogous to that of Lemma 2.1, with the only difference being the dimension of the algebra. □

5.2 The disjoint summation problem

Theorem 5.3. *Let $P(X), Q(X)$ be two n-variate polynomials over a ring $\mathbb{R}$, each consisting of multilinear monomials of degree $k/2$. The sum of the coefficients of the (degree k) multilinear terms in the product PQ, can be computed with $O^*(n^{\lceil k/2 \rceil})$ operations over $\mathbb{R}$.*

Proof. Let m be the number of rows of the matrix M in Theorem 5.1. Note that . The algorithm is as follows. Let $A : X \to \mathbb{R}[\mathbb{Z}_2^m]$ be the assignment $x_i \mapsto (\mathbf{v_0} + M_i)$, where $M_i \in \mathbb{Z}_2^m$ is the i^{th} column of M, and $\mathbf{v_0} \in \mathbb{Z}_2^m$ is the zero vector. Let $\bar{A}$ be the assignment $x_i \mapsto (\mathbf{v_0} - M_i)$. Compute $\Pi = P(A)Q(\bar{A})$ over $\mathbb{R}[\mathbb{Z}_2^m]$, and return the coefficient of $\mathbf{v_0}$ in Π.

Let us first consider the algorithm's correctness. Every term in PQ is a product of the form

$$\sigma_P \sigma_Q t_P(A) t_Q(\bar{A})$$

where t_P, t_Q are multilinear monomials in P and Q respectively, and σ_P, σ_Q are their coefficients. If t_P and t_Q both contain a variable x_i, then by commutativity $t_P(A) \cdot t_Q(\bar{A})$ is a multiple of $(\mathbf{v_0} + M_i)(\mathbf{v_0} - M_i)$

which is equal to 0, since $\mathbf{v_0}^2 = M_i^2 = \mathbf{v_0}$. If $t_P t_Q$ is multilinear, then because the columns of M_i are independent and by Lemma 2.2 of [16], the coefficient of $\mathbf{v_0}$ in $t_P(A) \cdot t_Q(\bar{A})$ equals $\sigma_P \sigma_Q$. This proves correctness.

Now we concentrate on the time complexity. We start with the evaluation of an individual term $t(A)$ of $P(A)$. By Lemma 2.2 of [16], we know that $t(A)$ is a sum of 2^k distinct vectors from $\mathbb{Z}_2^m$. Our goal is to compute the list L_t of these vectors, each requiring m bits of space for its description. We will grow the list L_t inductively. Assume we have computed the list $L_{t'}$ for $t'(A)$ where t' consists of the first k' factors of t. The list with the vectors in $t'(A)(\mathbf{v_0} + M_i)$ can be constructed fairly simply with $O^*(2^{k'})$ operations; for each vector v in $t'(A)$ append to the end of the list $L_{t'}$ the vector $M_i v$. Hence, $t(A)$ can be evaluated in sparse form with $\sum_{i=1}^{d} 2^i = O^*(2^k)$ operations. To compute $P(A)$, we note that it will be a weighted sum of the 2^m vectors of $\mathbb{Z}_2^m$. So its computation amounts to the computation of these 2^m coefficients. To do this we will keep an array with 2^m positions, storing the partial sum of the terms $t(A)$ we have computed so far. Every time a new term $t(A)$ is computed, we read its list L_t and we update accordingly the array, by adding σ_P in the 2^k positions corresponding to the vectors in L_t. The computation of $Q(A)$ is similar, with the only difference being that the vectors in the list L of a term will also have sign. Hence, $P(A)$ and $Q(A)$ can be computed with $O(2^{k/2}\binom{n}{k/2})$ operations, as there are at most $\binom{n}{k/2}$ monomials in P and Q. Finally, we want $P(A)Q(\bar{A})$. Since $P(A)$ and $Q(\bar{A})$ are both elements of $\mathbb{R}[\mathbb{Z}_2^m]$, their multiplication can be performed via a Fast Fourier Transform method with $O^*(2^m)$ operations [23]. □

5.3 The (k,k)-MLC problem

In this Section we consider a special case of the counting problem.

Theorem 5.4. *Let C be a circuit describing a polynomial $P(x_1, \ldots, x_k)$ over a ring $\mathbb{R}$. The coefficient σ of the multilinear monomial $x_1 \ldots x_k$ of degree k can be computed in time $O^*(2^k)$ and space $O(|C|)$.*

Proof. We first introduce an extra variable z and multiply each terminal/variable in C. It is clear that for the new polynomial $P(X,z)$ we have

$$P(X,z) = \sum_i z^i Q_i(X),$$

where $Q_i(X)$ is the sum of monomials of degree i in $P(X)$ (so $P(X) = \sum_i Q_i(X)$). We will find an assignment $\bar{X} : X \mapsto \mathbb{Z}_2[\mathbb{Z}_2^m]$ and evaluate $P(X,z)$ at $\bar{X}$. The output $P(\bar{X},z)$ will be an element of the group algebra $\mathbb{Z}_2[z][\mathbb{Z}_2^m]$. The assignment will be such that $P(\bar{X},z)$ encodes the desired value.

The assignment is $x_i \to e_i$. It is easy to verify that, since $\mathbf{e_i}^2 = \mathbf{v_0}$, any degree k non-multilinear monomial of $Q_k(X)$ to a vector v with strictly less than k ones, hence different from $\mathbf{j}$. Therefore the only monomial that evaluates to $\mathbf{j}$ is the multilinear monomial, and so its coefficient is equal to the desired quantity σ.

The algorithm needs only to compute the coefficient $Q(z)$ of $\mathbf{j}$ in $P(\bar{X})$. As discussed in the Appendix, $Q(z)$ appears in copies along the anti-diagonal of $\rho(P(\bar{X}))$. Let b_i denote the vector containing the k-bit binary form of i, and $\Lambda_i(P(\bar{X}))$ denote the i^{th} eigenvalue of $P(\bar{X})$. Using Equality 5 and the properties of the eigenvalue decompositions given in the Appendix, we have the identity

$$2^k Q(z) = \sum_{i=0}^{2^k-1} (-1)^{b_i^T \cdot \mathbf{j}} \Lambda_i(P(\bar{X})) = \sum_{i=0}^{2^k-1} (-1)^{b_i^T \cdot \mathbf{j}} P(\Lambda_i(\bar{X})) = \sum_{i=0}^{2^k-1} (-1)^{b_i^T \cdot \mathbf{j}} P(z(-1)^{\mathbf{e_1}^T b_i}, \ldots, z(-1)^{\mathbf{e_n}^T b_i}). \tag{1}$$

Equality 1 reduces the problem to 2^k evaluations of C with univariate real polynomials and provides an $O(|C|)$ space algorithm to evaluate $Q(z)$, from which the target coefficient can be read off. □

5.4 Application to permanents

Let $A = [a_{ij}]$ be a $k \times n$ matrix with real coefficients. The permanent of A is defined as

$$perm(A) = \sum_{\sigma:[k]\to n,\sigma \ is \ 1-1} \left(\prod_{i=1}^{k} a_{i,\sigma(i)} \right) \tag{2}$$

Theorem 5.5. *Let A be a $k \times n$ matrix with entries from a commutative semi-ring $\mathbb{S}$. The permanent of A can be computed in time and space $O^*(2^k)$, assuming a unit cost for the addition and multiplication operations over $\mathbb{S}$. If $\mathbb{S}$ is a ring $\mathbb{R}$, the space complexity can be reduced to $poly(n)$.*

Proof. Viewing $perm(A)$ as a function of the entries of A, it is clear that it is a polynomial consisting only of degree k monomials. To form one term in $perm(A)$, we first pick -among $\binom{n}{k}$ possibilities- a set $\mathcal{I}$ of k columns in A to form a $k \times k$ submatrix $A_{\mathcal{I}}$, and then picking -among $k!$ possibilities- k elements from $A_{\mathcal{I}}$ one from each different row and column of A.

We claim that $perm(A)$ is the degree k multilinear term of the k-variate polynomial

$$P(X) = \prod_{i=1}^{n} \left(1 + \sum_{j=1}^{k} a_{i,j} x_j \right). \tag{3}$$

To see why, note that every term of degree k in $P(X)$ appears as a term in a product of the form

$$\prod_{i\in\mathcal{I}} \sum_{j=1}^{k} a_{i,j} x_j, \tag{4}$$

where $\mathcal{I}$ is a set of k distinct numbers from $[1,n]$. This corresponds to picking k columns from A. Considering now the polynomial in expression 4, we can see that the coefficient of each of its k^k monomials is the product of k elements that belong to distinct columns of $A_{\mathcal{I}}$. However, if a monomial contains two elements from the same row j of $A_{\mathcal{I}}$, then it is a product of x_j^2. Hence, all multilinear monomials of polynomial 4 correspond to sets of k elements, one from each different row and column of A. This proves the claim.

The proof for the case $S = \mathbb{R}$ follows now from Theorem 5.4. Let us now consider the general semiring case. Let P be the polynomial in Equality 3. Let $\mathcal{R}(Q)$ denote the operation that restricts the polynomial Q to its multilinear terms (i.e. deletes all terms containing squares) Let P_j be the polynomial consisting of the multilinear terms in the product of the first j factors of P. It is clear that we are interested in $\mathcal{R}(P_n)$. We can compute this by using n times the simple property $\mathcal{R}(P_{j+1}) = \mathcal{R}(\mathcal{R}(P_j)P)$. Since P_j is k-variate, $\mathcal{R}(P_j)$ has at most 2^k terms. In addition, each factor of P has n terms. Hence, the multiplication and deletion of the non-multilinear terms required to compute $\mathcal{R}(\mathcal{R}(P_j)P)$ can be done in $O(n2^k)$ time. □

We note that the application of Equality 1 in the original polynomial $P(X)$ of Equality 3 recovers Ryser's formula. However, note that for the case $k = n$, the polynomial can be simplified to

$$P(X) = \prod_{i=1}^{n} \sum_{j=1}^{n} a_{i,j} x_j$$

In this case all terms of $P(x)$ have degree exactly n, and thus we can set $z = 1$ in Equality 1 to get the following Theorem.

Theorem 5.6. *Let n_i denote the number of ones in the n-bit binary form $b(i)$ of i. Let P_i denote the set of the positions of ones in $b(i)$. We have*

$$2^n perm(A) = \sum_{i=0}^{2^n-1} (-1)^{n_i} \prod_{i=1}^{n} \left(\sum_{j \notin P_i} a_{ij} - \sum_{j \in P_i} a_{ij} \right).$$

6 Final Remarks

A number of interesting questions remain open:

- Theorem 5.1 can be used to get a weak derandomization of our earlier results, and solve more sophisticated problems. To what extent can we improve on the construction of M in this result?
- Is there an algorithm for multilinear k-monomial counting that improves over the naive upper bounds for *general* circuits?
- Can the algebraic method be applied to improve the time complexity for *approximate* counting? For this question, color coding is still the best known approach.

7 Appendix A

7.1 Group algebras of $\mathbb{Z}_2^k$

Let $\mathbb{Z}_2^k$ be the group consisting of k-dimensional 0-1 vectors with the group multiplication being entry-wise addition modulo 2. The group algebra $R[\mathbb{Z}_2^k]$, where R is a ring, is the set of all linear combinations of the form $\sum_{v \in \mathbb{Z}_2^k} a_v v$ where $a_v \in K$. The addition operator of $R[\mathbb{Z}_2^k]$ is defined by

$$\sum_{v \in \mathbb{Z}_2^k} a_v v + \sum_{v \in \mathbb{Z}_2^k} b_v v = \sum_{v \in \mathbb{Z}_2^k} (a_v + b_v) v.$$

Multiplication by a scalar $\alpha \in R$ is defined by

$$\alpha \sum_{v \in \mathbb{Z}_2^k} a_v v = \sum_{v \in \mathbb{Z}_2^k} (\alpha a_v) v.$$

The multiplication operator of $R[\mathbb{Z}_2^k]$ is defined by

$$\left(\sum_{v \in \mathbb{Z}_2^k} a_v v \right) \cdot \left(\sum_{u \in \mathbb{Z}_2^k} b_u u \right) = \sum_{v,u \in \mathbb{Z}_2^k} (a_v b_u)(uv).$$

It can be verified that $R[\mathbb{Z}_2^k]$ is commutative, provided that R is commutative.

Matrices of dimension d with real entries form a group $M^{d \times d}$ with matrix multiplication and an algebra $\mathcal{M}^{d \times d}$ with matrix addition, multiplication by a scalar, and matrix multiplication. It is well known ([21], p. 172), that there is a one-to-one map $\rho : \mathbb{Z}_2^k \to M^{2^k \times 2^k}$ such that $\rho(uv) = \rho(u)\rho(v)$. The map ρ is an isomorphism. For $\mathbb{Z}_2$, the map $\rho : \mathbb{Z}_2 \to M^{2 \times 2}$ is defined by the representations of the $\mathbb{Z}_2$ elements:

$$\rho(0) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \text{ and } \rho(1) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

If $\rho(\mathbb{Z}_2^k)$ denotes the set of matrix representations of the $\mathbb{Z}_2^k$ elements, it is easy to prove that

$$\rho(\mathbb{Z}_2^k) = \bigcup_{X \in \rho(\mathbb{Z}_2^{k-1})} \{ \begin{pmatrix} X & 0 \\ 0 & X \end{pmatrix}, \begin{pmatrix} 0 & X \\ X & 0 \end{pmatrix} \}.$$

Among the elements of $\mathbb{Z}_2^k$, the representation of the identity $\mathbf{v_0}$ is the identity matrix, and the representation of the all ones vector $\mathbf{j}$ is the matrix who has ones in the anti-diagonal and zeros everywhere else.

Let $\mathbb{R}[z]$ denote the ring of univariate polynomials with real coefficients. The map ρ can be extended to a one-to-one map of $\mathbb{R}[z][\mathbb{Z}_2^k]$ to $\mathcal{M}^{2^k \times 2^k}$, as follows:

$$\rho(\sum_{v \in \mathbb{Z}_2^k} a_v v) = \sum_{v \in \mathbb{Z}_2^k} a_v \rho(v).$$

It can be verified that if w_1, w_2 are elements of $\mathbb{R}[\mathbb{Z}_2^k]$ and $\alpha \in \mathbb{R}$, we have $\rho(w_1 + w_2) = \rho(w_1) + \rho(w_2)$, $\rho(w_1 w_2) = \rho(w_1)\rho(w_2)$ and $\rho(\alpha w_1) = \alpha \rho(w_1)$. Hence, the map ρ defines an isomorphic matrix algebra which we will denote by $\rho[\mathbb{R}[z][\mathbb{Z}_2^k]]$. An easy consequence of the above considerations is that for any element μ of $\mathbb{R}[z][\mathbb{Z}_2^k]$, the anti-diagonal of $\rho(\mu)$ contains copies of the coefficient of $\mathbf{j}$ in μ.

The matrices $\rho(\mathbb{Z}_2^k)$ are simultaneously diagonalizable, i.e. there is a unitary matrix U such that for all $v \in \mathbb{Z}_2^k$, we have $\rho(v) = U^{-1}\Lambda_v U$, where Λ_v are the eigenvalues of $\rho(v)$, also known as the characters of v. If $b(i)$ is the vector containing the k-bit binary form of i, the i^{th} eigenvalue of $\rho(v)$ is given by $(-1)^{v^T b(i)}$ [21]. The matrix $2^k U$ contains only 1 and -1 entries. Let $P(X)$ be any polynomial with real coefficients, and $\bar{X}$ be an assignment $X \to \mathbb{R}[z][\mathbb{Z}_2^k]$. If Λ_i denotes the i^{th} eigenvalue, it is not hard to verify that

$$\Lambda_i(P(\bar{X})) = P(\Lambda_i(\bar{X})).$$

This implies that for any element μ we can write $\rho(\mu) = U^{-1}\Lambda_\mu U$. From this it can be seen that each element of the anti-diagonal of $\rho(\mu)$ is an identical linear combination of the eigenvalues of $\rho(\mu)$. In fact, if $a_{\mathbf{j}}$ is the coefficient of $\mathbf{j}$ in μ, it can be shown that

$$2^k a_{\mathbf{j}} = \sum_{i=0}^{2^k-1} (-1)^{\mathbf{j}^T b(i)} \Lambda_i(\mu). \tag{5}$$

This is an analog of the trace identity, where $\mathbf{j}$ is substituted by $\mathbf{v_0}$.

References

[1] Noga Alon, László Babai, and Alon Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *J. Algorithms*, 7:567–583, December 1986.

[2] Noga Alon, Phuong Dao, Iman Hajirasouliha, Fereydoun Hormozdiari, and S. Cenk Sahinalp. Biomolecular network motif counting and discovery by color coding. *Bioinformatics*, 24(13):241–249, 2008.

[3] Noga Alon and Shai Gutner. Parameterized and exact computation. chapter Balanced Hashing, Color Coding and Approximate Counting, pages 1–16. Springer-Verlag, Berlin, Heidelberg, 2009.

[4] Noga Alon, Raphael Yuster, and Uri Zwick. Color coding. *Journal of the ACM*, 42(4):844–856, 1995.

[5] Vikraman Arvind and Venkatesh Raman. Approximation algorithms for some parameterized counting problems. In *ISAAC*, pages 453–464, 2002.

[6] Andreas Björklund. Determinant sums for undirected Hamiltonicity. In *Proc. 51st IEEE Symposium on Foundations of Computer Science (FOCS '10)*, pages 173–182, 2010.

[7] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. The fast intersection transform with applications to counting paths. *CoRR*, abs/0809.2489, 2008.

[8] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Counting paths and packings in halves. In *ESA 2009, 17th Annual European Symposium on Algorithms*, pages 578–586, 2009.

[9] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Evaluation of permanents in rings and semirings. *Inf. Process. Lett.*, 110(20):867–870, 2010.

[10] Jianer Chen, Songjian Lu, Sing-Hoi Sze, and Fenghui Zhang. Improved algorithms for path, matching, and packing problems. In *Proc. 18th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 298–307, 2007.

[11] Rod G. Downey and Mike R. Fellows. *Parameterized Complexity*. Springer, 1999.

[12] Jörg Flum and Martin Grohe. The parameterized complexity of counting problems. In *FOCS '02: Proceedings of the 43rd Symposium on Foundations of Computer Science*, page 538, Washington, DC, USA, 2002. IEEE Computer Society.

[13] Tsutomu Kawabata and Jun Tarui. On complexity of computing the permanent of a rectangular matrix. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, E82-A5:741–744, 1999.

[14] Joachim Kneis, Daniel Mölle, Stefan Richter, and Peter Rossmanith. Divide-and-color. In Fedor Fomin, editor, *Graph-Theoretic Concepts in Computer Science*, volume 4271 of *Lecture Notes in Computer Science*, pages 58–67. Springer Berlin / Heidelberg, 2006.

[15] Joachim Kneis, Daniel Mölle, and Peter Rossmanith. Partial vs. complete domination: t-dominating set. In *Proceedings of the 33rd conference on Current Trends in Theory and Practice of Computer Science*, SOFSEM '07, pages 367–376, Berlin, Heidelberg, 2007. Springer-Verlag.

[16] Ioannis Koutis. Faster algebraic algorithms for path and packing problems. In *Proceedings of the 35th international colloquium on Automata, Languages and Programming, Part I*, pages 575–586, Berlin, Heidelberg, 2008. Springer-Verlag.

[17] Ioannis Koutis and Ryan Williams. Limits and applications of group algebras for parameterized problems. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming: Part I*, ICALP '09, pages 653–664, Berlin, Heidelberg, 2009. Springer-Verlag.

[18] Eyal Kushilevitz and Noam Nissan. *Communication Complexity*. Cambdridge University Press, 1996.

[19] Yunlong Liu, Jianer Chen, and Jianxin Wang. A randomized approximation algorithm for parameterized 3-d matching counting problem. In Guohui Lin, editor, *Computing and Combinatorics*, volume 4598 of *Lecture Notes in Computer Science*, pages 349–359. Springer Berlin / Heidelberg, 2007.

[20] F.J. MacWilliams and N. J. A. Sloane. *The theory of error correcting codes*. North-Holland, 2nd edition, 1977.

[21] A. Terras. *Fourier Analysis on Finite Groups and Applications*. Cambridge University, 1999.

[22] Virginia Vassilevska and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, STOC '09, pages 455–464, New York, NY, USA, 2009. ACM.

[23] Ryan Williams. Finding paths of length k in $O^*(2^k)$ time. *Inf. Process. Lett.*, 109:315–318, February 2009.