Parameterized Algorithms for List K-Cycle*

Fahad Panolan[1] **and Meirav Zehavi**[†2]

1 **Department of Informatics, University of Bergen, Norway**
fahad.panolan@ii.uib.no
2 **Department of Informatics, University of Bergen, Norway**
meirav.zehavi@ii.uib.no

Abstract

The classic K-Cycle problem asks if a graph G, with vertex set $V(G)$, has a simple cycle containing all vertices of a given set $K \subseteq V(G)$. In terms of colored graphs, it can be rephrased as follows: Given a graph G, a set $K \subseteq V(G)$ and an injective coloring $c : K \to \{1, 2, \ldots, |K|\}$, decide if G has a simple cycle containing each color in $\{1, 2, \ldots, |K|\}$ (once). Another problem widely known since the introduction of color coding is Colorful Cycle. Given a graph G and a coloring $c : V(G) \to \{1, 2, \ldots, k\}$ for some $k \in \mathbb{N}$, it asks if G has a simple cycle of length k containing each color in $\{1, 2, \ldots, k\}$ (once). We study a generalization of these problems: Given a graph G, a set $K \subseteq V(G)$, a list-coloring $L : K \to 2^{\{1,2,\ldots,k^*\}}$ for some $k^* \in \mathbb{N}$ and a parameter $k \in \mathbb{N}$, List K-Cycle asks if one can assign a color to each vertex in K so that G would have a simple cycle (of arbitrary length) containing exactly k vertices from K with distinct colors.

We design a randomized algorithm for List K-Cycle running in time $2^k n^{\mathcal{O}(1)}$ on an n-vertex graph, matching the best known running times of algorithms for both K-Cycle and Colorful Cycle. Moreover, unless the Set Cover Conjecture is false, our algorithm is essentially optimal. We also study a variant of List K-Cycle that generalizes the classic Hamiltonicity problem, where one specifies the size of a solution. Our results integrate three related algebraic approaches, introduced by Björklund, Husfeldt and Taslaman (SODA'12), Björklund, Kaski and Kowalik (STACS'13), and Björklund (FOCS'10).

1998 ACM Subject Classification G.2.2 Graph Algorithms, I.1.2 Analysis of Algorithms

Keywords and phrases Parameterized Complexity, K-Cycle, Colorful Path, k-Path

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2016.22

1 Introduction

For a graph G, let $V(G)$ and $E(G)$ denote its vertex set and edge set respectively. The input for the classic K-Cycle problem consists of an undirected graph G and a subset $K \subseteq V(G)$ of size k for some $k \in \mathbb{N}$, and the objective is to decide whether G has a K-cycle, that is, a simple cycle that contains all of the vertices in K. In terms of (partially) colored graphs, it can be rephrased as follows. Given an undirected graph G, a set of vertices $K \subseteq V(G)$ of size k for some $k \in \mathbb{N}$ and an injective coloring $c : K \to \{1, 2, \ldots, k\}$, it asks whether G has a simple cycle that contains each color in $\{1, 2, \ldots, k\}$ (once). Another problem widely known since the introduction of the color coding method [1] is Colorful Cycle. Given an

* The research leading to these results received funding from the European Research Council (ERC) via grants Rigorous Theory of Preprocessing, reference 267959 and PARAPPROX, reference 306992.
† Part of this work was done while M. Zehavi was visiting the Simons Institute for the Theory of Computing.

36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2016).
Editors: Akash Lal, S. Akshay, Saket Saurabh, and Sandeep Sen; Article No. 22; pp. 22:1–22:15

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

undirected graph G and a coloring $c : V \to \{1, 2, \ldots, k\}$ for some $k \in \mathbb{N}$, decide whether G has a simple cycle of length k that contains each color in $\{1, 2, \ldots, k\}$ (once).

We study the parameterized complexity of a generalization of both K-Cycle and Colorful Cycle, called List K-Cycle. In List K-Cycle, the input consists of an undirected graph G, a set of vertices $K \subseteq V(G)$, a list-coloring $L : K \to 2^{\{1,2,\ldots,k^*\}}$ for some $k^* \in \mathbb{N}$ and a parameter $k \in \mathbb{N}$. We need to decide whether one can find a function $c : K \to \{1, \ldots, k^*\}$ so that $c(v) \in L(v)$ for all $v \in V(G)$ and G would have a simple cycle containing exactly k vertices from K and these vertices have distinct colors. In case $k = k^*$, it is simply requested that the cycle contains each color in $\{1, 2, \ldots, k\}$ (once). If $|K| = k = k^*$ and for all $v \in K$, $|L(v)| = 1$, we obtain K-Cycle, while if $K = V$, $k = k^*$ and for all $v \in K$, $|L(v)| = 1$, we obtain Colorful Cycle. We also consider a variant of List K-Cycle, called Exact List K-Cycle. Given an undirected graph G, a set of vertices $K \subseteq V(G)$, a list-coloring $L : K \to 2^{\{1,2,\ldots,k^*\}}$ for some $k^* \in \mathbb{N}$ and parameters $k, \ell \in \mathbb{N}$, it asks whether one can assign a color to each vertex in K so that G would have a simple cycle of length ℓ containing exactly k vertices from K and these vertices have distinct colors. Exact List K-Cycle generalizes the classic ℓ-Path and Hamiltonicity problems.

We study the problem List K-Cycle in the realm of parameterized complexity. In parameterized complexity algorithms are measured in terms of input length and a parameter, which is expected to be small. More precisely, a problem is *fixed-parameter tractable (FPT)* with respect to a parameter k if an instance of size n can be solved in time $\mathcal{O}^*(f(k)) = \mathcal{O}(f(k) \cdot \mathsf{poly}(n))$ for some function f. For more details we refer to monographs [14, 11]

Related Work. As noted by Björklund et al. [4], the K-Cycle problem has been a central topic of graph theory since the 1960's (see [20] for some references). The special cases where $k = 1$ and $k = 2$ can be solved by breadth-first search and finding a flow of size 2 between two vertices, respectively. The special case where $k = 3$ has also long been known to be solvable in linear time [15, 24]. By the work of Robertson and Seymour on Disjoint Paths [28], for any constant k, K-Cycle is solvable in polynomial time. Kawarabayashi [20] showed that K-Cycle is solvable in polynomial time also when $k = \mathcal{O}((\log \log n)^{1/10})$, where $n = |V(G)|$. The best known randomized algorithm for K-Cycle was given by Björklund et al. [4], and runs in time $\mathcal{O}^*(2^k)$. Recently, Wahlström [31] gave an alternative algorithm that solves K-Cycle in time $\mathcal{O}^*(2^k)$ and showed that the problem admits a polynomial compression. We note that for directed graphs, K-Cycle is NP-hard already when $k = 2$ [18], and that other variants of K-Cycle have also been considered in the literature. For example, Kawarabayashi et al. [21] gave an algorithm for detecting a K-cycle whose length has a given parity, and Kobayashi et al. [33] gave an algorithm for detecting an induced K-cycle in a planar graph.

The Colorful Cycle problem was introduced in the breakthrough work on color coding by Alon et al. [1]. In that paper, the authors proposed an $\mathcal{O}^*(2^k)$-time deterministic algorithm for Colorful Cycle. Although the problem became widely known, faster algorithms have not been obtained. Recently, Kowalik et al. [23] explained the source of difficulty: they showed that unless the Set Cover Conjecture [10] is false, there does not exist a constant $\epsilon > 0$ such that Colorful Cycle can be solved in time $\mathcal{O}^*((2 - \epsilon)^k)$. The conjecture states that there does not exist a constant $\epsilon > 0$ such that Set Cover can be solved in time $\mathcal{O}^*((2-\epsilon)^n)$, where n is the size of the universe. Finally, we note that the ℓ-Path problem has been extensively studied in the field of parameterized complexity, and there has been a race towards obtaining the fastest algorithm that solves it [25, 8, 1, 13, 19, 9, 22, 32, 2, 3, 17, 16, 30, 35]. Currently, the fastest algorithm (randomized) for ℓ-Path runs in time $\mathcal{O}^*(1.66^\ell)$ [2, 3].

Our Contribution. Our main result is a randomized algorithm solving List K-Cycle in time $\mathcal{O}^*(2^k)$, matching the best known running time in which one can solve K-Cycle [4] as well as Colorful Cycle [1]. We note that our algorithm can also be used to find a shortest cycle among solutions. Moreover, since List K-Cycle generalizes Colorful Cycle, we conclude that unless the Set Cover Conjecture is false, our algorithm is essentially optimal. To complement our main result, we also show that Exact List K-Cycle is solvable in time $\mathcal{O}^*(2^k 1.66^{\ell-k})$. Our results integrate three related algebraic approaches: the "unlabel inessential elements" used to solve K-Cycle [4], the "double-labeling" used to solve a problem called Graph Motif [6], and the "partitioned single-labeling" used to solve ℓ-Path and Hamiltonicity [2, 3]. More precisely, our main result integrates the first two approaches, while our Exact List K-Cycle algorithm integrates the latter two.

We remark that the algebraic technique developed in [2, 3] became a standard tool to develop parameterized algorithms (see, e.g., [11, 5, 27, 26]), yet the other two techniques not well known. To the best of our knowledge, the specific technique of [4] has no known additional applications, and the only additional applications of [6] are given in [7, 34].

2 Preliminaries

For $q \in \mathbb{N}$, let $[q] = \{1, 2, \ldots, q\}$ and $[q]_0 = \{0, 1, \ldots, q\}$. For a function $g : A \to [q]_0$, let $\mathsf{count}(g) = |\{a \in A : g(a) \neq 0\}|$. For $(i, j), (i', j') \in \mathbb{N} \times \mathbb{N}$, we say that (i, j) is smaller than (i', j'), if either (a) $i < i'$ or (b) $i = i'$ and $j < j'$. A *fixed-point-free involution* is a permutation that is its own inverse and has no fixed points. For a function $f : A \to B$ and $S \subseteq A$, let $f|_S : S \to B$ be the function such that for all $s \in S$, $f|_S(s) = f(s)$.

For a graph G, let $V(G)$ and $E(G)$ denote its vertex-set and edge-set, respectively. For $v \in V(G)$, let $N(v)$ denote its neighbor-set. A *walk* in a graph G is a sequence of vertices $v_1 \ldots v_\ell$ such that $\{v_i, v_{i+1}\} \in E(G)$ for all $i \in [\ell - 1]$. For a graph G and $e \in E(G)$, we use $G - e$ to denote the graph with vertex set $V(G)$ and edge set $E(G) \setminus \{e\}$. For a graph G and $V' \subseteq V(G)$, we use $E(V')$ to denote the set $\{\{u, v\} \in E(G) \mid u, v \in V'\}$. For a walk W, we use $V(W)$ and $E(W)$ to denote the set of vertices and edges, respectively, contained in W.

3 An Algorithm for List K-Cycle

In this section we show that albeit the "list" requirement, the time in which one can solve List K-Cycle matches the best known time in which one can solve both K-Cycle and Colorful Cycle . That is, we develop an algorithm solving List K-Cycle in time $\mathcal{O}^*(2^k)$.

Our algorithm is based on the algebraic technique which underlies the algorithm for K-Cycle by Björklund et al. [4], where one associates a polynomial over a finite field with the structure that should be found. Yet, there are essential differences between our algorithm and the one in [4]. To cancel potential solutions that "visit" vertices in $V(G) \setminus K$ more than once, Björklund et al. [4] rely on a pairing argument whose correctness is based on the requirement that computations are performed over a field of characteristic 2 – our algorithm also relies on this pairing argument. However, to ensure that each vertex in K is encountered exactly once while attempting to construct a solution, Björklund et al. [4] explicitly store the set of vertices from K that have been encountered so far. This approach cannot be taken by our algorithm, since while constructing a solution, we would have stored each vertex in K that has been encountered so far as well as its color. This could lead to an algorithm with running time $\mathcal{O}^*(\binom{|K|}{k}\binom{k^*}{k})$. This solution does not even show that List K-Cycle is FPT when parameterized by k (since $|K|$ and k^* could be significantly larger than k).

An $\mathcal{O}^*(4^k)$-time algorithm for List K-Cycle can be obtained as follows. We mark each color once it is visited to ensure that it is not visited again via the labeling-based approach of Björklund et al. [2, 3] (since k^* can be significantly larger than k) at the cost of a factor of $\mathcal{O}^*(2^k)$ in the running time. Here, we also use this labeling-based approach on top of the algorithm for K-Cycle by Björklund et al. [4] to ensure that the vertices that marked colors are not visited more than once (at the cost of a factor of $\mathcal{O}^*(2^k)$ in the running time). The remaining vertices in $V(G)$ are treated in the same manner as the algorithm for K-Cycle by Björklund et al. [4] does (at the cost of a polynomial factor). However, this is still not sufficient, since our purpose is to match the best known running times in which one can solve K-Cycle and Colorful Cycle. By using the double-labeling-based approach introduced by Björklund et al. [6] to solve the Graph Motif problem, we are able to circumvent the need to keep track of colors and vertices that mark these colors separately.

Initialization. Instead of solving List K-Cycle, we will solve the following problem: Given an n-vertex undirected graph G, a subset $K \subseteq V(G)$, a list-coloring $L : K \to 2^{\{1,2,\ldots,k^*\}}$ for some $k^* \in \mathbb{N}$, a parameter $k \in \mathbb{N}$ and two vertices $s, t \in V(G)$, List K-Path asks whether one can assign a color to each vertex in K so that G would have a simple path between s and t containing exactly k vertices from K and these vertices have distinct colors.

To solve List K-Cycle in time $\mathcal{O}^*(2^k)$, it is sufficient to show that one can solve List K-Path in time $\mathcal{O}^*(2^k)$. Indeed, an instance (G, K, L, k^*, k) is a Yes-instance of List K-Cycle if and only if there is an edge $e = \{s, t\} \in E(G)$ such that $(G - e, K, L, k^*, k, s, t)$ is a Yes-instance of List K-Path.

Structures. We define some notations which are essential for our algorithm. Let the input instance of List K-Path be (G, K, L, k^*, k, s, t). For $q \in [n] \setminus \{1\}$ and $u, v \in V(G)$, a function $f : [q] \to V(G)$ is called a (q, u, v)-*function* if f satisfies the following properties: (i) $f(1) = s$, $f(q-1) = u$ and $f(q) = v$, (ii) for all $i \in [q-1]$, $\{f(i), f(i+1)\} \in E(G)$, and (iii) for all $i \in [q-2]$ such that $f(i) = f(i+2)$ it holds that $f(i+1) \notin K$. Observe that the function f defines a walk in G, and that if f is injective, it defines a simple path in G. We will use (q, u, v)-functions to construct a simple path that is a solution. We need to consider walks rather than simple paths as during the construction, we do not want to keep information regarding vertices that have already been visited – later we will use pairing arguments (in Lemma 6) to cancel such incorrect constructions. To make the pairing argument work, we will need property (iii) of f, which is also the reason why we explicitly mention u, one vertex before the last vertex we have visited, in the definition of f.

In addition to f, to know whether we are constructing a solution, we need to know which colors in $[k^*]$ have been visited. For this purpose, we need the following term. For $k' \in [k]_0$, a (q, u, v, k')-*structure* is a pair (f, g) such that (a) f is a (q, u, v)-function, (b) $g : [q] \to [k^*]_0$ is a function such that $\mathsf{count}(g) = k'$, and (c) for every $i \in [q]$, if $f(i) \in K$ then $g(i) \in L(f(i))$ and otherwise $g(i) = 0$. In other words, for each occurrence of a vertex in K which we have visited, g specifies a color, and overall, it specifies exactly k' occurrences of colors. To simplify the presentation, given a (q, u, v, k')-structure (f, g), we define $\mathsf{col}(g) = \{i \in [q] : g(i) \neq 0\}$. That is, $\mathsf{col}(g)$ is the set of indices associated with vertices in K (which receive colors). When $v = t$ and $k' = k$, for any $u \in V(G)$, a (q, u, v, k')-structure is also called a q-*structure*.

▶ **Definition 1.** A q-structure (f, g) is called *good q-structure* if (i) f is injective, and (ii) for all $i, j \in \mathsf{col}(g)$, $g(i) \neq g(j)$ (i.e, $g|_{\mathsf{col}(g)}$ is injective).

▶ **Observation 2.** *The input instance of* List K-Path *is a* Yes*-instance if and only if there exists a good q-structure.*

Labels. We are now going to assign labels to some of the occurrences of vertices of a (q, u, v, k')-structure. Later we will use these labels to ensure that potential structures that are not good get cancelled when computations are performed over a field of characteristic 2. More precisely, we will swap distinct labels assigned to occurrences of vertices that obtained the same color, an ability that will be used by one of our pairing arguments (see Lemma 6).

We use labels for each position in a walk which corresponds to a vertex from K. For a (q, u, v, k')-structure (f, g) and $h : \mathsf{col}(g) \to [k]$, we say that a triple (f, g, h) is a (q, u, v, k')-labeling. When (f, g) is a q-structure or a good q-structure, we say that (f, g, h) is a q-labeling or good q-labeling, respectively. Finally, if h is bijective, we say that the entire triple (f, g, h) is bijective. We give special consideration to bijective functions h since such functions ensure that all good q-labelings survive and all the q-labelings which are not good cancel each other in the polynomial we construct later.

Next, we define two sets of q-labelings. First, we let B_1^q denote the set of all bijective q-labelings. Now, we let $B_2^q \subseteq B_1^q$ denotes the set of all bijective good q-labelings. Given a good q-structure (f, g), we can arbitrarily order the elements in $\mathsf{col}(g)$, and let h assign to each element in $\mathsf{col}(g)$ its location in this order. This results in a bijection h, and overall, in a bijective q-labeling (f, g, h). Thus, by Observation 2, we get the following observation.

▶ **Observation 3.** *The input instance of* List K-Path *is a* Yes*-instance if and only if there exists $q \in [n] \setminus \{1\}$ such that $B_2^q \neq \emptyset$.*

Monomials. We will now associate a monomial with each (q, u, v, k')-labeling. Towards this, we introduce the following variables. First, for each $e \in E(G)$, we introduce a variable x_e. Now, for each $v \in K$ and color $a \in L(v)$, we introduce a variable $y_{v,a}$. Finally, for each color $a \in [k^*]$ and label $i \in [k]$, we introduce a variable $z_{a,i}$. Let N be the total number of variables created. Notice that $N = n^{\mathcal{O}(1)}$. We are now ready to define monomial for each (q, u, v, k')-labeling.

▶ **Definition 4.** For a (q, u, v, k')-labeling (f, g, h), the *monomial of* (f, g, h), denoted by $m(f, g, h)$, is defined as follows.

$$m(f,g,h) = \left(\prod_{i \in [q-1]} x_{\{f(i), f(i+1)\}}\right) \cdot \left(\prod_{i \in \mathsf{col}(g)} y_{f(i),g(i)} \cdot z_{g(i),h(i)}\right).$$

On the one hand, observe that if (f, g, h) is a bijective good q-labeling, then it can be uniquely recovered from its monomial. That is, we have the following observation.

▶ **Observation 5.** *For any $q \in [n] \setminus \{1\}$ and $(f, g, h) \in B_2^q$, there does not exist $(f', g', h') \in B_1^q \setminus \{(f, g, h)\}$ such that $m(f, g, h) = m(f', g', h')$.*

When the input instance (G, K, L, k^*, k, s, t) is a No-instance of List K-Path , then we can get a fixed-point free involution on B_1^q with some properties:

▶ **Lemma 6.** *Let (G, K, L, k^*, k, s, t) be a* No*-instance of* List K-Path *and B_1^q is the set of all bijective q-labeling for any $q \in [n] \setminus \{1\}$. Then there exists a fixed-point-free involution $\phi : B_1^q \to B_1^q$ such that for all $(f, g, h) \in B_1^q$, $m(f, g, h) = m(\phi(f, g, h))$.*

Proof. Since (G, K, L, k^*, k, s, t) is a No-instance, by Observation 3, we have that $B_2^q = \emptyset$. Let $(f, g, h) \in B_1^q$. We choose a pair $(i, j) \in [q] \times [q]$ and by carefully modifying f, g and h between i and j we get a q-labeling that can be mapped to (f, g, h) by ϕ. We will consider several cases and in each case, we assume that the conditions of the previous cases are

false, define $\phi(f,g,h)$, and prove that $\phi(f,g,h) \neq (f,g,h)$, $m(f,g,h) = m(\phi(f,g,h))$ and $\phi(\phi(f,g,h)) = (f,g,h)$. Since $(f,g,h) \notin B_2^q$, we know that either (a) there exist $i,j \in \mathsf{col}(g)$, $i<j$, such that $g(i)=g(j)$ or (b) there exist $i,j \in [q], i<j$ such that $f(i)=f(j)$.

Case 1: There exist $i,j \in \mathsf{col}(g)$, $i<j$, such that $g(i)=g(j)$. Among all such pairs, that we call bad pairs of type 1, let (i,j) be the smallest one. We define $(f',g',h') = \phi(f,g,h)$ as follows. First, we simply let $f'=f$ and $g'=g$. Now, we let $h'(i)=h(j)$ and $h'(j)=h(i)$, and for all $r \in \mathsf{col}(g) \setminus \{i,j\}$, we let $h'(r)=h(r)$. Since $(f',g')=(f,g)$ is a q-structure and h' is bijective, we have that $(f,g,h') \in B_1^q$. Since h is bijective, we have that $h' \neq h$, and therefore $\phi(f,g,h) \neq (f,g,h)$. Moreover, $z_{g(i),h(i)} = z_{g'(j),h'(j)}$ and $z_{g(j),h(j)} = z_{g'(i),h'(i)}$, and therefore $m(f,g,h) = m(\phi(f,g,h))$. Finally, because $(f,g)=(f',g')$, the sets of bad pairs of type 1 of (f,g,h) and (f',g',h') are same. Also, by swapping the values assigned to i and j in h', we obtain h, and so we have that $\phi(\phi(f,g,h)) = (f,g,h)$.

Case 2: There exist two indices $i,j \in \mathsf{col}(g)$, $i<j$, such that $f(i)=f(j)$. Among all such pairs, that we call bad pairs of type 2, let (i,j) be the smallest one. Since $i \in \mathsf{col}(g)$, $f(i) \in K$. We define $(f',g',h') = \phi(f,g,h)$ as follows. First, we simply let $f'=f$. Now, we let $g'(i)=g(j)$ and $g'(j)=g(i)$, and for all $r \in [q] \setminus \{i,j\}$, we let $g'(r)=g(r)$. Finally, we let $h'(i)=h(j)$, $h'(j)=h(i)$, and for all $r \in \mathsf{col}(g) \setminus \{i,j\}$, we let $h'(r)=h(r)$. Again, it is clear that $\phi(f,g,h) \in B_1^q$. Because Case 1 is false, we have that $g(i) \neq g(j)$, and therefore $\phi(f,g,h) \neq (f,g,h)$. Moreover, since $f(i)=f(j)$, it holds that $y_{f(i),g(i)} = y_{f'(j),g'(j)}$ and $y_{f(j),g(j)} = y_{f'(i),g'(i)}$, and it also holds that $z_{g(i),h(i)} = z_{g'(j),h'(j)}$ and $z_{g(j),h(j)} = z_{g'(i),h'(i)}$. Therefore, $m(f,g,h) = m(\phi(f,g,h))$. By our definition of g', we have that there are no bad pairs of type 1. Moreover, our definitions of f' and g' ensures that the sets of bad pairs of type 2 of (f,g,h) and (f',g',h') are the same. Also, by swapping the values assigned to i and j by g' as well as h', we get g and h respectively, and so we have that $\phi(\phi(f,g,h)) = (f,g,h)$.

Case 3: Cases 1 and 2 are false. Let $I=[q]$. If there exists a pair $(i',j') \in I \times I$, $i'<j'$, such that $f(i')=f(j')$, then we choose such a pair with the following inductive priorities in the order: (1) i' is minimized and (2) j' is maximized. If $f(i')f(i'+1)\dots f(j')$ is not a palindrome, then we set $(i,j)=(i',j')$. Otherwise, we set $I := I \setminus [j']$ and continue the process of choosing a pair as above from $I \times I$. Observe that since Cases 1 and 2 are false and yet the input instance is a No-instance, at least one pair (i',j') will be chosen.

First we show that we succeed in finding a pair (i,j). Towards this we first show that for any pair (i',j') considered by the above process but not considered as the pair (i,j), we have that $f(i'), f(i'+1), \dots, f(j') \notin K$. Suppose there is a vertex $w \in K$ appearing in the sequence $f(i')f(i'+1)\dots f(j')$. We know that $f(i')f(i'+1)\dots f(j')$ is a palindrome and Case 2 is not applicable. This implies that w appears only once and it is in the middle of the palindrome $f(i')f(i'+1)\dots f(j')$. But then this will contradict property (iii) of the (q,u,t)-function f (since (f,g) is a q-structure, f is a (q,u,t)-function for some $u \in V(G)$). Let $(i_1,j_1),\dots,(i_\ell,j_\ell)$ be the pairs considered in the above process in the given order. Suppose $(i_\ell,j_\ell) \neq (i,j)$ (that is, the process terminated before we set (i,j)). Then, $f|_I$ is injective, where $I = [q] \setminus (\bigcup_{r \in [\ell]} \{i_r, i_r+1, \dots, j_r\})$. Also we know that for $i,j \in \mathsf{col}(g), i<j$, $g(i) \neq g(j)$, because Case 1 is not applicable. Then this implies that $(f|_I, g)$ is a good q'-structure for some $q'<q$, which is a contradiction because (G,K,L,k^*,k,s,t) is a No-instance. Thus we have shown that the above process succeeds in finding (i,j) such that $f(i)\cdots f(j)$ is not a palindrome.

We define $(f',g',h') = \phi(f,g,h)$ as follows. For every index $r \in [q]$ such that $r<i$ or $r>j$, we let $f'(r)=f(r)$, $g'(r)=g(r)$ and $h'(r)=h(r)$. Next, we are going to redirect the

subwalk from i to j to be from j to i, adjusting the values assigned by f, g and h accordingly. Formally, for each index $i \leq r \leq i + \lfloor (j-i)/2 \rfloor$, define $s(r) = j - (r-i)$ and $s(j-(r-i)) = r$. Now, for each index $i \leq r \leq j$, let $f'(r) = f(s(r))$, $g'(r) = g(s(r))$ and $h'(t) = h(s(t))$.

Because $f(i)f(i+1)\cdots f(j)$ is not a palindrome, we have that $\phi(f,g,h) \neq (f,g,h)$. Notice that f and f' corresponds to two walks in the graph G. Observe that because $f(i) = f(j)$, redirecting the subwalk between i and j does not change the edges which the walk contains (only the order in which we visit them changes), and since upon changing the location of an occurrence of a vertex, we update all of the three functions f, g and h accordingly, we have that $m(f,g,h) = m(\phi(f,g,h))$. We show that f' is a (q,u',t)-function for some u'. Properties (i) and (ii) of (q,u',t)-function trivially hold. The only occurrences of vertices whose adjacent occurrences of vertices might change are $f'(i)$ and $f'(j)$, and because $f'(i) \notin K$ (since Case 2 not applicable), it is still true that for all $r \in [q-2]$ such that $f'(r) = f'(r+2)$ it holds that $f'(r+1) \notin K$. Hence $(f',g',h') = \phi(f,g,h) \in B_1^q$.

Finally we prove that $\phi(f',g',h') = (f,g,h)$. Notice that for (f',g',h'), Cases 1 and 2 are not applicable. Looking at (f',g',h') and applying the process above in Case 3 will result in the same pair (i,j), since the updates that are performed with respect to I only concern indices before i (which were not change), and the only occurrences of vertices whose location has changed lie between i and j, and therefore they do not affect the minimality of i and maximality of j relevant to the choice of (i,j). Thus, since by redirecting the subwalk between i and j yet again we obtain the original walk, and hence we conclude that $\phi(\phi(f,g,h)) = (f,g,h)$. ◀

Now, for all $q \in [n]$, we define a polynomial that is evaluated over the finite field $\mathbb{F}_p$, where $p = 2^{\lceil \log(3(n+2k)) \rceil}$.

▶ **Definition 7.** $P^q = \sum_{(f,g,h) \in B_1^q} m(f,g,h)$.

▶ **Lemma 8.** *(G,K,L,k^*,k,s,t) is a* YES*-instance of* LIST K-PATH *if and only if there is $q \in [n] \setminus \{1\}$ such that P^q is not identically 0.*

Proof. Suppose (G,K,L,k^*,k,s,t) is a YES-instance and $n = |V(G)|$. Then Observation 3, there is a $q \in [n] \setminus \{1\}$ such that $B_2^q \neq \emptyset$. Then by Observation 5 P^q contains a unique monomial corresponding to (f,g,h) where $(f,g,h) \in B_2^q$. This implies that $P^q \not\equiv 0$.

Suppose (G,K,L,k^*,k,s,t) is a NO-instance, then by Lemma 6 and the fact that $\mathbb{F}_p$ has characteristic 2, we can conclude that for all $q \in [n] \setminus \{1\}$, $P^q \equiv 0$. ◀

Thus, it remains to determine whether at least one polynomial P^q is not identically 0.

Evaluation. Given $I \subseteq [k]$, a (q,u,v,k',I)-*labeling* (f,g,h) is a (q,u,v,k')-labeling whose set of assigned labels belongs to I (i.e., the image of h is a subset of I). Now, we define polynomials whose evaluations, which will be done below, can be considered as "intermediate" steps towards evaluating P^q. To this end, we let $S(q,u,v,k',I)$ denote the set of all (q,u,v,k',I)-labelings.

▶ **Definition 9.** $P^q(u,v,k',I) = \sum_{(f,g,h) \in S(q,u,v,k',I)} m(f,g,h)$.

By the inclusion-exclusion principle and because $\mathbb{F}_p$, the field over which we evaluate polynomials, has characteristic 2, we have the following observation.

▶ **Observation 10.** $P^q = \sum_{I \subseteq [k]} \sum_{u^* \in N(t)} P^q(u^*,t,k,I)$.

It is easier to compute polynomials of the form $P^q(u^*, t, k, I)$ rather than the polynomial P^q since then we do not need to ensure that labels are not repeated, but only need to ensure that certain labels are not used at all. We next show that a polynomial of the form $P^q(u^*, t, k, I)$ can be computed in polynomial time by using a procedure based on dynamic programming.

▶ **Lemma 11** ($\star$[1]). *Let $I \subseteq [k]$, $u^* \in N(t)$ and let $\{x_1, \ldots, x_N\}$ be the set of variables in $P^q(u^*, t, k, I)$. Let $a_1, a_2, \ldots, a_N \in \mathbb{F}_p$ denote the chosen values for the N variables. Then, the polynomial $P^q(u^*, t, k, I)$ can be evaluated at $(a_1, a_2, \ldots, a_N)$ in time $N^{\mathcal{O}(1)}$.*

Combining Observation 10 with Lemma 11, and since there are 2^k subsets I of $[k]$, we have, the following result.

▶ **Lemma 12.** *Given an assignment of values from $\mathbb{F}_p$ to the variables of P^q, the polynomial P^q can be evaluated in time $2^k N^{\mathcal{O}(1)}$ and in space $N^{\mathcal{O}(1)}$.*

The Algorithm. To calculate the number of evaluations we should perform, we rely on the following well-known result, proved in [12, 29, 36].

▶ **Lemma 13.** *Let $p(x_1, x_2, \ldots, x_m)$ be a nonzero polynomial of n variables and total degree at most d over the finite field $\mathbb{F}$. Then, for $a_1, a_2, \ldots, a_n \in \mathbb{F}$ selected independently and uniformly at random: $Pr(p(a_1, a_2, \ldots, a_n) \neq 0) \geq 1 - d/|\mathbb{F}|$.*

We are now ready to conclude this section with our main result.

▶ **Theorem 14.** *There is a polynomial space randomized algorithm for* List K-Cycle *running in time $\mathcal{O}^*(2^k)$ with one sided constant error probability.*

Proof. As explained earlier, we actually solve List K-Path. Our algorithm is defined as follows. Let (G, K, L, k^*, k, s, t) be the input instance and $n = |V(G)|$. It examines every length $q \in [n]$. Now, consider a specific length q. First, uniformly at random, it chooses N values from $\mathbb{F}_p$ to be assigned to the variables of P^q (recall that N is the number of variables which is bounded by $n^{\mathcal{O}(1)}$). Then, it evaluates P^q accordingly using the computation presented in Lemma 12. Finally, if the result in not zero, it accepts. Eventually, if for every length q the result is zero, it rejects.

By Lemma 8, if the input instance of List K-Path is a No -instance, for every length q the result of evaluating P^q is zero, and therefore the algorithm will reject. Now, suppose that the input instance of List K-Path is a Yes-instance. By Lemma 8, there exists $q \in [n]$ such that P^q is not identically zero. Observe that P^q is a polynomial of total degree at most $n + 2k$. Thus, by Lemma 13, the result of evaluating P^q is not zero with probability at least $2/3$. Therefore, the algorithm accepts with probability at least $2/3$. Clearly, the probability of success can be increased via multiple runs, or, in a more direct manner, by choosing a larger field (choosing a field of size $p = 2^{\lceil \log(c'(n+2k)) \rceil}$ for some constant c' will result in a success probability of at least $1 - 1/c'$). ◀

4 An Algorithm for Exact List K-Cycle

In this section we show that Exact List K-Cycle can be solved in time $\mathcal{O}^*(2^k 1.66^{\ell-k})$. On the one hand, Exact List K-Cycle generalizes Colorful Cycle – indeed, Colorful

[1] Due to space constraints, proofs of results marked with $\star$ are omitted.

Cycle is the special case of Exact List K-Cycle where $K = V(G)$, $k = k^* = \ell$ and for all $v \in K$ it holds that $|L(v)| = 1$. Therefore, unless the Set Cover Conjecture is false, we cannot obtain an algorithm that runs in time $\mathcal{O}^*((2-\epsilon)^k)$ even if $k = \ell$. On the other hand, Exact List K-Cycle generalizes ℓ-Path and Hamiltonicity (which is a special case of ℓ-Path where $\ell = |V(G)|$), and there are randomized algorithms for ℓ-Path and Hamiltonicity running in times $\mathcal{O}^*(1.66^\ell)$ and $\mathcal{O}^*(1.66^{|V(G)|})$, respectively [3, 2]. We present a multivariate algorithm for Exact List K-Cycle that can be viewed as a compromise between the dependency on k and the dependency on ℓ. Our approach employs the double-labeling-based approach of Björklund et al. [6], used to solve the Graph Motif problem, on top of the technique underlying the ℓ-Path algorithm of Björklund et al. [3].

Due to similarities between this algorithm and the one given in the previous section, we present it in a more concise manner. Again, by the explanation given in Section 3, it is sufficient to solve the following problem, called Exact List K-Path, in time $\mathcal{O}^*(2^k 1.66^{\ell-k})$. Given an n-vertex undirected graph G, a set of vertices $K \subseteq V(G)$, a list-coloring $L : K \to 2^{\{1,2,\ldots,k^*\}}$ for some $k^* \in \mathbb{N}$, parameters $k, \ell \in \mathbb{N}$ and two vertices $s, t \in V(G)$, and this problem asks whether one can assign a color to each vertex in K so that G would have a simple path of length ℓ between s and t containing exactly k vertices from K and these vertices have distinct colors.

In most of this section, we will assume that we are given a partition (V_1, V_2) of $V(G) \setminus K$, which will be computed later. Accordingly, we define the Partitioned Exact List K-Path (K-PEL Path) problem as the Exact List K-Path problem where one is also given parameters $\ell_1, \ell_2 \in [\ell]$, and the solution is also required to contain exactly ℓ_1 vertices from V_1 and ℓ_2 edges from $E(V_2)$.

Structures and Labels. Given $q \in [\ell] \setminus \{1\}$, $q_1 \in [\ell_1]_0$, $q_2 \in [\ell_2]_0$, vertices $u, v \in V(G)$ and $k' \in [k]$, a function $f : [q] \to V(G)$ is called a (q, q_1, q_2, u, v, k')-*function* if the following properties hold:

- **(i)** $f(1) = s$, $f(q-1) = u$ and $f(q) = v$,
- **(ii)** for all $i \in [q-1]$, $\{f(i), f(i+1)\} \in E(G)$,
- **(iii)** for all $i \in [q-2]$ such that $f(i) = f(i+2) \in V_2$ it holds that $f(i+1) \in V_2$,
- **(iv)** $q_1 = |V_1(f)|$, where $V_1(f) = \{i \in [q] : f(i) \in V_1\}$,
- **(v)** $q_2 = |E_2(f)|$, where $E_2(f) = \{(i, i+1) \ : \ i \in [q-1], \{f(i), f(i+1)\} \subseteq V_2\}$,
- **(vi)** $k' = |\{i \in [q] : f(i) \in K\}|$.

Observe that $V_1(f)$ and $E_2(f)$ are sets of indices. In addition to f, to know whether we are constructing a solution, we need to know which colors in $[k^*]$ have been used. For this purpose, we need the following term. For a (q, q_1, q_2, u, v, k')-function f and a function $g : [q] \to [k^*]_0$, the pair (f, g) is called (q, q_1, q_2, u, v, k')-*structure* if the following condition holds: for any $i \in [q]$, if $f(i) \in K$ then $g(i) \in L(f(i))$ and otherwise $g(i) = 0$. In other words, for each occurrence of a vertex in K which we have visited, g specifies a color, and overall, since f is a (q, q_1, q_2, u, v, k')-function, g specifies exactly k' occurrences of colors. To simplify the presentation, for a (q, q_1, q_2, u, v, k')-structure (f, g), we define $\mathsf{col}(g) = \{i \in [q] : g(i) \neq 0\}$. That is, $\mathsf{col}(g)$ is the set of indices associated with vertices in K (which receive colors). For any vertex $u \in V(G)$, a $(\ell, \ell_1, \ell_2, u, t, k)$-structure is also called a *final structure (f-structure)*. When both f and $g|_{\mathsf{col}(g)}$ are also injective, we say that it is an *excellent final structure (ef-structure)*. By the definition of ef-structures, we have the following observation.

▶ **Observation 15.** *The input instance of K-PEL Path is a Yes-instance if and only if there exists an ef-structure.*

As before, the main idea is to use labels for specific elements in (q, q_1, q_2, u, v, k')-structures. Here we use labels on $\mathsf{col}(g) \cup V_1(f) \cup E_2(f)$. For a (q, q_1, q_2, u, v, k')-structure (f, g) and a function $h : \mathsf{col}(g) \cup V_1(f) \cup E_2(f) \to [k + \ell_1 + \ell_2]$, we say that the triple (f, g, h) is a (q, q_1, q_2, u, v, k')-labeling. When (f, g) is an f-structure or an ef-structure, we say that (f, g, h) is an f-labeling or an ef-labeling, respectively. Moreover, if g is bijective, we say that the triple (f, g, h) is bijective. Next, we define two sets of f-labelings. First, we let B_1 denote the set of all bijective f-labelings. Now, we let $B_2 \subseteq B_1$ denote the set of all bijective ef-labelings.

Observe that for any ef-structure (f, g) (which corresponds to a solution), $|\mathsf{col}(g) \cup V_1(f) \cup E_2(f)| = k + \ell_1 + \ell_2$, and therefore the sizes of the domain and codomain of the function h are equal. Thus, given an ef-structure (f, g), we can arbitrarily order the elements in $\mathsf{col}(g) \cup V_1(f) \cup E_2(f)$, and let h assign to each element in $\mathsf{col}(g) \cup V_1(f) \cup E_2(f)$ its location in this order. This results in a bijective function h, and overall, in a bijective ef-labeling (f, g, h). Thus, by Observation 15, we have the following observation.

▶ **Observation 16.** *The input instance of* K-PEL Path *is a* Yes*-instance if and only if* $B_2 \neq \emptyset$.

Monomials. Now we explain how to assign monomial for each (q, q_1, q_2, u, v, k')-labeling and show that the polynomial which is sum of monomials corresponding to f-labeling will give us the answer for K-PEL Path through Polynomial Identity Testing(PIT). First we introduce the following variables. For each edge $e \in E(G)$, introduce a variable x_e. Now, for each vertex $v \in V_1$ and label $i \in [k + \ell_1 + \ell_2]$, introduce a variable $y_{v,i}$, and for each edge $e \in E(V_2)$ and label $i \in [k + \ell_1 + \ell_2]$, introduce a variable $y_{e,i}$. For each vertex $v \in K$ and color $a \in L(v)$, introduce the variable $z_{v,a}$. Finally, for each color $a \in [k^*]$ and label $i \in [k + \ell_1 + \ell_2]$, introduce the variable $z_{a,i}$. Let N be the total number of variables created.

Now, we associate a monomial with each (q, q_1, q_2, u, v, k')-labeling.

▶ **Definition 17.** Given a $(q, q_1, q_2, u, v, w, k')$-labeling (f, g, h), the *monomial of* (f, g, h), denoted by $m(f, g, h)$, is defined as follows.

$$m(f,g,h) = \left(\prod_{i\in[q-1]} x_{\{f(i),f(i+1)\}}\right)\left(\prod_{i\in V_1(f)} y_{f(i),h(i)}\right)\left(\prod_{(i,i+1)\in E_2(f)} y_{\{f(i),f(i+1)\},h((i,i+1))}\right)$$

$$\cdot\left(\prod_{i\in\mathsf{col}(g)} z_{f(i),g(i)}\cdot z_{g(i),h(i)}\right).$$

If (f, g, h) is a bijective ef-labeling, then it can be uniquely recovered from its monomial. That is, we have the following observation.

▶ **Observation 18.** *For all* $(f, g, h) \in B_2$*, there does not exist* $(f', g', h') \in B_1 \setminus \{(f, g, h)\}$ *such that* $m(f, g, h) = m(f', g', h')$.

For bijective f-labelings that are not ef-labelings, we have the following lemma.

▶ **Lemma 19.** *Let* $(G, K, L, k^*, k, s, t, \ell, V_1, V_2, \ell_1, \ell_2)$ *be a* No*-instance of* K-PEL Path *and* B_1 *is the set of all bijective f-labeling. Then there is a fixed-point-free involution* $\phi : B_1 \to B_1$ *such that for all* $(f, g, h) \in B_1$, $m(f, g, h) = m(\phi(f, g, h))$.

Proof. Since $(G, K, L, k^*, k, s, t, \ell, V_1, V_2, \ell_1, \ell_2)$ is aNo-instance of K-PEL Path , by Observation 16, $B_2 = \emptyset$. Let $(f, g, h) \in B_1$. We will consider several cases, and in each case, we assume that previous cases are false, define $\phi(f, g, h)$, and prove that $\phi(f, g, h) \neq (f, g, h)$, $m(f, g, h) = m(\phi(f, g, h))$ and $\phi(\phi(f, g, h)) = (f, g, h)$. Since $B_2 = \emptyset$ and $(f, g, h) \in B_1$, we have that (f, g) is a f-structure but not an ef-structure. This implies that either f is not injective or $g|_{\mathsf{col}(g)}$ is not injective. This implies that there exist $i, j \in [\ell], i < j$ such that either (a) $f(i) = f(j)$ or (b) $i, j \in \mathsf{col}(g)$ and $g(i) = g(j)$.

Case 1: There exists $i, j \in [\ell], i < j$ such that $f(i) = f(j) \in V_1$. Among all such pairs, called bad pairs of type 1, let (i, j) be the smallest one. We define $(f', g', h') = \phi(f, g, h)$ as follows. First, let $f' = f$ and $g' = g$. Now, let $h'(i) = h(j)$ and $h'(j) = h(i)$, and for all $r \in (\mathsf{col}(g) \cup V_1(f) \cup E_2(f)) \setminus \{i, j\}$, let $h'(r) = h(r)$. Since h is bijective, we have that $h' \neq h$, and therefore $\phi(f, g, h) = (f', g', h') \neq (f, g, h)$. Moreover, $y_{f(i),h(i)} = y_{f'(j),h'(j)}$ and $y_{f(j),h(j)} = y_{f'(i),h'(i)}$, and therefore $m(f, g, h) = m(\phi(f, g, h))$. Notice that the smallest bad pair of type 1 in (f', g', h') is (i, j). So by swapping the values assigned to i and j in h', we obtain h, and hence we have that $\phi(\phi(f, g, h)) = (f, g, h)$.

Case 2: There exists $i, j \in [\ell], i < j$ such that $f(i) = f(j) \in K$. Among all such pairs, that we call bad pairs of type 2, let (i, j) be the smallest one. We define $(f', g', h') = \phi(f, g, h)$ as follows. First, let $f' = f$. Now, let $g'(i) = g(j)$ and $g'(j) = g(i)$, and for all $r \in [q] \setminus \{i, j\}$, let $g'(r) = g(r)$. Finally, let $h'(i) = h(j)$, $h'(j) = h(i)$, and for all $r \in (\mathsf{col}(g) \cup V_1(f) \cup E_2(f)) \setminus \{i, j\}$, we let $h'(r) = h(r)$. Since h is bijective, $h \neq h'$ and $\phi(f, g, h) \neq (f, g, h)$. Moreover, since $f(i) = f(j)$, it holds that $z_{f(i),g(i)} = z_{f'(j),g'(j)}$ and $z_{f(j),g(j)} = z_{f'(i),g'(i)}$. It also holds that $z_{g(i),h(i)} = z_{g'(j),h'(j)}$ and $z_{g(j),h(j)} = z_{g'(i),h'(i)}$. Therefore, $m(f, g, h) = m(\phi(f, g, h))$. Since there is no bad pair of type 1 in (f, g, h), there is no bad pairs of type 1 in (f', g', h'). Notice that the is the smallest bad pair of type 2 in (f', g', h') is (i, j). By swapping the values assigned to i and j by g' and h' we get g and h, and hence we have that $\phi(\phi(f, g, h)) = (f, g, h)$.

Case 3: There exists $i, j \in [\ell], i < j$ such that $f(i) = f(j) \in V_2$. Among all such pairs, that we call bad pairs of type 3, let (i, j) be the smallest one. We have two sub-cases based $f(i) \dots f(j)$ is a palindrome or not.

Case 3(a): $f(i) \dots f(j)$ is a palindrome. Since $f(i) \dots f(j)$ is a walk in a simple graph G and $f(i) \dots f(j)$ is a palindrome, the length of the sequence $f(i) \dots f(j)$ is odd and $j - i$ is even. Let $r = i + \frac{j-i}{2}$. Notice that r is the middle index in the sequence $i, i + 1, \dots j$. Since Cases 1 and 2 are not applicable, we have that for all $r' \in \{i, \dots, j\} \setminus \{r\}$ $f(r') \in V_2$. Now consider the sequence $f(r - 1)f(r)f(r + 1)$. We know that $f(r - 1) = f(r + 1) \in V_2$. Thus by property (iii) of (q, q_1, q_2, u, v, k')-function, we have that $f(r) \in V_2$. Hence, we have that $f(i), \dots, f(j) \in V_2$.

Now, we define $(f', g', h') = \phi(f, g, h)$ as follows. First, let $f' = f$ and $g' = g$. Now, let $h'((i, i + 1)) = h((j - 1, j))$, $h'((j - 1, j)) = h((i, i + 1))$ and for all $r' \in (\mathsf{col}(g) \cup V_1(f) \cup E_2(f)) \setminus \{(i, i + 1), (j - 1, j)\}$, let $h'(r) = h(r)$. Since h is bijective, we have that $h' \neq h$, and therefore $\phi(f, g, h) \neq (f, g, h)$. Since $\{f(i), f(i + 1)\} = \{f(j - 1), f(j)\}$, we have that $y_{\{f(i),f(i+1)\},h((i,i+1))} = y_{\{f'(j-1),f'(j)\},h'((j-1,j))}$ and $y_{\{f(j-1),f(j)\},h((j-1,j))} = y_{\{f'(i),f'(i+1)\},h'((i,i+1))}$, and therefore $m(f, g, h) = m(\phi(f, g, h))$. Finally, because $f = f'$, it still holds that there are no bad pairs of types 1 and 2 in (f', g', h'), and the sets of bad pairs

of type 3 of (f, g, h) and (f', g', h') are same. Since by swapping the values assigned to i and j by h', we obtain h, and hence we have that $\phi(\phi(f, g, h)) = (f, g, h)$.

Case 3(b): $f(i) \dots f(j)$ is not a palindrome. We define $(f', g', h') = \phi(f, g, h)$ as follows. For any $r \in [q]$ such that $r < i$ or $r > j$, we let $f'(r) = f(r)$, $g'(r) = g(r)$. For any $r \in \mathsf{col}(g) \cup V_1(f)$, $r < i$ or $r > j$, $h'(r) = h(r)$. (In this context, recall that the domain of h is a set of indices.) For all $(i', i'+1)$, $i' < i$ or $i' \geq j$ such that $(i', i'+1) \in E_2(f)$, let $h'((i', i'+1)) = h((i', i'+1))$. Next, we are going to redirect the subwalk from i to j to be from j to i, adjusting f, g and h accordingly. Formally, for each index $i \leq r \leq i + \lfloor \frac{j-i}{2} \rfloor$, define $s(r) = j - (r - i)$ and $s(j - (r - i)) = r$. Now, for each index $i \leq r \leq j$, let $f'(r) = f(s(r))$, $g'(r) = g(s(r))$ and, if $r \in \mathsf{col}(g) \cup V_1(f)$, $h'(r) = h(s(r))$, and if $(r, r+1) \in E_2(f)$ and $r + 1 \leq j$ $h'((r, r+1)) = h(s(r), s(r+1))$. Since $f(i) \dots f(j)$ is not a palindrome, we have that $f \neq f'$ and hence, $\phi(f, g, h) \neq (f, g, h)$. Notice that f and f' defines two walks in the graph G. Observe that because $f(i) = f(j)$, redirecting the subwalk between i and j does not change the edges which the walk contains (but only the order between them changes), and since upon changing the location of an occurrence of a vertex, we update f, g and h accordingly to get f', g' and h', we have that $m(f, g, h) = m(f', g', h') = m(\phi(f, g, h))$. Finally, (f', g', h') does not have bad pairs of types 1 and 2, and it has the same bad pairs of type 3 as (f, g, h). Also, since by redirecting the subwalk between i and j yet again we obtain the original walk, and so we conclude that $\phi(\phi(f, g, h)) = (f, g, h)$.

We also need to show that $(f', g', h') \in B_1$. Towards that it is enough to show that f' is a $(\ell, \ell_1, \ell_2, u, t, k)$-function for some $u \in V(G)$. Since f is such a function, all the properties except the property (iii) of $(\ell, \ell_1, \ell_2, u, t, k)$-function hold trivially. Now we show that in fact property (iii) also true for f'. That is for all $i' \in [q-2]$, $f'(i') = f'(i'+2) \in V_2$ implies that $f'(i'+1) \in V_2$. Since f satisfies property (iii), it holds that for all $i' \notin \{i-1, i, j-1, j\}$, if $f'(i') = f'(i'+2) \in V_2$ implies that $f'(i'+1) \in V_2$. Now consider the case of $i-1$ and $f'(i-1), f'(i), f'(i+1)$. Since $f'(i) \in V_2$, statement mentioned in property (iii) holds for $i-1$. Now consider the case i and $f'(i), f'(i+1), f'(i+2)$. Notice that $f'(i) = f(i)$ and $f'(i+2) = f(j-2)$. Since (i, j) is a smallest pair with $f(i) = f(j) \in V_2$, we have that $f(i) \neq f(j-2)$ and hence $f'(i) \neq f'(i+2)$. Hence statement mentioned in property (iii) holds for i. Now consider the case of $j-1$ and $f'(j-1), f'(j), f'(j+1)$. Since $f'(j) \in V_2$, statement mentioned in property (iii) holds for $j-1$. Now consider the case of j and $f'(j), f'(j+1), f'(j+2)$. Since $f'(j) = f(j), f'(j+1) = f(j+1), f'(j+2) = f(j+2)$ and property (iii) holds for f, we have that property (iii) holds for j. Hence $(f', g', h') \in B_1$.

Case 4: $i, j \in \mathsf{col}(g)$ and $g(i) = g(j)$. Among all such pairs, that we call bad pairs of type 4, let (i, j) be the smallest one. We define $(f', g', h') = \phi(f, g, h)$ as follows. First, let $f' = f$ and $g' = g$. Now, let $h'(i) = h(j)$ and $h'(j) = h(i)$, and for all $r \in (\mathsf{col}(g) \cup V_1(f) \cup E_2(f)) \setminus \{i, j\}$, we let $h'(r) = h(r)$. Since h is bijective, we have that $h' \neq h$, and therefore $\phi(f, g, h) \neq (f, g, h)$. Moreover, $z_{g(i),h(i)} = z_{g'(j),h'(j)}$ and $z_{g(j),h(j)} = z_{g'(i),h'(i)}$, and therefore $m(f, g, h) = m(\phi(f, g, h))$. Finally, because $(f, g) = (f', g')$, there are still no bad pairs of types $1, 2$ and 3, and the sets of bad pairs of type 4 of (f, g, h) and (f', g', h') are same. Also, by swapping the values assigned to i and j in h', we obtain h, and so we have that $\phi(\phi(f, g, h)) = (f, g, h)$. ◀

Now, we define a polynomial $Q = \sum_{(f,g,h) \in B_1} m(f, g, h)$ which will be evaluated over the finite field $\mathbb{F}_p$, where $p = 2^{\lceil \log(3(\ell+\ell_1+\ell_2+2k)) \rceil}$. Since $\mathbb{F}_p$ has characteristic 2, by Observa-

tions 16 and 18, and Lemma 19, we get the following lemma (its proof is identical to the proof of Lemma 8).

▶ **Lemma 20.** *The input instance of* K-PEL PATH *is a* YES*-instance if and only if* $P \not\equiv 0$.

Evaluation For $I \subseteq [k+\ell_1+\ell_2]$, a $(q, q_1, q_2, u, v, k', I)$*-labeling* (f, g, h) is a (q, q_1, q_2, u, v, k')-labeling such that the image of h is a subset of I. Let $S(q, q_1, q_2, u, v, k', I)$ denote the set of all $(q, q_1, q_2, u, v, k', I)$-labelings. Let $P(q, q_1, q_2, u, v, k', I) = \sum_{(f,g,h)\in S(q,q_1,q_2,u,v,k',I)} m(f, g, h)$.

By the inclusion-exclusion principle and because $\mathbb{F}_p$, the field over which we evaluate polynomials, has characteristic 2, we have the following observation.

▶ **Observation 21.** $Q = \sum_{I\subseteq[\ell_1+\ell_2+k]} \sum_{u^*\in N(t)} P(\ell, \ell_1, \ell_2, u^*, t, k, I)$.

We next show that a polynomial of the form $P(\ell, \ell_1, \ell_2, u^*, t, k, I)$ can be evaluated at any point in time polynomial in N using dynamic programming, where N is an upper bound on number of variables.

▶ **Lemma 22** (⋆)**.** *Let* $I \subseteq [k]$, $u^* \in N(t)$. *Let* $x_1, \ldots, x_N$ *be the variables in* $P(\ell, \ell_1, \ell_2, u^*, t, k, I)$. *Let* $a_1, a_2, \ldots, a_N \in \mathbb{F}_p$ *denote the chosen values for the variables. Then, the polynomial* $P(\ell, \ell_1, \ell_2, u^*, t, k, I)$ *can be evaluated at* $(a_1, a_2, \ldots, a_N)$ *in time* $N^{\mathcal{O}(1)}$ *using space* $N^{\mathcal{O}(1)}$.

Combining Observation 21 with Lemma 22, and since there are $2^{\ell_1+\ell_2+k}$ subsets I of $[\ell_1 + \ell_2 + k]$, we have the following result.

▶ **Lemma 23.** *Given an assignment of values from* $\mathbb{F}_p$ *to the variables of* P, *the polynomial* P *can be evaluated in time* $2^{\ell_1+\ell_2+k}N^{\mathcal{O}(1)}$ *using space polynomial in* N.

The Algorithm. By Lemmata 13, 20 and 23, we get the following lemma (its proof is identical to Theorem 14).

▶ **Lemma 24.** *There is a polynomial space randomized algorithm for* K-PEL PATH *running in time* $\mathcal{O}^*(2^{\ell_1+\ell_2+k})$ *with one sided constant error probability.*

▶ **Lemma 25** ([3])**.** *Let* W *be a simple path of length* ℓ *in a graph* G. *Then, for a partition* (V_1, V_2) *of* $V(G)$ *chosen uniformly at random, let* $p(\ell, \ell_1, \ell_2)$ *be the probability that* $|V(W) \cap V_1| = \ell_1$ *and* $|E(W) \cap E(V_2)| = \ell_2$. *Then for any* ℓ, *we can find* $\ell'_1, \ell'_2 \in \mathbb{N}$ *such that* $\ell'_1 + \ell'_2 \leq \ell$ *and* $\frac{2^{\ell'_1+\ell'_2}}{p(\ell,\ell'_1,\ell'_2)} \leq 1.6569^{\ell}|V(G)|^c$ *for some constant* c.

We use Lemma 25 to prove the main theorem of the section.

▶ **Theorem 26.** *There is a polynomial space randomized algorithm for* EXACT LIST K-CYCLE *running in time* $\mathcal{O}^*(2^k 1.6569^{\ell-k})$ *with one sided constant error probability.*

Proof. Let (G, K, L, k^*, k, ℓ) be the input instance and $n = |V(G)|$. We describe an algorithm $\mathcal{A}$ for EXACT LIST K-CYCLE . Let ℓ', ℓ'_2 be the integers guaranteed by the Lemma 25 and let $p(\ell, \ell'_1, \ell'_2)$ be the probability mentioned in Lemma 25. Now for each $\ell_1 \leq \ell'_1$ and $\ell_2 \leq \ell'_2$, we randomly choose a partition (V_1, V_2) of $V(G) \setminus K$ and run algorithm $\mathcal{B}$ mentioned in Lemma 24 for K-PEL PATH. Algorithm $\mathcal{A}$ repeats algorithm $\mathcal{B}$ $\frac{1}{p(\ell,\ell'_1,\ell'_2)}$ many times. If at least once algorithm $\mathcal{B}$ outputs YES, then $\mathcal{A}$ outputs YES and otherwise outputs NO.

Clearly if (G, K, L, k^*, k, ℓ) is a NO instance, then our algorithm will output NO . Now suppose (G, K, L, k^*, k, ℓ) is a YES instance. Then there is a simple path W of length ℓ

with required property. We know that for a partition (V_1', V_2') of $V(G)$ chosen uniformly at random, the probability that $|V(W) \cap V_1| = \ell_1'$ and $|E(W) \cap E(V_2)| = \ell_2'$ is $p(\ell, \ell_1', \ell_2')$. This implies that there exist $\ell_1 \leq \ell_1'$ and $\ell_2 \leq \ell_2'$ such that for a partition (V_1, V_2) of $V(G) \setminus K$ chosen uniformly at random, the probability that $|V(W) \cap V_1| = \ell_1$ and $|E(W) \cap E(V_2)| = \ell_2$ is $p(\ell, \ell_1', \ell_2')$. For that choice of ℓ_1 and ℓ_2 the probability that algorithm $\mathcal{B}$ outputs YES is $p(\ell, \ell_1', \ell_2')$. Since algorithm $\mathcal{B}$ runs $\frac{1}{p(\ell,\ell_1',\ell_2')}$ many times with parameters ℓ_1 and ℓ_2, algorithm $\mathcal{A}$ outputs YES with constant probability.

The running time of algorithm $\mathcal{A}$ is upper bounded by $\frac{2^{\ell_1'+\ell_2'+k}n^{c'}}{p(\ell,\ell_1',\ell_2')}$ for some constant c'. By Lemma 25, we have that $\frac{2^{\ell_1'+\ell_2'}}{p(\ell,\ell_1',\ell_2')} \leq 1.6569^{\ell}n^{c}$, where c is some constant. This implies that the running time of algorithm $\mathcal{A}$ is upper bounded by $\mathcal{O}^*(2^k 1.6569^{\ell-k})$. Since algorithm $\mathcal{B}$ uses only polynomial space, $\mathcal{A}$ is a polynomial space algorithm. This completes the proof of the theorem. ◀

References

1 N. Alon, R. Yuster, and U. Zwick. Color coding. *J. ACM*, 42(4):844–856, 1995.

2 A. Björklund. Determinant sums for undirected hamiltonicity. *SIAM J. Comput.*, 43(1):280–299, 2014.

3 A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto. Narrow sieves for parameterized paths and packings. *CoRR abs/1007.1161*, 2010.

4 A. Björklund, T. Husfeldt, and N. Taslaman. Shortest cycle through specified elements. In *SODA*, pages 1747–1753, 2012.

5 A. Björklund, V. Kamat, L. Kowalik, and M. Zehavi. Spotting trees with few leaves. In *ICALP*, pages 243–255, 2015.

6 A. Björklund, P. Kaski, and L. Kowalik. Constrained multilinear detection and generalized graph motifs. *Algorithmica*, 74(2):947–967, 2016.

7 A. Björklund, P. Kaski, J. Lauri, and L. Kowalik. Engineering motif search for large graphs. In *ALENEX*, pages 104–118, 2016.

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

9 J. Chen, J. Kneis, S. Lu, D. Mölle, S. Richter, P. Rossmanith, S. H. Sze, and F. Zhang. Randomized divide-and-conquer: Improved path, matching, and packing algorithms. *SIAM J. on Computing*, 38(6):2526–2547, 2009.

10 M. Cygan, H. Dell, D. Lokshtanov, D. Marx, J. Nederlof, Y. Okamoto, P. Paturi, S. Saurabh, and M. Wahlström. On problems as hard as CNF-SAT. In *CCC*, pages 74–84, 2012.

11 M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized algorithms*. Springer, 2015.

12 R. A. DeMillo and R. J. Lipton. A probabilistic remark on algebraic program testing. *Inform. Process Lett.*, 7(4):193–195, 1978.

13 P. Deshpande, R. Barzilay, and D. R. Karger. Randomized decoding for selection-and-ordering problems. In *HLT-NAACL*, pages 444–451, 2007.

14 R. Downey and M. Fellows. *Fundamentals of parameterized complexity*. Springer, 2013.

15 H. Fleischner and G. H. Woeginger. Detecting cycles through three fixed vertices in a graph. *Inform. Process Lett.*, 41:29–33, 1992.

16 F. V. Fomin, D. Lokshtanov, F. Panolan, and S. Saurabh. Representative sets of product families. In *ESA*, pages 443–454, 2014.

17 F. V. Fomin, D. Lokshtanov, and S. Saurabh. Efficient computation of representative sets with applications in parameterized and exact agorithms. In *SODA*, pages 142–151, 2014.

18 S. Fortune, J. Hopcroft, and J. Wyllie. The directed subgraph homeomorphism problem. *Theor. Comput. Sci.*, 10:111–121, 1980.

19 F. Hüffner, S. Wernicke, and T. Zichner. Algorithm engineering for color-coding with applications to signaling pathway detection. *Algorithmica*, 52(2):114–132, 2008.

20 K. Kawarabayashi. An improved algorithm for finding cycles through elements. In *IPCO*, pages 374–384, 2008.

21 K. Kawarabayashi, Z. Li, and B. A. Reed. Recognizing a totally odd k_4-subdivision, parity 2-disjoint rooted paths and a parity cycle through specified elements. In *SODA*, pages 318–328, 2010.

22 I. Koutis. Faster algebraic algorithms for path and packing problems. In *ICALP*, pages 575–586, 2008.

23 L. Kowalik and J. Lauri. On finding rainbow and colorful paths. *Theor. Comput. Sci.*, 2016.

24 A. S. LaPaugh and R. L. Rivest. The subgraph homomorphism problem. *J. Comput. Sys. Sci.*, 20:133–149, 1980.

25 B. Monien. How to find long paths efficiently. *Annals Disc. Math.*, 25:239–254, 1985.

26 R. Y. Pinter, H. Shachnai, and M. Zehavi. Improved parameterized algorithms for network query problems. In *IPEC*, pages 294–306, 2014.

27 R. Y. Pinter and M. Zehavi. Algorithms for topology-free and alignment network queries. *JDA*, 27:29–53, 2014.

28 N. Robertson and P. D. Seymour. Graph minors XIII: The disjoint paths problem. *J. Combin. Theory Ser. B*, 63:65–110, 1995.

29 J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. Assoc. Comput. Mach.*, 27(4):701–717, 1980.

30 H. Shachnai and M. Zehavi. Representative families: A unified tradeoff-based approach. In *ESA*, pages 786–797, 2014.

31 M. Wahlström. Abusing the Tutte matrix: an algebraic instance compression for the k-set-cycle problem. In *STACS*, pages 341–352, 2013.

32 R. Williams. Finding paths of length k in $O^*(2^k)$ time. *Inf. Process. Lett.*, 109(6):315–318, 2009.

33 Kobayashi Y. and K. Kawarabayashi. Algorithms for finding an induced cycle in planar graphs and bounded genus graphs. In *SODA*, pages 1146–1155, 2009.

34 M. Zehavi. Parameterized algorithms for the module motif problem. In *MFCS*, pages 825–836, 2013.

35 M. Zehavi. Mixing color coding-related techniques. In *ESA*, pages 1037–1049, 2015.

36 R. Zippel. Probabilistic algorithms for sparse polynomials. In *EUROSAM*, pages 216–226, 1979.