Faster algebraic algorithms for path and packing problems

Ioannis Koutis
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213 USA
ioannis.koutis@cs.cmu.edu

April 30, 2008

Abstract

We study the problem of deciding whether an n-variate polynomial, presented as an arithmetic circuit G, contains a degree k square-free term with an odd coefficient. We show that if G can be evaluated over the integers modulo 2^{k+1} in time t and space s, the problem can be decided with constant probability in $O((kn+t)2^k)$ time and $O(kn+s)$ space. Based on this, we present new and faster algorithms for two well studied problems: (i) an $O^*(2^{mk})$ algorithm for the m-set k-packing problem and (ii) an $O^*(2^{3k/2})$ algorithm for the simple k-path problem, or an $O^*(2^k)$ algorithm if the graph has an induced k-subgraph with an odd number of Hamiltonian paths. Our algorithms use $poly(n)$ random bits, comparing to the $2^{O(k)}$ random bits required in prior algorithms, while having similar low space requirements.

1 Introduction

This paper presents new and faster randomized algorithms for the well studied parameterized simple path and set packing problems.

The k-path problem asks for a simple path of length k in a graph of n vertices V and m edges E, or a report that no such path exists. In the general case where k is not considered a parameter, the longest path and the Hamiltonian path problems are well known to be NP-hard. The earliest algorithm for the k-path in general directed graphs was given by Monien in [10]; the complexity of the algorithm is $O(k!nm)$. For undirected graphs Bodlaender gave a $O(2^k k!n)$ algorithm [2]. Papadimitriou and Yannakakis conjectured that the $O(\log n)$-path problem can be solved in polynomial time [11]. Alon, Yuster and Zwick confirmed the conjecture by describing an $O^*(5.44^k)$ randomized algorithm and an $O^*((2c)^k)$ deterministic algorithm for some constant $c > 4000$ [1] [1]. The algorithm of [1] is based on the observation that the Hamiltonian path problem can be solved by a fairly simple dynamic programming algorithm which essentially records all subsets $S \subseteq V$ for which there is a simple path that uses exactly the vertices of S; since there are 2^n such subsets, the algorithm runs in time and space $O^*(2^n)$. This deterministic dynamic programming algorithm can be used in the k-path problem when coupled with the *color-coding* technique of [1]: each vertex v is assigned one of k colors C, and the dynamic programming algorithm records all subsets $S \subseteq C$ for which there is a simple path that uses exactly the colors in C. In this way the running and space requirements become $O^*(2^k)$ and the algorithm finds a

[1] Following the parameterized complexity notation, we use $O^*(f(k))$ to hide a $poly(n)$ factor, but we will also use $O^*(f(k)poly(n))$ to hide factors of lower order.

k-path, if and only a k-path P existing in G is properly colored. As shown in [1] an expected number of $O(e^k)$ colorings must be tried before P gets properly colored; this gives the randomized algorithm. The derandomization is achieved by using a family F of $O(c^k)$ colorings, where each subset $S \subseteq V$ with $|S| = k$ is properly colored by at least one of the colorings in F. A big step towards closing the gap between the randomized and the deterministic complexity was taken by Kneis et. al. who presented an $O^*(16^k)$ deterministic algorithm [7]. The deterministic complexity was further reduced by Chen et. al. [3], who presented more efficient color-coding schemes and gave an $O^*(12.8^k)$ time deterministic algorithm. A new randomized divide-and-conquer algorithm -the first to deviate from the color-coding approach- was given in [7] and independently in [3]. Its time complexity is $O^*(4^k mk^{3.42})$ and its space complexity is $O(k \log kn + m)$, a remarkable improvement over the exponential (in k) space complexity of the color-coding approach.

The m-set k-packing problem gives a universe of n elements and a collection C of N sets each consisting of m elements; it asks for a sub-collection of k pairwise disjoint sets from C or a report that no such collection exists. In the general case where k is not considered a parameter, the set packing problems for all $m \geq 3$ are well known to be NP-hard. For the special case $m = 3$, the problem was first considered in [4] where a deterministic $O^*(2^{O(k)}(3k)!)$ algorithm was given. The bound was subsequently improved to $O^*(5.7k)^k$ in [6]. Later, it was realized that the m-set packing problem is amenable to a dynamic programming approach, which led to an $O^*(5.44^{mk})$ randomized algorithm in [8], and (when $m = 3$) to an $O^*(12.7D)^{3k}$ for $D \geq 10.4$ deterministic algorithm in [5]. In both cases the space complexity is $O^*(2^{3k})$. The deterministic time complexity was improved to $O^*(4.68^{3k})$ in [9]. The randomized divide-and-conquer approach of [3] improved the randomized time complexity to $O^*(2.52^{3k}n)$ and the space complexity to $O(nk \log k)$.

While the recent algorithmic progress in the path and packing problems has been impressive, both the color-coding and the randomized divide-and-conquer approaches seem to have an inherent time complexity limitation, namely the $O^*(2^k)$ bound for the k-path and the $O^*(2^{3k})$ bound for the 3-set k-packing. For example, in the k-path packing problem, the color-coding method requires the call of the $O(2^k)$ dynamic-programming algorithm for a number of different colorings. Breaking the $O(2.72^k)$ lower bound of colorings ([3]) would require some extensive combinatorial preprocessing of the graph, and even if this is possible a bound better than $O(c^k)$ seems unimaginable. In the randomized divide-and-conquer approach, at least $O(2^k)$ tries are required to "hit" the correct top level division. Breaking the additional $O^*(2^k)$ factor due to the recursion would require a very complicated re-usage of the computation in previous failed trials; even if this is possible, again it looks unimaginable that a complexity better than $O^*((2 + c)^k)$ can be achieved. On the other hand, the Hamiltonian path and the general m-set (n/m)-packing problem have fairly easy $O^*(2^n)$ algorithms. In view of this, a question presents itself.

Question: Is there a $O^*(2^k n)$ algorithm for the k-path and an $O^*(2^{mk})$ algorithm for the m-set k-packing?

1.1 Our approach and contributions

We answer the question in the affirmative for the m-set k-packing. We describe an algorithm that decides the problem in $O^*(2^{km}(km)^2 N)$ expected time, and based on that, an algorithm that finds a packing in $O^*(2^{km}(km)^2 N^2)$ expected time. Both algorithms require $O(kn)$ space. The answer is "almost" positive for the k-path problem as well. In the case the graph contains an odd number of k-paths (or generally when any subgraph induced by k vertices has an odd number of Hamiltonian paths), we describe an algorithm that finds a k-path in $O^*(2^k k^2 m(n + \min(k^2, m)))$ expected time and $O(kn + m)$ space. In the general case we describe an algorithm that decides the problem in

$O^*(2^{3k/2}k^2m)$ expected time and finds a k-path in $O^*(2^{3k/2}k^2m(n+\min(k^2,m)))$ expected time and $O(k^2m)$ space. As we will discuss in more detail later, the possibility that our algorithm can be derandomized with only a polynomial slowdown, doesn't look very remote.

Our approach is based on reductions to the problem of detecting a square-free term of degree k in polynomials represented as arithmetic circuits. The reduction for the Hamiltonian path problem was mentioned in [13]. The basic idea for set packing was presented in [8] and we shortly describe it here. Let x_i be a variable corresponding to the i^{th} element of the universe U of n elements, and Y_i be a monomial corresponding to the i^{th} of the N sets in the given collection C. We let Y_i be the product of the elements contained in the corresponding set. For example, if $S_i = \{1, 3, 5\}$, we let $Y_i = x_1x_3x_5$. Then it is not hard to see that the polynomial $(Y_1 + \ldots + Y_n)^k$ contains a square-free term of degree mk if and only if C contains a k-packing.

The problem of detecting square-free (or multilinear) terms in a polynomial presented as an arithmetic circuit was implicitly considered in [8]. While the dynamic programming/color-coding approach of [8] remains the only known solution for the general problem, in this paper we describe a faster and space efficient algorithm for detecting square-free terms with odd coefficients. In Section 2 we show that if the polynomial P can be evaluated over the integers modulo 2^{k+1} in time t and space s, the existence of the "odd" square-free term in P can be decided in time $O((kn+t)2^k)$ and space $O(kn+s)$ with constant probability. In Sections 3 and 4 we describe reductions of the m-set k-packing and the k-path respectively, to the odd square-free term detection problem. Finally in Section 5 we discuss some related open questions.

2 Detecting square-free terms with odd coefficients

Let $X = x_1, \ldots, x_n$ and let $K[X]$ be the commutative ring of polynomials with coefficients from the field K. Any non-zero polynomial $\mathbb{Z}_2[X]$ is by definition a sum (or equivalently a set) of monomials. A monomial is called square-free or multilinear if it is linear in all its variables. The total degree of a monomial is the sum of the degrees of its variables. In general, any polynomial $P \in \mathbb{Z}_2[X]$ can be represented as an arithmetic circuit which is a directed acyclic graph with addition and multiplication gates, and terminals corresponding to the variables.

Definition 2.1. *The ODD MULTILINEAR k-TERM problem: Given an arithmetic circuit G decide whether the polynomial $P(X) \in \mathbb{Z}_2[X]$ represented by G contains a multilinear term of total degree k or less.*

The main idea of our algorithm for this problem is the evaluation of the given polynomial over a suitably selected *commutative algebra*. In a commutative algebra the addition and multiplication operators behave in the same way as in the common algebra of integers or reals. This enables looking at the polynomial in two equivalent ways; its circuit representation allows its fast evaluation, whereas its expanded form as a sum of monomials allows us reasoning about its value in terms of the individual evaluations of its monomials. The slightly counterintuitive fact is that a commutative algebra may contain elements whose square is 0. We will exploit this to annihilate non-multilinear terms in the evaluation of P. It turns out that this can be done using commutative *group algebras* of $\mathbb{Z}_2^k$. We refer the reader to the Appendix for definitions and facts. We now give an algorithm for the ODD MULTILINEAR k-TERM problem, that works with the assumption that P contains only multilinear terms of degree exactly k. We will then see how this restriction can be easily removed.

decide-multilinear : Given an instance of the ODD MULTILINEAR k-TERM problem: **(i)** For each $x_i \in X$, independently pick a random vector $v_i \in \mathbb{Z}_2^k$ and assign to it the value $(v_0 + v_i) \in$

$\mathbb{Z}_2[\mathbb{Z}_2^k]$, where v_0 is the k-dimensional zero vector; Let $\bar{X}$ denote the assignment $x_i \leftarrow v_0 + v_i$. **(ii) -[option 1]:** If the coefficient of v_0 in $P(\bar{X})$ is equal to 1, then return "yes" otherwise return "no". **(ii) -[option 2]:** Let b_t denote the k dimensional vector containing the binary form of t, $\bar{\Lambda}_t$ denote the assignment $x_i \leftarrow 1 + (-1)^{v_i^T b_t}$ and $Z = \sum_{t=0}^{2^k-1} P(\bar{\Lambda}_t)$. If Z is equal to 2^k *mod* 2^{k+1} then return "yes", otherwise return "no".

It may appear that given the two options for step (ii), **decide-multilinear** gives two algorithms. However, in Theorem 2.5 we will prove that option 2 is equivalent to option 1 and thus it just provides an alternative implementation; from this we will derive our complexity claims. We will prove the soundness of the algorithm in Theorem 2.4, using option 1. In order to proceed with the proofs we need to introduce some notation. For simplicity let $\mathcal{A}$ denote $\mathbb{Z}_2[\mathbb{Z}_2^k]$. If $S \subset \mathbb{Z}_2^k$ is a set of vectors, we denote by $\pi(S)$ their product in $\mathcal{A}$. By convention, for all sets $V \subseteq \mathbb{Z}_2^k$ we will consider the empty set as a subset of S and we will let $\pi(\emptyset) = v_0$. Note that $\pi(A)\pi(B) = v_0$ if and only if $\pi(A) = \pi(B)$. We let J denote the element of A which is the sum of all vectors in $\mathbb{Z}_2^k$, that is $J = \sum_{v \in \mathbb{Z}_2^k} v$. We also say that an element w of $\mathcal{A}$ is *split* if it is the sum of exactly 2^{k-1} distinct vectors.

By construction, each monomial evaluates to an element of the form $\Pi(V) = \prod_{v \in V}(v_0 + v)$, where $V \subset \mathbb{Z}_2^k$. Using the fact that for all $v \in \mathbb{Z}_2^k$ we have $v_0 v = v$, we can expand $\Pi(V)$ into a sum, to get

$$\Pi(V) = \prod_{v \in V}(v_0 + v) = \sum_{S \subseteq V} \pi(S). \tag{1}$$

Lemma 2.2. *If the vectors in $V \subseteq \mathbb{Z}_2^k$ are linearly dependent over $\mathbb{Z}_2$, $\Pi(V)$ evaluates to 0. If the vectors in V are linearly independent, $\Pi(V)$ is a sum of $2^{|V|}$ distinct vectors (including v_0).*

Proof. By definition, when the vectors in V are linearly dependent, there is $V' \subseteq V$ such that $\pi(V') = v_0$. Then for all $S \subseteq V'$ we have $\pi(S)\pi(V' - S) = v_0$, which implies that $\pi(S) = \pi(V' - S)$. Hence, every term in the sum expansion (equality 1) of $\Pi(V')$ is generated an even number of times, which gives us $\Pi(V') = 0$. This in turn implies $\Pi(V) = 0$, because $\Pi(V) = \Pi(V)\Pi(V - V')$. This shows the first part of the Lemma. For the second part of the Lemma we observe that for all $S_a \neq S_b \subseteq V$ we have $\pi(S_a) \neq \pi(S_b)$. To see why, note that if $\pi(S_a) = \pi(S_b)$, then $\pi(S_a)\pi(S_b) = \pi(S_a S_b) = v_0$, which implies that the vectors in $(S_a \cup S_b - S_a \cap S_b)$ are linearly dependent, a contradiction. Therefore, since there are $2^{|V|}$ possible subsets of V (including $\emptyset$), $\Pi(V)$ is a sum of $2^{|V|}$ distinct vectors (including v_0). □

Lemma 2.3. *Let $P_{k-1} \in \mathbb{Z}_2[X]$ be a sum of multilinear monomials of degree exactly $k-1$.* **[a]** *For all assignments $\bar{X}$ of the form $x_i \leftarrow (v_0 + v_i)$, $P_{k-1}(\bar{X})$ is either split, or equal to 0, or equal to J.* **[b]** *In the case $P_{k-1}(\bar{X})$ is split, we have $Prob_{v \in \mathbb{Z}_2^k}((v_0+v)P_{k-1}(\bar{X}) = J) = Prob_{v \in \mathbb{Z}_2^k}((v_0+v)P_{k-1}(\bar{X}) = 0) = 1/2$.*

Proof. Let $P_{k-1} = \sum_j M_j$ where each M_j is a monomial of degree $k-1$. We will derive the Lemma by looking at $I = (v_0 + v)P_{k-1}(\bar{X})$ for a proper vector v. Note that $\mathbb{Z}_2^k$ contains 2^k vectors. Lemma 2.2 then implies that for all $v \in \mathbb{Z}_2^k$ and all M_j, we have $(v_0 + v)M_j(\bar{X}) = 0$ or $(v_0 + v)M_j(\bar{X}) = J$. Therefore, we have $I = 0$ or $I = J$, so the coefficient of any vector in I completely determines the value of I.

Clearly, this allows the possibilities $P_{k-1}(\bar{X}) = 0$ and $P_{k-1}(\bar{X}) = J$. Now assume that $P_{k-1}(\bar{X})$ is a sum of t distinct vectors, where $1 < t < 2^k$. We have

$$I = (v_0 + v)P_{k-1}(\bar{X}) = P_{k-1}(\bar{X}) + vP_{k-1}(\bar{X}).$$

Since $vv_1 = vv_2$ implies $v_1 = v_2$, it must be that $vP_{k-1}(\bar{X})$ is a sum of t distinct vectors in $\mathbb{Z}_2^k$. Hence, every vector in the expansion of I is generated $0, 1$ or 2 times. If some vector w is generated two times, its coefficient in I will be 0, and hence $I = 0$.

Now pick a vector v such that $vP_{k-1}(\bar{X})$ contains a vector w which is not in $P_{k-1}(\bar{X})$; this is clearly always possible. In that case the coefficient of w in I is 1, thus $I = J$. In addition I is a sum of at most $2t$ vectors, and since J is the sum of 2^k vectors, we must have $t \geq 2^{k-1}$. If $t > 2^{k-1}$, a simple pigeonhole argument shows that there must be a vector w' which is generated two times in I, implying that $I = 0$. This is a contradiction, so we must have $t = 2^{k-1}$. The **[b]** claim follows from the fact that $t = 2^{k-1}$ and the observation that I contains the vector v_0 with probability $1/2$, with respect to the choice of v. □

We are ready to prove the soundness of the algorithm.

Theorem 2.4. *If P does not contain a multilinear term, the algorithm* **decide-multilinear** *returns "no". Otherwise it returns "yes" with probability greater than 1/4.*

Proof. For the first claim, note that every monomial q which is not multilinear can be written as $x_i^2 q'$ for some variable x_i and monomial q'. Now observe that $\bar{x}_i^2 = (v_0 + v_i)^2 = 0$. Hence $\bar{x}_i^2 q' = 0$. So, if P does not contain multilinear terms, all its terms evaluate to 0. Thus, $P(\bar{X}) = 0$, and the algorithms returns "no". We will show the other direction using induction on the number of multilinear terms in P. Recall our assumption that all these terms have degree exactly k.

Base case: Let $V = \{v_1, \ldots, v_k\}$ be k random vectors drawn independently from $\mathbb{Z}_2^k$. The probability that a multilinear monomial of degree k evaluates to J is by construction equal to $Pr(\prod_{i=1}^{k}(v_0 + v_i) = J)$. By Lemma 2.2, this is equal to the probability that the vectors in V are linearly independent. By standard linear algebra facts, given that $\{v_1, \ldots, v_{j-1}\}$ are linearly independent, the vectors $\{v_1, \ldots, v_j\}$ are independent if and only if v_j is not in the vector space $\mathcal{S}$ generated by $\{v_1, \ldots, v_{j-1}\}$. In Lemma 2.2 we showed that there are exactly 2^{j-1} distinct linear combinations of the $j-1$ vectors, so there are $2^k - 2^{j-1}$ vectors that are not in $\mathcal{S}$. Hence,

$$Prob(\prod_{i=1}^{j}(v_0 + v_i) \neq 0) = (1 - \frac{2^{j-1}}{2^k}) Prob(\prod_{i=1}^{j-1}(v_0 + v_i) \neq 0) \geq \prod_{i=1}^{k}(1 - \frac{1}{2^i}).$$

The last product is lower bounded by $(1/2)\prod_{i=2}^{k}(1 - 1/i^2) = (k+1)/(4k) > 1/4$.

Inductive argument: If P is not a single monomial, then there is a variable x such that $P = xP_{k-1} + P'$ where x does not appear in $P' \neq 0$, and P_{k-1} is a sum of multilinear terms of degree $k-1$. Using Lemma 2.3**[a]** we can consider all possible cases under which P evaluates to J, to get

$$\begin{aligned}
&Prob(xP_{k-1} + P' = J) && = \\
&Prob(P' = J) Prob(P_{k-1} = 0 \text{ or } P_{k-1} = J/P' = J) + \\
&Prob(P' = J) Prob(P_{k-1} \text{ is split}/P' = J) Prob(xP_{k-1} = 0/P_{k-1} \text{ is split}) + \\
&Prob(P' = 0) Prob(P_{k-1} \text{ is split}/P' = 0) Prob(xP_{k-1} = J/P_{k-1} \text{ is split}) && \geq \\
&(1/2) Prob(P_{k-1} \text{ is split}) = Prob(xP_{k-1} = J).
\end{aligned}$$

The inequality came from dropping the first term and -after applying Lemma 2.3**[b]**- combining the remaining two. The probabilities are taken with respect to the random assignment $\bar{X}$. The polynomial xP_{k-1} contains less monomials than $xP_{k-1} + P'$, hence the inductive hypothesis applies. □

Let A be a commutative algebra and $\bar{Y}$ be an assignment $x_i \leftarrow \bar{y_i} \in A$, for $i = 1, \ldots, n$. We denote by $P_A(\bar{Y})$ the evaluation of P at $\bar{Y}$ over A.

Theorem 2.5. *Options 1 and 2 for step (ii) of* **decide-multilinear** *are equivalent. Furthermore, if the input circuit G can be evaluated over the integers modulo 2^{k+1} in time t and space s, option 2 can be performed in $O((nk+t)2^k)$ time and $O(nk+s)$ space.*

Proof. Let G be an arithmetic circuit with n variables X and $P \in \mathbb{Z}[X]$ be the polynomial represented by G. Also, let $\bar{X}$ be the assignment $x_i \leftarrow (v_0 + v_i)$, as defined in **decide-multilinear** . Let $\rho(u)$ denote the matrix representation of $u \in \mathbb{Z}[\mathbb{Z}_2^k]$ and $trace(M)$ denote the sum of the diagonal elements of the matrix M. Observe that

$$P_{\mathbb{Z}[\mathbb{Z}_2^k]}(\bar{X}) = \sum_{g \in \mathbb{Z}_2^k} a_g g \Rightarrow P_{\mathbb{Z}_2[\mathbb{Z}_2^k]}(\bar{X}) = \sum_{g \in \mathbb{Z}_2^k} (a_g \ mod\ 2) g. \tag{2}$$

Thus, it is enough to consider the parity of the coefficient of v_0 in $P_{\mathbb{Z}[\mathbb{Z}_2^k]}(\bar{X})$. Moving to $\mathbb{Z}[\mathbb{Z}_2^k]$ allows us working with the matrix representation of $P_{\mathbb{Z}[\mathbb{Z}_2^k]}(\bar{X})$, which is given by

$$\rho(P_{\mathbb{Z}[\mathbb{Z}_2^k]}(\bar{X})) = \sum_{g \in \mathbb{Z}_2^k} a_g \rho(g).$$

For each $g \in \mathbb{Z}_2^k$, $\rho(g)$ is a permutation matrix of dimension 2^k with zeros in the diagonal, with the exception of the identity v_0 of $\mathbb{Z}_2^k$, for which $\rho(v_0)$ is the identity matrix. Hence all the diagonal entries of $\rho(P_{\mathbb{Z}[\mathbb{Z}_2^k]}(\bar{X}))$ are equal. This, in combination with equality 2 implies that

$$\begin{aligned} P_{\mathbb{Z}_2[\mathbb{Z}_2^k]}(\bar{X}) = 0 \quad &\Rightarrow \quad trace(\rho(P_{\mathbb{Z}[\mathbb{Z}_2^k]}(\bar{X}))) = 0 \ mod\ 2^{k+1} \\ P_{\mathbb{Z}_2[\mathbb{Z}_2^k]}(\bar{X}) = J \quad &\Rightarrow \quad trace(\rho(P_{\mathbb{Z}[\mathbb{Z}_2^k]}(\bar{X}))) = 2^k \ mod\ 2^{k+1}. \end{aligned}$$

Now instead of evaluating P at $x_i \leftarrow (v_0 + v_i)$ over $\mathbb{Z}[\mathbb{Z}_2^k]$, we can equivalently evaluate it at $x_i \leftarrow \rho(v_0 + v_i)$ over the isomorphic matrix algebra $\mathcal{M} = \rho(\mathbb{Z}[\mathbb{Z}_2^k])$, and then compute (modulo 2^{k+1})

$$trace(P_{\mathcal{M}}(\rho(\bar{X}))) = trace(\rho(P_{\mathbb{Z}[\mathbb{Z}_2^k]}(\bar{X}))).$$

By the representation theory of $\mathbb{Z}_2^k$, there is a matrix U of dimension 2^k such that for all $v \in \mathbb{Z}_2^k$, $\rho(V) = U \Lambda_v U^{-1}$, where Λ_v is a diagonal matrix with the eigenvalues of $\rho(v)$. Let Λ_i denote the diagonal matrix containing the eigenvalues of $\rho(v_0 + v_i)$ and $\bar{\Lambda}$ denote the assignment $x_i \leftarrow \Lambda_i$. Let $\Lambda_{i,j}$ denote the j^{th} diagonal entry of Λ_i, and $\bar{\Lambda_j}$ denote the assignment $x_i \leftarrow \Lambda_{i,j}$. Using the well known relationship of the trace with the eigenvalues, we have (taking all quantities modulo 2^{k+1})

$$trace(P_{\mathcal{M}}(\rho(\bar{X}))) = trace(U P_{\mathcal{M}}(\bar{\Lambda}) U^{-1}) = trace(P_{\mathcal{M}}(\bar{\Lambda})) = \sum_{j=1}^{2^k} P_{\mathbb{Z}_{2^{k+1}}}(\bar{\Lambda}_j).$$

If b_j is the k-bit binary form of j, we can fix a U so that $\Lambda_{i,j} = 1 + (-1)^{v_i^T b_{j-1}}$ [12]. This completes the proof for the equivalence of options 1 and 2.

We have reduced the original problem to 2^k evaluations of P and the summation of the outputs over $\mathbb{Z}_{2^{k+1}}$. The 2^k evaluations can be performed sequentially, re-using the space, while the output sum is updated. The algorithm needs to maintain in the memory the assignment $\bar{X}$ which takes

space $O(kn)$. For each j, the algorithm computes the input $\bar{\Lambda}_j$ in $O(nk)$ time. The evaluation of $P(\bar{\Lambda}_j)$ can be done in time $O(t)$, and space $O(nk+s)$, by assumption. Hence the total time is $O((nk+t)2^k)$ and the space requirement is $O(nk+s)$. $\square$

Remark. If the smallest multilinear term in P has degree $k-j$, we can consider $P_j = (y_1 \dots y_j)P$. By Lemma 2.2, any term of degree greater than k always evaluates to 0, so running **decide-multilinear** on P' has the same effect as running it with the assumed restriction. We omit the details to the full paper.

3 Reducing m-set k-packing to multilinear mk-term

Let us start with a formal definition of the set packing problem.

Definition 3.1. *The m-SET k-PACKING problem (decision and search): Given a collection C of N sets, each containing m elements from a universe U of n elements, decide whether there is a collection $C' \subseteq C$ of k mutually-disjoint sets. If yes, find such a collection C'.*

The main result of this Section is the construction of a family $P_{\mathcal{A}}(X)$ of $2^{|\mathcal{A}|}$ packing-encoding polynomials, parameterized by a set $\mathcal{A}$ of variables taking binary values.

path-encoding polynomials: Given an instance I of the m-SET k- PACKING problem: (i) assign variables $X = \{x_i\}$, $i = 1, \dots, n$ to the elements of U, (ii) assign to the set $S_i \in C$ the degree m *set-monomial* Y_i, defined as the product of the variables corresponding to the elements of S_i. (iii) Let $\mathcal{A} = \{a_{i,j} : i \in [1,k], j \in [1,N]\}$ and define $P_{\mathcal{A}}(X) = \prod_{i=1}^{k} \left(\sum_{j=1}^{N} a_{i,j} Y_j \right)$.

Theorem 3.2. *Let $P \in \mathbb{Z}_2[X]$ be a polynomial picked uniformly at random from $P_{\mathcal{A}}(X)$ by letting $Prob(a_{i,j} = 0) = Prob(a_{i,j} = 1) = 1/2$, independently for all i, j. If I is a "yes" instance of the m-set k-packing problem, the polynomial $P \in \mathbb{Z}_2[X]$ has a multilinear term of degree mk with probability at least $1/4$. Otherwise, P has no multilinear terms.*

Proof. It is clear that P can be expanded to a sum of what we will call *set-products*, each of which is a product of k set-variables Y_i. Each set-product is itself a monomial and has total degree mk. If a set-product is a multiple of $Y_i Y_j$ for two intersecting sets S_i and S_j, then by construction it is not multilinear. It follows that if I does not contain a k-set packing, P has no multilinear terms.

Claim A: The coefficient $cf(Y_1 \dots Y_k)$ (and thus of any product of k distinct set variables) in P is odd with probability at least $1/4$, with respect to the random assignment to the coefficients $a_{i,j}$.

To prove Claim A consider the $k \times k$ matrix $M_{i,j} = a_{i,j}$. Let S_k denote the set of all permutations of $1, \dots, k$, and $perm(M), det(M)$ denote the permanent and the determinant of M. We have

$$cf(Y_1 \dots Y_k) = \sum_{\pi \in S_k} \prod_{i=1}^{k} a_{i,\pi(i)} = perm(M) = det(M) \ (mod\ 2).$$

By standard linear algebra facts $det(M)\ mod\ 2 = 0$ if and only if the columns of M are linearly dependent over $\mathbb{Z}_2$. However, the columns of M are random vectors picked independently and uniformly from $\mathbb{Z}_2^k$. In the proof of Theorem 2.4 (base case of induction), we showed that the probability that they are linearly independent is at least $1/4$. This finishes the proof of Claim A $\bullet$

Assume now that I has at least one k-set packing, and fix one. Clearly its set-product is a multilinear term T of degree mk. The same multilinear term T can be generated by a collection C_T of distinct set-products, corresponding to different k-set packings that cover the same subset of U. It is then enough to show that the probability (with respect to the random assignment to the coefficients $a_{i,j}$) that C_T generates an odd number of copies of T is at least $1/4$. Let us denote this probability by $Prob(C_T \rightsquigarrow 1)$.

Claim B: For all $C' \subseteq C_T$, there is a $C'' \subset C'$ such that

$$Prob(C'' \rightsquigarrow 1) \leq Prob(C' \rightsquigarrow 1).$$

Let $\mathcal{Y} = \{Y_1, \ldots, Y_N\}$ be the set of set-variables. Let C_Z be a collection of set-products that generate the same multilinear term T and share the common factor $Z \in \mathcal{Y}$. Let $\mathcal{A}_S$ denote the set of assignments to the coefficients $a_{i,j}$ corresponding to the variables of $S \subseteq \mathcal{Y}$ in the factors of P. Now fix an assignment $A \in \mathcal{A}_{\mathcal{Y}-Z}$ and let $a_{i,Z}$ denote the coefficients that multiply Z in the factors of P. Considering P as a function of $a_{i,j}$, its partial evaluation at A is always of the form

$$P(A) = R + T\left(\sum_{i=1}^{k} a_{i,Z} b_i\right)$$

where R is a sum of terms different than T, and $b_i = 0$ or $b_i = 1$ depending only on A. It can be seen that there are two cases: (a) for all i, we have $b_i = 0$ in which case for all assignments in $\mathcal{A}_Z$ the polynomial $P(A)$ does not contain T, (b) there is at least one j for which $b_j = 1$. In this case we say that C_Z is *Z-dependent* under the assignment A. Then, it is not hard to see that

$$Prob_{\mathcal{A}_Z}(C_Z \rightsquigarrow 1 / C_Z \textit{ is } Z\textit{-dependent}) = 1/2. \tag{3}$$

where as indicated by the notation the probability is taken with respect to the assignments in $\mathcal{A}_Z$. Using the same notation, it follows that

$$Prob_{\mathcal{A}}(C_Z \rightsquigarrow 1) = Prob_{\mathcal{A}}(C_Z \rightsquigarrow 0) = (1/2) Prob_{\mathcal{A}_{Y-Z}}(C_Z \textit{ is } Z\textit{-dependent}) \tag{4}$$

We are now ready to move to the main part of the proof of Claim B. In the following, the probability subscripts are implied by the context, and we will drop them for simplicity. Let C' be an arbitrary subset of C_T. Clearly, unless $|C'| = 1$, there is $Z \in \mathcal{Y}$ such that $C' = C_Z \cup C_{\bar{Z}}$ where C_Z contains all set-products in C' that are multiple of a common factor of Z and $C_{\bar{Z}} = C' - C_Z \neq \emptyset$. Considering all possibilities, and appropriately using equalities 3 and 4, we have

$$\begin{aligned} Prob(C' \rightsquigarrow 1) &= Prob(C_{\bar{Z}} \rightsquigarrow 1) Prob(C_Z \textit{ is not } Z\textit{-dependent} / C_{\bar{Z}} \rightsquigarrow 1) + \\ &\quad Prob(C_{\bar{Z}} \rightsquigarrow 1) Prob(C_Z \textit{ is } Z\textit{-dependent} / C_{\bar{Z}} \rightsquigarrow 1)(1/2) + \\ &\quad Prob(C_{\bar{Z}} \rightsquigarrow 0) Prob(C_Z \textit{ is } Z\textit{-dependent} / C_{\bar{Z}} \rightsquigarrow 0)(1/2) \\ &\geq (1/2) Prob(C_Z \textit{ is } Z\textit{-dependent}) = Prob(C_Z \rightsquigarrow 1). \end{aligned}$$

The inequality came from dropping the first term and combining the remaining two. The set C_Z is the set C'' stated in Claim B. This completes the proof for Claim B •

Finally we observe that Claim B can be used repeatedly to show that there is a chain $C_T \supset C_1 \ldots \supset C_\nu$, where $|C\nu| = 1$, such that $Prob(C_T \rightsquigarrow 1) \geq Prob(C_1 \rightsquigarrow 1) \geq \ldots \geq Prob(C_\nu \rightsquigarrow 1)$. Claim A gives that $Prob(C_\nu \rightsquigarrow 1) \geq 1/4$ and the proof is completed. □

Theorem 3.3. *There is an $O^*(2^{km}(km)^2N)$ time algorithm such that, on a instance I of the m-set k-packing problem, its output is "yes" with probability at least $1/16$ only if I contains a k-packing. In a "yes" instance, a k-packing can be found with constant probability by $O(N \log N)$ calls to the decision algorithm. Both the decision and the search algorithms use $O(kn)$ space.*

Proof. A random polynomial P from the family of packing-encoding polynomials $P_{\mathcal{A}}(X)$ can be constructed in $O(kN)$ time. The polynomial P can be evaluated over $\mathbb{Z}_2^{km+1}$ with $O(kmN)$ addition and multiplication operations involving $(km+1)$-bits numbers, each of which takes $O^*(km)$ time. It is clear that $O(kn)$ space is required to store an assignment over $\mathbb{Z}_2^k$ to the variables, and it is not hard to see that P can be evaluated in $O(kn)$ space over $\mathbb{Z}_2^k$ because apart from the assignment to the variables, only three values must be kept around at any given time. The proof follows by applying Theorem 2.5. The details of the search algorithm can be found in [8]. The key observation is that if $C' \subseteq C$ contains a k-packing, then $O(\log N)$ calls of the decision algorithm on $C' - Z$ decide with probability $1 - 1/N$ whether $Z \in C'$ is contained in all k-packings of C' or $C' - Z$ contains a k-packing. Starting from C, the algorithm applies this procedure at most N times to find the k-packing with probability $(1 - 1/N)^N > 1/4$. □

4 Reducing k-simple path to multilinear $3k/2$-term

We start with a definition of the problem. We will work with directed graphs; if the graph is undirected we can see it as a directed graph in the obvious way.

Definition 4.1. The k-PATH problem (decision and search): *Given a graph G with n vertices V and m directed edges E, decide whether the graph contains a path of length $k-1$ which connects k distinct vertices. If yes, find such a path.*

Let $\mathcal{P}_{t,v}$ denote the set of directed paths of length t ending in $v \in V$. Let $X = \{x_{v1}, \ldots, x_{vn}\}$ be a set of n variables corresponding to the vertices of G, and $Y = \cup_{t=1}^{k} \cup_{(v_i,v_j)\in E} y_{v_i,v_j,t}$ be a set of nk variables where each (directed) edge corresponds to k variables; the k different copies are intended to encode the position of the edge along a path of length k. If $p = (v_1, v_2, \ldots, v_t)$ is a path of length $t-1$ we define its encoding to be

$$enc(p) = \left(\prod_{i=1}^{t} x_{v_i}\right)\left(\prod_{j=1}^{\lfloor t/2 \rfloor} y_{v_{2j-1},v_{2j},2j-1}\right)$$

and the t-path encoding polynomial for $v \in V$ as $P(t,v) = \sum_{p\in\mathcal{P}_{t,v}} enc(p)$.

Lemma 4.2. *For a path p, the encoding $enc(p)$ is multilinear if and only if p is simple. If p_1, p_2 are two distinct simple paths then $enc(p_1) \neq enc(p_2)$. Hence, the path encoding polynomial $P_k = \sum_{v\in V} P(k,v)$ contains a multilinear term of degree $k + \lfloor k/2 \rfloor$ if and only if G contains a simple path k.*

Proof. If p is not a simple path then clearly the first factor in $enc(p)$ contains a squared variable. If p is simple, then all the vertices and edges it uses are distinct so $enc(p)$ is multilinear. Now notice that a directed path p is completely determined by: (i) its sink vertex, (ii) the list of the directed edges whose source is at even distance from the first vertex on the path, along with their positions

in p. This list is completely determined by the second factor in $p(e)$, while if the sink vertex is missing in the list (when p has an even number of edges) it can be recovered from the first factor in $p(e)$. Hence every simple k-path ending in v is mapped to a distinct (thus with a unit coefficient) multilinear monomial in P_k, and the monomial has degree $k + \lfloor k/2 \rfloor$. □

We now describe a circuit for the computation of P_k. By convention we define $P(0, v) = x_v$. Assume we have constructed a circuit for $P(t-1, v)$ for all $v \in V$. It is not hard to see that the circuit for $P(t, v)$ can be constructed as follows:

$$P(t, v) = \sum_{(u,v)\in E} x_v (y_{u,v,t})^{(t \bmod 2)} P(t-1, u).$$

Theorem 4.3. *There is an $O^*(2^{3k/2}k^2m)$ such that, on a graph G, it returns "yes" with probability at least $1/4$ only if the graph contains a simple k-path. In a "yes" instance, a simple k-path can be found with $O^*(n + \min(k^2, m))$ applications of the decision algorithm. Both the decision and the search algorithm use $O(k^2m)$ space.*

Proof. The algorithm runs **decide-multilinear** on $P_k = \sum_{v\in V} P(k, v)$. Given the values for $P(t-1, v)$, the values $P(t, v)$ can be computed with $O(m)$ additions and multiplications in $\mathbb{Z}_2^{k+1}$. Hence the polynomial $P(k, v)$ can be evaluated over $\mathbb{Z}_2^{k+1}$ with $O(km)$ operations taking $O^*(k)$ time each. For all t the circuit needs to remember $((k/2)m + n)$ variable values and the n values of $P(t-1, v)$, taking space $O(k^2m)$. The claim follows by applying Theorem 2.5. The algorithm and proof for the search version of the problem is similar to those for the search version of the m-set k-packing. Roughly, the algorithm will keep removing vertices until it is left with an induced k-subgraph which still contains a k-path, and then it will remove up to $O(k^2)$ edges until it is left with a k-path. □

Remark 1. Assume that there is a subset $V' \subseteq V$ with $|V'| = k$ such that the number $n_{V'}$ of (directed) Hamiltonian paths in the graph induced by V' is odd. If we set the edge variables to $Y = 1$ in the definitions of the path encodings, it is not hard to see that the coefficient of $\prod_{v\in V'} x_v$ in $P_k(Y = 1)$ is equal to $n_{V'}$.This observation can be used to reduce the time complexity to $O^*(2^k k^2 m)$ and the space to $O(kn + m)$.

5 Open questions

It is a possibility that the algorithms in this paper can be derandomized. A basic step seems to be the construction of a family of assignments $\mathcal{F} : X \to \mathbb{Z}_2^k$, such that for each k-subset X' of X, there is an assignment in $\mathcal{F}$ under which the vectors in X' are linearly independent over $\mathbb{Z}_2$. Since a random assignment satisfies this with probability at least $1/4$, it is a plausible conjecture that there is a polynomial size $\mathcal{F}$ with the required property.

6 Appendix. Group algebras of $\mathbb{Z}_2^k$ and their representation

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 $K[\mathbb{Z}_2^k]$, where K is a field, 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 $K[\mathbb{Z}_2^k]$ 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 $\alpha \in K$ 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 $K[\mathbb{Z}_2^k]$ is defined by

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

It can be verified that $K[\mathbb{Z}_2^k]$ is commutative. In the particular case of $\mathbb{Z}_2[\mathbb{Z}_2^k]$, an element can also be seen as a subset of $\mathbb{Z}_2^k$.

Matrices of dimension d with integer 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 ([12], 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

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

The map ρ can be extended to a one-to-one map of $\mathbb{Z}[\mathbb{Z}_2^k]$ 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 $\mathbb{Z}[\mathbb{Z}_2^k]$ and $\alpha \in \mathbb{Z}$, 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 $\rho(\mathbb{Z}[\mathbb{Z}_2^k])$.

The matrices $\rho(\mathbb{Z}_2^k)$ are simultaneously diagonalizable, i.e. there is a 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 $(-1)^{v^T b(i)}$ [12].

7 Acknowledgments

This work was partially supported by the National Science Foundation under grant number CCF-0635257.

References

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

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

[3] Jianer Chen, Songjian Lu, Sing-Hoi Sze, and Fenghui Zhang. Improved algorithms for path, matching, and packing problems. In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 298–307, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

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

[5] Michael R. Fellows, Christian Knauer, Naomi Nishimura, Prabhakar Ragde, Frances A. Rosamond, Ulrike Stege, Dimitrios M. Thilikos, and Sue Whitesides. Faster fixed-parameter tractable algorithms for matching and packing problems. In *Algorithms - ESA 2004, 12th Annual European Symposium, Bergen, Norway, September 14-17, 2004, Proceedings*, pages 311–322, 2004.

[6] Weijia Jia, Chuanlin Zhang, and Jianer Chen. An efficient parameterized algorithm for m-set packing. *J. Algorithms*, 50(1):106–117, 2004.

[7] Joachim Kneis, Daniel Mölle, Stefan Richter, and Peter Rossmanith. Divide-and-color. In *WG: Graph-Theoretic Concepts in Computer Science, 32nd International Workshop*, pages 58–67, 2006.

[8] Ioannis Koutis. A faster parameterized algorithm for set packing. *Information Processing Letters*, 94(1):4–7, 2005.

[9] Yang Liu, Songjian Lu, Jianer Chen, and Sing-Hoi Sze. Greedy localization and color-coding: Improved matching and packing algorithms. In *Parameterized and Exact Computation, Second International Workshop, IWPEC 2006, Zürich, Switzerland, September 13-15, 2006, Proceedings*, pages 84–95, 2006.

[10] Burkhard Monien. How to find long paths efficiently. *Annals of Discrete Mathematic*, 25:239–254, 1955.

[11] Christos H. Papadimitriou and Mihalis Yannakakis. On limited nondeterminism and the complexity of the V-C dimension. *J. Comput. Syst. Sci.*, 53(2):161–170, 1996.

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

[13] Leslie G. Valiant. Why is boolean complexity difficult? *Boolean Function Complexity, Lond. Math. Soc. Lecure Note Ser.*, 169.