ssdnm
środowiskowe
studia doktoranckie
z nauk matematycznych

Sylwia Antoniuk

Uniwersytet A. Mickiewicza w Poznaniu

Simple algebraic methods vs. NP–hard problems

Praca semestralna nr 3

(semestr letni 2011/12)

Opiekun pracy: Łukasz Kowalik

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Simple algebraic methods vs. NP–hard problems

Sylwia Antoniuk

Abstract

In this survey, we present how simple algebraic methods like the inclusion-exclusion principle can be used in solving NP–hard problems. In particular, we present a method due to Björklund and Husfeldt [4] which finds the chromatic number of a given graph in $O^*(2^n)$ time. We also show a randomized algorithm for determining undirected Hamiltonicity in $O^*(1.657^n)$ time presented by Björklund in [3] and a randomized algorithm which solves the k–path problem in $O^*(1.657^k)$ time which was introduced in [5] by Björklund, Husfeldt, Kaski and Koivisto. Both algorithms are based on algebraic techniques.

1 Preliminaries

As solving NP–hard problems often requires exponential time algorithms, unless P=NP, the real challenge is to look for such algorithms which would significantly reduce the best currently known constants in the exponent. There are number of problems which could serve as an example of this peculiar battle for constant. Some of the techniques and algorithms are complex and advanced, but the real beauty lies in finding simple solutions to well-studied problems and therefore remarkably improving old results.

In this survey we collect a few examples of employing simple algebraic techniques such as the inclusion-exclusion principle in order to improve constants in the exponent of the time complexity of algorithms which solve some well-known graph problems. In particular, we show a method for counting the chromatic number of a given graph with n vertices in $O^*(2^n)$ time. The first solution to this problem comes from Lawler [8] who in 1976 presented a solution running in $O(2.4423^n)$ time. This constant has been gradually reduced until Björklund and Husfeldt [4] presented a solution running in $O^*(2^n)$ time.

Next, we deal with the problem of undirected Hamiltonicity. It is a special case of the Traveling Salesman Problem (TSP) which was first formulated in 1930. The $O^*(2^n)$ time solution for general TSP has been described in the early 1960' by Bellman [2] and independently by Held and Karp [7]. However, for many years the problem of finding a $O^*(c^n)$ solution with some constant $c < 2$ remained open. There have been partial results for restricted classes of graphs such as claw–free graphs, cubic graphs or graphs of maximum degree four but only in 2010 Björklund presented a first solution for general graphs. In [3] he described a Monte Carlo algorithm running in $O^*(1.657^n)$ time.

Finally, we consider a k–path problem where we are interested in a solution parametrized by k rather than n. A first deterministic $O^*(c^k)$, $c > 8000$ algorithm for a k–path problem is due to Alon, Yuster and Zwick [1]. In the same paper

they present a randomized algorithm for a k–path problem which runs in $O^*((2e)^k)$ time. This constant has been reduced ever since and now the best known result for a randomized algorithm is $c = 1.657$ which has been obtained by Björklund, Husfeldt, Kaski and Koivisto in [5]. However, their algorithm works only for undirected graphs. The best result for directed graphs is due to Williams [10] who showed a randomized algorithm running in $O^*(2^k)$ time.

1.1 Notation

A *graph* $G = (V, E)$ consists of a set V, called the set of *vertices*, and a set of pairs of vertices E, called *edges*. A *directed* graph is a graph $D = (V, A)$ where every edge is an ordered pair, we call them *arcs*. An *independent set* in G is any subset $V' \subseteq V$ of vertices which does not contain an edge of G. A *walk* P is any sequence of vertices $v_1, v_2, \ldots, v_m$, such that $(v_i, v_{i+1}) \in E$, $i = 1, \ldots, m-1$. A *(simple) path* is a walk where every vertex appears exactly once. A *closed* walk is a walk for which $v_1 = v_m$. A *cycle* C is a closed walk such that every vertex, apart from the first and the last one, appears in C exactly once. Moreover, we call $G' = (V', E')$ a *subgraph* of G if $V' \subseteq V$ and $E' \subseteq E'$. A subgraph G' is called *induced* if E' contains all the edges of G with both ends in V'. We denote induced subgraphs by $G[V']$.

Furthermore, we let $O^*(f(k))$ denote $f(k) \cdot n^{O(1)}$, where n is the input size such as e.g. the number of vertices. Therefore, O^* hides the factors which are polynomial in the parameter size.

1.2 The Inclusion-exclusion principle

Throughout the paper we employ the following well-known counting technique called *the inclusion-exclusion principle*.

Theorem 1.1 (The inclusion-exclusion principle for unions)**.** *Let U be a finite set and $A_1, \ldots, A_n \subseteq U$ be a family of arbitrary subsets of U. Then the number of elements contained in the union of the subsets A_i can be counted as follows*

$$|\bigcup_{i=1,\ldots,n} A_i| = \sum_{X \subseteq [n], X \neq \emptyset} (-1)^{|X|-1} |\bigcap_{i \in X} A_i|.$$

In some cases we are more interested in counting the number of elements contained in the intersection of the subsets $\overline{A_i}$ rather than in their union. The following theorem shows how this can be done.

Theorem 1.2 (The inclusion-exclusion principle for intersections)**.** *Let U be a finite set and $A_1, \ldots, A_n \subseteq U$ be a family of arbitrary subsets of U. Moreover, set $\overline{A_i} = U - A_i$. Then the number of elements contained in the intersection of the subsets A_i can be counted as follows:*

$$|\bigcap_{i=1,\ldots,n} \overline{A_i}| = |U| + \sum_{X \subseteq [n], X \neq \emptyset} (-1)^{|X|} |\bigcap_{i \in X} A_i|. \tag{1}$$

We consider also the weighted version of the above theorem.

Theorem 1.3 (The weighted version of inclusion-exclusion principle for intersections)**.** *Let U be a finite set and $A_1, \ldots, A_n \subseteq U$ be a family of arbitrary subsets of U. Furthermore, let $w : U \to \mathbb{R}$ be a real-valued weight function extended to the domain 2^U by setting $w(A) = \sum_{a \in A} w(a)$. Then the following holds*

$$w(\bigcap_{i=1,\ldots,n} \overline{A_i}) = w(U) + \sum_{X \subseteq [n], X \neq \emptyset} (-1)^{|X|} w(\bigcap_{i \in X} A_i). \tag{2}$$

1.3 Schwartz-Zippel lemma

Another tool is the following lemma which we use in the analysis of the polynomial identity testing in Monte Carlo algorithms.

Lemma 1.4 (Schwartz-Zippel [9])**.** *Let $P(x_1, \ldots, x_n)$ be a non–zero n–variate polynomial of total degree d over a field F. Then for $r_1, \ldots, r_n \in F$ chosen uniformly at random*

$$Pr(P(r_1, \ldots, r_n) = 0) \leq \frac{d}{|F|}.$$

2 Graph coloring in $O^*(2^n)$ time

For a given graph $G(V, E)$, a *graph coloring* is a function $c : V \to \mathbb{N}$ such that $c(v) \neq c(w)$ for any pair v, w of adjacent vertices. A coloring $c : V \to \{1, \ldots, k\}$ is called a *k–coloring* and the smallest k for which graph G admits a k–coloring is called the *chromatic number* of graph G and is denoted by $\chi(G)$. Finding the chromatic number of a given graph is NP–hard. Björklund and Husfeldt show in [4] how to compute the chromatic number of a given graph in $O^*(2^n)$ time and exponential space. They also present a polynomial space algorithm for computing the chromatic number in $O(2.2461^n)$ time. Their algorithm is based on the inclusion-exclusion principle and simple dynamic programming. Below we present main ideas of the algorithm.

A k–coloring of a graph G can be seen as a partition of the set of vertices into k independent sets. It is easy to observe that to show the existence of a k–coloring it is enough to show that there is a covering of the set of vertices V by a family of k independent sets. Therefore we use the inclusion-exclusion principle to count the number of such coverings.

Let U be the family of k–tuples $(I_1, \ldots, I_k)$, such that each $I_i \subseteq V$ is an independent set. Next, let $A_v = \{(I_1, \ldots, I_k) \in U : v \in \bigcup_{i=1}^{k} I_i\}$. Then G has a k–coloring if and only if

$$|\bigcap_{v \in V} A_v| \neq 0.$$

Using the inclusion-exclusion principle we can reduce this problem to computing the values of

$$|\bigcap_{v \in X} \overline{A_v}| = |\{(I_1, \ldots, I_k) \in U : I_1, \ldots, I_k \subseteq V - X\}| = s(V - X)^k,$$

for each $X \subseteq V$, where $s(Y)$ is the number of independent sets contained in Y. To compute $s(Y)$ for each $Y \subseteq V$ we use dynamic programming. We set $s(\emptyset) = 1$ and

for Y such that $|Y| \geq 1$ we employ the following formula

$$s(Y) = s(Y - \{y\}) + s(Y - \{y\} - N_G(y)),$$

where $y \in Y$ is an arbitrary element of Y and $N_G(y)$ denotes the neighborhood of y, that is the set off all vertices adjacent to y in G. Next, we use (1) to compute $|\bigcap_{v \in V} A_v|$. This gives us an algorithm which uses $O^*(2^n)$ time and space.

Theorem 2.1. *Let $G = (V, E)$ be a graph with n vertices. Then*

1) *finding a k–coloring of G or deciding that such a coloring does not exist can be done in $O^*(2^n)$ time and space;*

2) *the chromatic number of G can be computed in $O^*(2^n)$ time and space.*

To get polynomial space we can use the Gray-code enumeration of the subsets of X to compute $s(X)$ for every X with $|X| = i$ in $O(i \cdot 2^i)$ time, which gives us a total running time $O^*(3^n)$. However, this can be improved. Using the result by Fürer and Kasiviswanathan [6] we can compute $s(Y)$ in $O(1.2461^n)$ time and polynomial space. Therefore we obtain an algorithm which runs in $O(\sum_{k=0}^{n} \binom{n}{k} 1.2461^n) = O(2.2461^n)$ and uses only polynomial space.

Theorem 2.2. *Let $G = (V, E)$ be a graph with n vertices. Then determining whether G admits a k-coloring or not can be done in $O(2.2461^n)$ time and polynomial space.*

3 Determining undirected Hamiltonicity in $O^*(1.657^n)$ time

Another NP–hard problem is to decide whether a given graph $G = (V, E)$ has a *Hamiltonian cycle*, that is a cycle which contains all the vertices of graph G. It is a special case of the Traveling Salesman Problem (TSP) which asks for finding a tour of minimal total weight visiting every vertex in an edge weighted graph exactly once. The general TSP, and thus the Hamiltonicity problem, can be solved in $O(n^2 2^n)$ time using dynamic programming. In [3] Björklund presented an elegant Monte Carlo algorithm for Hamiltonicity detection in $O^*(1.657^n)$ time which is based on the use of counting methods. We present main ideas of his approach.

A *cycle cover* of a graph is a set of cycles such that each vertex belongs to exactly one cycle. A Hamiltonian cycle is a special case of a cycle cover, namely a cycle cover which consists of only one cycle. For a directed graph $D = (V, A)$, let $cc(D)$ denote the family of all cycle covers of D and $hc(D)$ the family of all Hamiltonian cycles of D. In the following we consider an extension of graph G to a biderected graph D and associate a set of labels with every edge in D in such a way that deciding whether G is Hamiltonian reduces to investigating a certain polynomial for D called the *labeled cycle cover sum*.

Definition 3.1. The *labeled cycle cover sum* for a directed graph $D = (V, A)$, a set of labels L and a function $f : A \times 2^L \setminus \{\emptyset\} \to R$ on some codomain ring R is

$$\Lambda(D, L, f) = \sum_{C \in cc(D)} \sum_{g: L \twoheadrightarrow C} \prod_{a \in C} f(a, g^{-1}(a)), \tag{3}$$

where the inner sum is taken over all surjective functions $g : L \twoheadrightarrow C$.

The main trick is to choose a proper function f, which has a ring of characteristic two as a codomain, so that all the summands in (3) which come from non-Hamiltonian cycle covers would cancel out. We must also make sure that the summands which come from Hamiltonian cycles would not vanish. Let s be any vertex of V. We say that f is an *s–oriented mirror function* if $f(uv, Z) = f(vu, Z)$ for all Z and all $u \neq s$, $v \neq s$.

Lemma 3.2. *Let $D = (V, A)$ be a bidirected graph, L a set of labels. Moreover, for an arbitrarily chosen $s \in V$ let f be an s–oriented mirror function with a codomain of characteristic two. Then*

$$\Lambda(D, L, f) = \sum_{H \in hc(D)} \sum_{g: L \twoheadrightarrow H} \prod_{a \in H} f(a, g^{-1}(a)).$$

Proof. Since every non-Hamiltonian cycle cover consists of at least two cycles, there is one, call it C, which does not contain s. By redirecting C we obtain a different cycle cover, but the contribution of both cycle covers to (3) is the same and cancels out because we are in a ring of characteristic two. □

Later we define f and L in such a way that every Hamiltonian cycle contributes at least one unique monomial to the associated labeled cycle cover sum seen as a polynomial in the underlying variables. Therefore deciding whether G contains a Hamiltonian cycle reduces to testing if the labeled cycle cover sum is a non-zero polynomial.

To solve labeled cycle cover sum quickly we use the fact that in rings of characteristic two matrix determinants coincide with the permanents and the latter can be interpreted as the sums of weighted cycle covers in directed graphs, i.e. for a directed graph $D = (V, A)$ and a weight function $w : A \to R$ we define a $|V| \times |V|$ matrix $\mathbf{A}$

$$\mathbf{A}_{i,j} = \begin{cases} w(ij) & , \quad ij \in A, \\ 0 & , \quad \text{otherwise.} \end{cases}$$

Then

$$\det(\mathbf{A}) = \operatorname{per}(\mathbf{A}) = \sum_{C \in cc(D)} \prod_{a \in C} w(a).$$

Next, we introduce a polynomial $p(f, r)$ in an indeterminate r, which computes an associated labeled cycle cover sum in characteristic two and can be thought of as an inclusion-exclusion formula

$$p(f, r) = \sum_{Y \subseteq L} \det \left(\sum_{Z \subseteq Y} r^{|Z|} \mathbf{M}_f(Z) \right),$$

where for every $Z \subseteq L$, $\mathbf{M}_f(Z)$ is a matrix given by:

$$\mathbf{M}_f(Z)_{i,j} = \begin{cases} f(ij, Z) & , \quad ij \in A, Z \neq \emptyset, \\ 0 & , \quad \text{otherwise.} \end{cases}$$

The indeterminate r controls the total rank of the subsets used as labels in the labeled cycle covers and it turns out that the value of the labeled cycle cover sum coincides with the coefficient of the monomial $r^{|L|}$ in $p(f,r)$.

Lemma 3.3. *For a directed graph D, a set of labels L and any function $f : A \times 2^L \setminus \{\emptyset\} \to GF(2^k)$*

$$[r^{|L|}]p(f,r) = \Lambda(D,L,f).$$

Above relation enables us to compute the labeled cycle cover sum in time exponential in the number of labels and polynomial in the size of the input graph, that is in $O^*(2^{|L|})$. Therefore our goal is to use only small number of labels.

Lemma 3.4. *The labeled cycle cover sum $\Lambda(D,L,f)$, where f is a function with codomain $GF(2^k)$, D is a directed graph on n vertices, such that $2^k > |L|n$, can be computed using $O((|L|^2n+|L|n^{1+\omega})2^{|L|}+|L|^2n^2)$ arithmetic operations over $GF(2^k)$, where ω is the square matrix multiplication exponent.*

The general idea is to split vertices of G into two sets, use one of the sets as labels and consider labeled cycle cover sums in the other.

We start with a uniformly randomly chosen partition $V = V_1 \cup V_2$ with $|V_1| = |V_2|$. Let H be a Hamiltonian cycle. We partition arcs of H into two sets, $\mathcal{L}(H)$ and $\mathcal{U}(H)$. $\mathcal{L}(H)$ is the set of arcs which have at least on end in V_2, we call them *labeled* by V_2. $\mathcal{U}(H)$ consists of the remaining arcs and we call them *unlabeled* by V_2. Next, denote by $hc^m_{V_2}(G)$ the subset of $hc(G)$ consisting of those Hamiltonian cycles, which have exactly m arcs unlabeled by V_2.

For every edge $uv \in E$ such that $u \in V_2$ or $v \in V_2$ we introduce two variables x_{uv} and x_{vu}, and we identify x_{uv} with x_{vu} whenever $v \neq s$ and $u \neq s$.

We consider a complete bidirected graph $D = (V_1, F)$. We add a set L_m of size m of additional labels for labeling arcs unlabeled by V_2. We also introduce variables $x_{uv,d}$ and $x_{vu,d}$ for every edge uv in $G[V_1]$ and every element $d \in L_m$ and we let $x_{vu,d}$ coincide with $x_{uv,d}$ whenever $u \neq s$ and $v \neq s$.

Next, we define function f. Let $\mathcal{P}_{u,v}(X)$ denote the family of all simple paths from u to v in G going exactly through the vertices in X. For $uv \in F$ and $\emptyset \subset X \subseteq V_2$, we set

$$f(uv,X) = \sum_{P \in \mathcal{P}_{u,v}(X)} \prod_{wz \in P} x_{wz}.$$

Moreover, we set

$$f(uv,\{d\}) = x_{uv,d}$$

for every $d \in L_m$ and $uv \in F$ such that uv is an edge in $G[V_1]$. For all other points f vanishes.

Lemma 3.5. *For $G, D, V_2, \mathcal{U}, \mathcal{L}, m, L_m$ and f defined as above,*

1. $\Lambda(D, V_2 \cup L_m, f) = \sum_{H \in hc^m_{V_2}(G)} \left(\sum_{\sigma:\mathcal{U}(H)\to L_m} \prod_{uv \in \mathcal{U}(H)} x_{uv,\sigma(uv)}\right) \left(\prod_{uv \in \mathcal{L}(H)} x_{uv}\right)$, *with σ one–to–one.*
2. $\Lambda(D, V_2 \cup L_m, f)$ *is the zero polynomial if and only if* $hc^m_{V_2}(G) = \emptyset$.

Next we describe the Monte Carlo algorithm which detects whether a given undirected graph $G = (V, E)$ with n vertices is Hamiltonian or not.

The algorithm runs r times. In each run we choose uniformly at random a partition $V = V_1 \cup V_2$, $|V_1| = |V_2| = n/2$. Next the algorithm loops over m, the number of edges unlabeled by V_2, from 0 to m_{max}. For each m a labeled cycle cover sum $\Lambda(D, V_2 \cup L_m, f)$ is build for a bidirected graph D with $n/2$ vertices and $n/2+m$ labels $V_2 \cup L_m$ as in lemma 3.5.

Next a vector p which assigns values to the variables in f is chosen uniformly at random from $GF(2^k)$, where k is set large enough so that $2^k > cn$ for some constant $c > 1$. To count the value of the polynomial in p the algorithm tabulates the value of f in p for all subsets of V_2 using a variant of Bellman–Held–Karp recursion ([2], [7]). Finally, the algorithm evaluates $\Lambda(D, V_2 \cup L_m, f)$ in p using lemma 3.5. If the result is nonzero in at least one run then the input graph G is Hamiltonian. Otherwise we conclude that it is not.

The total running time of the algorithm is $O^*(r2^{0.5n+m_{max}})$ and the probability of false negatives is at most $Pr(\sum_{m=0}^{m_{max}} |hc_{V_2}^m(G)| = 0)^r$. By lemma 1.4 we get a positive answer with probability at least $1 - 1/c$ if only G has a Hamiltonian cycle. Therefore, setting $m_{max} = 0.205n$ and $r = n^{O(1)}2^{0.024n}$ we get a total runtime complexity to be $O^*(1.657^n)$ and exponentially small probability of false positive.

Theorem 3.6. *There is a Monte Carlo algorithm detecting whether an undirected graph on n vertices is Hamiltonian or not running in $O^*(1.657^n)$ time, with no false positives and false negatives with probability exponentially small in n.*

4 Solving the k–path problem in $O^*(1.657^k)$ time

The k–path problem asks whether a given graph G on n vertices contains a simple path of length k. In a naive way this problem can be solved in $O^*(n^k)$ time. However, Alon, Yuster and Zwick showed in [1] an algorithm which solves the problem in $O^*(c^k)$ time, with some constant $c > 8000$. This constant has been reduced in a number of papers and recently Björklund, Husfeldt, Kaski and Koivisto presented a Monte Carlo algorithm which solves the k–path problem in $O^*(1.657^k)$ time [5]. In this section, we present the outline of their algorithm.

The main idea is somewhat similar to the problem of determining Hamiltonicity. The algorithm uses the inclusion-exclusion principle to sieve over a multivariate polynomial obtained by considering labeled walks in graph G.

A *k–walk* in a graph G consists of a sequence of k vertices $v_1, \ldots, v_k$ and $k-1$ edges $e_1, \ldots, e_{k-1}$ such that $e_i = v_i v_{i+1}$, $i = 1, \ldots, k-1$. Let V_1, V_2 be a partition of the vertices of graph G and W a k–walk in G. We denote by $V_1\{W\}$ the multiset of vertices from V_1 which occur on the walk W (possibly with multiplicities). Similarly, we denote by $E_2\{W\}$ the multiset of edges which occur on the walk W and have both ends in V_2. Next, we define a *labeled k–walk* to be a k–walk in which every vertex from $V_1\{W\}$ and every edge from $E_2\{W\}$ is assigned a unique label.

The time complexity of the algorithm depends on the number of labels. Therefore we want to reduce their number and we focus only on a specific subset of k–walks that we call admissible. Let E_2 denote the set of edges of G with both ends in V_2. Let k, k_1, l_2 be nonnegative integers and s be a fixed vertex of G. We say that a k–walk $W = \{v_1, e_1, \ldots, e_{k-1}, v_k\}$ is *admissible* with parameters k, k_1, l_2, s if

1. W starts at s, that is $v_1 = s$;
2. $|V_1\{W\}| = k_1$;
3. $|E_2\{W\}| = l_2$;
4. W is $V_2EV_1EV_2$–palindromless, that is for any $i = 1, \ldots, k-1$, $e_i = e_{i+1}$ implies that either $v_i \notin V_2$ or $v_{i+1} \notin V_1$.

Lemma 4.1 (Admissibility)**.** *Let k_1, l_2 be nonnegative integers and let P be a k–path in G. For (V_1, V_2) selected uniformly at random, we have*

$$Pr\left(|V_1\{P\}| = k_1 \text{ and } |E_2\{P\}| = l_2\right) = 2^{-k}\binom{k_1+1}{k-k_1-l_2}\binom{k-k_1-1}{l_2}.$$

We use the following labeling to go from the graphical domain of vertices and edges into a set of abstract labels whose number depends not on the size of the graph but on the parameters k_1 and l_2. Let K_1 be a set of k_1 labels and L_2 be a set of l_2 labels. Next, let W be an admissible walk, $\kappa_1 : V_1\{W\} \to K_1$ and $\lambda_2 : E_2\{W\} \to L_2$ be arbitrary functions. We call the three-tuple (W, κ_1, λ_2) a *labeled admissible walk*. κ_1 and λ_2 induce a labeling of vertices from V_1 and edges from E_2 in W. If both κ_1 and λ_2 are bijections then we call this labeling *bijective*.

The sieve in the algorithm operates over a multivariate polynomial ring with the coefficient field $\mathbb{F}_{2^b}$ and the following indeterminates. For every edge $e \in E$ we introduce one indeterminate x_e. Moreover, for every pair $(v, i) \in V_1 \times K_1$ we introduce one indeterminate $y_{v,i}$. Finally, for every pair $(e, i) \in E_2 \times L_2$ we introduce one indeterminate $z_{e,i}$.

We now associate with each labeled admissible walk (W, κ_1, λ_2) a monomial

$$m(W, \kappa_1, \lambda_2) = \prod_{e \in E\{W\}} x_e \prod_{v \in V_1\{W\}} y_{v,\kappa_1(v)} \prod_{e \in E_2\{W\}} z_{e,\lambda_2(e)}.$$

Lemma 4.2 (Identifiability)**.** *The monomial $m(W, \kappa_1, \lambda_2)$ of a labeled admissible walk (W, κ_1, λ_2) uniquely determines the edges and their multiplicities of occurrence in W. In particular, any path is uniquely identified. Furthermore, if W is a path and κ_1, λ_2 are bijections, then $m(W, \kappa_1, \lambda_2)$ uniquely identifies (W, κ_1, λ_2).*

Next, we use the inclusion-exclusion principle to sieve for bijective labelings. Let $\mathcal{L}$ denote the set of all labeled admissible walks. For $I_1 \subseteq K_1$ and $J_2 \subseteq L_2$, let $\mathcal{L}[I_1, J_2]$ be the set of all labeled admissible walks that avoid labels in I_1 and J_2. Moreover, denote by $\mathcal{B}$ the set of all bijectively labeled admissible walks. Then from the inclusion-exclusion principle it follows that

$$\sum_{(W,\kappa_1,\lambda_2) \in \mathcal{B}} m(W, \kappa_1, \lambda_2) = \sum_{I_1 \subseteq K_1} \sum_{J_2 \subseteq L_2} (-1)^{|I_1|+|J_2|} \sum_{(W,\kappa_1,\lambda_2) \in \mathcal{L}[I_1,L_2]} m(W, \kappa_1, \lambda_2). \tag{4}$$

We now partition the set of all bijectively labeled admissible walks into two sets, namely walks which are paths, we denote them by $\mathcal{P}$, and walks which are non-paths, we denote them by $\mathcal{R}$. Therefore

$$\sum_{(W,\kappa_1,\lambda_2)\in\mathcal{B}} m(W,\kappa_1,\lambda_2) = \sum_{(W,\kappa_1,\lambda_2)\in\mathcal{P}} m(W,\kappa_1,\lambda_2) + \sum_{(W,\kappa_1,\lambda_2)\in\mathcal{R}} m(W,\kappa_1,\lambda_2). \quad (5)$$

By choosing appropriate pairing on the set of non-paths, one can show that the monomials coming from bijectively labeled non-paths cancel over a field of characteristic two.

To find such a pairing we use the fact that every walk W that is not a path contains at least on closed subwalk $C(W)$ with some vertex $c(W)$ being its first and at the same time its last element. To construct a pairing, for every bijectively labeled admissible walk (W,κ_1,λ_2) we need to point out a bijectively labeled admissible walk $(W',\kappa_1',\lambda_2')$ such that $m(W,\kappa_1,\lambda_2) + m(W',\kappa_1',\lambda_2') = 0$. If $c(W) \in V_1$, then it is enough to switch the labels assigned by κ_1 to $c(W)$ as the first and the last vertex on the subwalk $C(W)$, thus obtaining a new function κ_1', W and λ_2 stays the same. If $c(W) \in V_2$, then the trick is to reverse the subwalk $C(W)$, choose proper functions κ_1' and λ_2', and use the fact that W is $V_2EV_1EV_2$–palindromless.

We now describe the algorithm. Let us assume that k, k_1, l_2 have been fixed. We repeat the following procedure to decide whether there exists a k–path starting at a particular vertex s.

In each run the algorithm starts with selecting uniformly at random a partition (V_1, V_2) of the set of vertices. By lemma 4.1 we know that a fixed k–path P that starts at s is admissible with positive probability. By (4) and (5) we get a multivariate polynomial

$$\sum_{(W,\kappa_1,\lambda_2)\in\mathcal{P}} m(W,\kappa_1,\lambda_2) = \sum_{I_1\subseteq K_1} \sum_{J_2\subseteq L_2} (-1)^{|I_1|+|J_2|} \sum_{(W,\kappa_1,\lambda_2)\in\mathcal{L}[I_1,J_2]} m(W,\kappa_1,\lambda_2) \quad (6)$$

of degree at most $k - 1 + k_1 + l_2$. Lemma 4.2 ensures that this polynomial is not identically zero if and only if G has an admissible k–path starting at s. Therefore, we use lemma 1.4 and test whether this is true by evaluating the right-hand side of (6) for a random assignment of values in $\mathbb{F}_{2^b}$ to the indeterminates.

The procedure iterates over each $I_1 \subseteq K_1$ and $J_2 \subseteq L_2$ and applies dynamic programming to evaluate the rightmost sum in (6). We reduce the problem by evaluating this sum only for k–walks having a fixed "suffix" $T = (v_1', e_1', v_2', e_2', v_3')$ and later summing over all possible suffixes, i.e. we need to compute the polynomial

$$M(k,k_1,l_2,T) = \sum_{\substack{(W,\kappa_1,\lambda_2)\in\mathcal{L}[I_1,J_2] \\ T \text{ is a "suffix" of } W}} m(W,\kappa_1,\lambda_2).$$

For a logical proposition P, let $[P]$ be equal to 1 if P is true and 0 otherwise. We use the following recursion

$$
\begin{aligned}
M(k,k_1,l_2,T) &= \\
&= [T \text{ is not a } V_2EV_1EV_2\text{–palindrome}] \\
&\times \sum_{\substack{e\in E\\ e=\{v,v_1'\}}} ([v\notin V_1]+[v\in V_1]\sum_{j\in K_1\setminus I_1} y_{v,j})([e\notin E_2]+[e\in E_2]\sum_{j\in L_2\setminus J_2} z_{e,j}) \\
&\times M(k-1,k_1-[v\in V_1],l_2-[e\in E_2],(v,e,v_1',e_1',v_2')).
\end{aligned}
$$

Therefore, the evaluation of (6) can be done in $O(2^{k_1+l_2}k^3n^4)$ arithmetic operations over $\mathbb{F}_{2^b}$.

Finally, to optimize the algorithm we set the number of repetitions of the procedure to be $r = \lceil 1/P(k,k_1,l_2)\rceil$, where $P(k,k_1,l_2)$ is the probability that a k–path P starting at s is admissible with parameters k, k_1, l_2, s. Moreover, we set $b = \lceil \log_2 6k\rceil$, $k_1 = \lfloor 0.5k\rfloor$ and $l_2 = \lfloor 0.207107k\rfloor$. In consequence, the total running time of the algorithm is $O^*(1.6569^k)$.

Theorem 4.3. *The undirected k–path problem for a graph with n vertices can be solved in $O^*(1.657^n)$ time by a randomized algorithm with no false positives and constant error.*

References

[1] N. Alon, R. Yuster, and U. Zwick, Color-coding. J. Assoc. Comput. Mach. 42: 844-856, 1995.

[2] R. Bellman, Dynamic programming treatment of the travelling salesman problem, J. Assoc. Comput. Mach. 9: 61-63, 1962.

[3] A. Björklund, Determinant Sums for Undirected Hamiltonicity. 51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010 (October 23-26, 2010, Las Vegas, Nevada, USA). IEEE Computer Society 2010, pp. 173-182.

[4] A. Björklund, T. Husfeldt, Inclusion-exclusion based algorithms for graph colouring. Electronic Colloquium on Computational Complexity (ECCC) 13(044), 2006.

[5] A. Björklund, T. Husfeldt, P. Kaski, M. Koivisto, Narrow sieves for parametrized paths and packings. CoRR, abs/1007.1161, 2010.

[6] M. Fürer, S. P. Kasiviswanathan, Algorithms for Counting 2-Sat Solutions and Colorings with Applications. Electronic Colloquium on Computational Complexity (ECCC) TR05-033, 2005.

[7] M. Held, R. M. Karp, A dynamic programming approach to sequencing problems, J. Soc. Indust. Appl. Math. 10: 196-210, 1962.

[8] E. L. Lawler. A note on the complexity of the chromatic number problem. Information Processing Letters, 5(3): 66-67, 1976.

[9] J. T. Schwartz, Fast probabilistic algorithms for verification of polynomial identities, J. Assoc. Comput. Mach. 27: 701-717, 1980.

[10] R. Williams, Finding paths of length k in $O^*(2^k)$, Information Processing Letters, 109(6): 301-338, 2009.