Fixed-Parameter Tractability of Maximum Colored Path and Beyond

Fedor V. Fomin† Petr A. Golovach† Tuukka Korhonen† Kirill Simonov‡ Giannos Stamoulis§

Abstract

We introduce a general method for obtaining fixed-parameter algorithms for problems about finding paths in undirected graphs, where the length of the path could be unbounded in the parameter. The first application of our method is a randomized algorithm, that given a colored $n$-vertex undirected graph, vertices $s$ and $t$, and an integer $k$, finds an $(s,t)$-path containing at least $k$ different colors in time $2^kn^{O(1)}$. This is the first FPT algorithm for this problem, and it generalizes the algorithm of Björklund, Husfeldt, and Taslaman [SODA 2012] on finding a path through $k$ specified vertices. It also implies the first $2^kn^{O(1)}$ time algorithm for finding an $(s,t)$-path of length at least $k$.

Our method yields FPT algorithms for even more general problems. For example, we consider the problem where the input consists of an $n$-vertex undirected graph $G$, a matroid $M$ whose elements correspond to the vertices of $G$ and which is represented over a finite field of order $q$, a positive integer weight function on the vertices of $G$, two sets of vertices $S,T \subseteq V(G)$, and integers $p,k,w$, and the task is to find $p$ vertex-disjoint paths from $S$ to $T$ so that the union of the vertices of these paths contains an independent set of $M$ of cardinality $k$ and weight $w$, while minimizing the sum of the lengths of the paths. We give a $2^{p+O(k^2 \log(q+k))}n^{O(1)}w$ time randomized algorithm for this problem.

*The research leading to these results has received funding from the Research Council of Norway via the project BWCA (grant no. 314528). Kirill Simonov acknowledges support by the Austrian Science Fund (FWF, project Y1329). Giannos Stamoulis acknowledges support by the ANR project ESIGMA (ANR-17-CE23-0010) and the French-German Collaboration ANR/DFG Project UTMA (ANR-20-CE92-0027).

†Department of Informatics, University of Bergen, Norway.
‡Algorithms and Complexity Group, TU Wien, Austria.
§LIRMM, Université de Montpellier, CNRS, France.
1 Introduction

The study of long cycles and paths in graphs is a popular research direction in parameterized algorithms. Starting from the color-coding of Alon, Yuster, and Zwick [1], powerful algorithmic techniques have been developed [2, 3, 15, 16, 21, 29, 46, 49], see also [14, Chapter 10], for finding long cycles and paths in graphs. However, most of the known methods are applicable only in the scenario when the size of the solution is bounded by the parameter. Let us explain what we mean by that by the following example.

Consider two very related problems, \textit{k-Cycle} and \textit{Longest Cycle}. In both problems, we are given a graph \(G\) and an integer parameter \(k\). In \textit{k-Cycle} we ask whether \(G\) has a cycle of length \textit{exactly} \(k\). In \textit{Longest Cycle}, we ask whether \(G\) contains a cycle of length \textit{at least} \(k\). While in the first problem any solution should have exactly \(k\) vertices, in the second problem the solution could be even a Hamiltonian cycle on \(n\) vertices. The essential difference in applying color-coding (and other methods) to these problems is that for \textit{k-Cycle}, a random coloring of the vertices of \(G\) in \(k\) colors will color the vertices of a solution cycle with different colors with probability \(e^{-k}\). Such information about colorful solutions allows dynamic programming to solve \textit{k-Cycle} (as well as the related \textit{k-Path} problem, the problem of finding a path of length \textit{exactly} \(k\)). However, since a solution cycle for \textit{Longest Cycle} is not upper-bounded by a function of \(k\), the coloring argument falls apart. As Fomin et al. write in [21] “This is why color-coding and other techniques applicable to \textit{k-Path} do not seem to work here.” Sometimes, like in the case of \textit{Longest Cycle}, a simple “edge contraction” trick, see [14, Exercise 5.8], allows reducing the problem to \textit{k-Cycle}. We are not aware of general methods for solving problems related to cycles and paths when the size of the solution is not upper-bounded by the parameter.

The main result of this paper is a theorem that allows deriving algorithms for various parameterized problems about paths, cycles, and beyond, in the scenario when the size of the solution is not upper-bounded by the parameter. We discuss numerous applications of the theorem in the next subsection.

Our theorem is about finding a \textit{k-colored \((S,T)\)-linkage} in a colored graph. Let \(G\) be a graph, \(S\) and \(T\) be sets of vertices of \(G\), and \(p\) be a positive integer. An \((S,T)\)-linkage of order \(p\) is a set \(\mathcal{P}\) of \(p = |\mathcal{P}|\) vertex-disjoint paths, each starting in \(S\) and ending in \(T\). The set of vertices in the paths of \(\mathcal{P}\) is denoted by \(V(\mathcal{P})\). The total length (or often just the length) of an \((S,T)\)-linkage is the total number of vertices in its paths, i.e., \(|V(\mathcal{P})|\). For a coloring \(c: V(G) \to [n]\) of \(G\), an \((S,T)\)-linkage \(\mathcal{P}\) is called \textit{k-colored} if \(V(\mathcal{P})\) contains at least \(k\) different colors, i.e., \(|c(V(\mathcal{P}))| \geq k\).

Let us note that in the above definition the sets \(S\) and \(T\) are not necessarily disjoint and that the coloring \(c\) is not necessarily a proper coloring in the graph-coloring sense. We also note that for vertices \(s,t \in V(G)\), an \((\{s\}, \{t\})\)-linkage of order 1 corresponds to an \((s,t)\)-path.

\textbf{Theorem 1.} There is a randomized algorithm, that given as an input an \(n\)-vertex graph \(G\), a coloring \(c: V(G) \to [n]\) of \(G\), two sets of vertices \(S,T \subseteq V(G)\), and integers \(p,k\), in time \(2^{k+p}n^{O(1)}\) either returns a \(k\)-colored \((S,T)\)-linkage of order \(p\) and of the minimum total length, or determines that \(G\) has no \(k\)-colored \((S,T)\)-linkage of order \(p\).

Few remarks are in order. First, Theorem 1 cannot be extended to directed graphs. It is easy to show, see Proposition 1, that finding a 2-colored \((s,t)\)-path in a 2-colored directed graph is already \text{NP}-hard. Second, by another simple reduction, see Proposition 2, it can also be observed that if the time complexity of Theorem 1 could be improved to \((2-\varepsilon)^{k+p}n^{O(1)}\) for \(\varepsilon > 0\), even in the case when \(p = 1\), \(G\) is colored with \(k\) colors, and \(S = T = V(G)\), then \text{Set Cover}

\footnote{In this paper, graphs are assumed to be undirected if it is not explicitly mentioned to be otherwise.}
would admit a \((2 - \varepsilon)^n (mn)^{O(1)}\) time algorithm, contradicting the Set Cover Conjecture (SeCoCo) of Cygan et al. \cite{cygan2015parameterized}. We also remark that actually we prove an even more general result than Theorem 1, our result in full generality will be stated as Theorem 4. It can be also observed that by a simple reduction that subdivides edges, the coloring could be on the edges of \(G\) instead of vertices (or on both vertices and edges).

The algorithm in Theorem 1 invokes DeMillo-Lipton-Schwartz-Zippel lemma for polynomial identity testing and thus is “heavily” randomized. We do not know whether Theorem 1 could be derandomized. The special case of Theorem 1 when the coloring is a bijection, the problem of finding an \((S,T)\)-linkage of order \(p\) and of length at least \(k\), can be reduced to the (rooted) topological minor containment. To see why, observe that if we enumerate all possible collections \(P\) of \(p\) paths of total length \(k\), then we can check for each collection \(P\) if it is contained as a rooted topological minor in \(G\). The topological minor containment admits a deterministic FPT algorithm parameterized by the size of the pattern graph \cite{grohe2010query}. However the running time of the algorithm of Grohe et al. \cite{grohe2010query} is bounded by a tower of exponents in \(k\) and \(p\). Our next theorem gives a deterministic algorithm for computing an \((S,T)\)-linkage of order \(p\) and of length at least \(k\) whose running time is single-exponential in the parameter \(k\) for any fixed value of \(p\). The other advantage of the algorithm in Theorem 2 is that it works on directed graphs too. In the following statement, a directed \((S,T)\)-linkage is defined analogously to an \((S,T)\)-linkage, but is composed of directed paths from \(S\) to \(T\).

**Theorem 2.** There is a deterministic algorithm that, given an \(n\)-vertex digraph \(G\), two sets of vertices \(S, T \subseteq V(G)\), an integer \(p\), and an integer \(k\), in time \(p^{O(kp)} n^{O(1)}\) either returns a directed \((S,T)\)-linkage of order \(p\) and of total length at least \(k\), or determines that \(G\) has no directed \((S,T)\)-linkage of order \(p\) and total length at least \(k\).

### 1.1 Applications of Theorem 1

Theorem 1 implies FPT algorithms for several problems. It encompasses a number of fixed-parameter-tractability results and improves the running times for several fundamental well-studied problems.

**Longest path/cycle.** When the coloring \(c : V(G) \to [n]\) is a bijection, and thus all vertices of \(G\) are colored in different colors, then an \((S,T)\)-linkage is \(k\)-colored if and only if its length is at least \(k\). In this case, Theorem 1 outputs an \((S,T)\)-linkage of order \(p\) with at least \(k\) vertices in time \(2^k n^{O(1)} + kp n^{O(1)}\). In particular, for \(p = 1\) it implies that LONGEST \((s,t)\)-Path (i.e., for \(s, t \in V(G)\) and \(k \geq 0\), to decide whether there is an \((s,t)\)-path of length at least \(k\)) is solvable in time \(2^k n^{O(1)}\). Since one can solve LONGEST CYCLE (to decide whether \(G\) contains a cycle of length at least \(k\)) by solving for every edge \(st \in E(G)\) the LONGEST \((s,t)\)-Path problem, Theorem 1 also yields an algorithm solving LONGEST CYCLE in time \(2^k n^{O(1)}\). To the best of our knowledge, the previous best known algorithm for LONGEST \((s,t)\)-Path runs in time \(4.884^k n^{O(1)}\) \cite{fomin2010exact} and the previous best known algorithm for LONGEST CYCLE runs in time \(1.66^{2k} n^{O(1)} = 2.76^k n^{O(1)}\) \cite{bjoerklund2011fpt, zehavi2013approximation}. The latter algorithm follows by combining the result of Zehavi \cite{zehavi2013approximation} stating that LONGEST CYCLE is solvable in time \(t(G,2k)n^{O(1)}\), where \(t(G,k)\) is the best known running time for solving \(k\)-Path, with the fastest algorithm for \(k\)-Path of Björklund et al. \cite{bjoerklund2011fpt}.

For \(p = 2\), the problem of finding an \((S,T)\)-linkage of length at least \(k\) is equivalent to the problem of finding a cycle of length at least \(k\) passing through a given pair of vertices \(s, t\). A randomized algorithm of running time \((2e)^k n^{O(1)}\) for this problem, known as LONGEST \((s,t)\)-Cycle, was given by Fomin et al. in \cite[Theorem 4]{fomin2012improved} (see also \cite{fomin2014improved}).
As we already have mentioned the problem of finding an $(S,T)$-linkage of order $p$ and of length at least $k$ can be reduced to the (rooted) topological minor containment. For $p \geq 3$, Theorems 1 and 2 provide the first (randomized and deterministic) single-exponential in $k + p$ and single-exponential in $k$ for constant $p$, respectively, algorithms for computing an $(S,T)$-linkage of order $p$ and of length at least $k$. For directed graphs, Theorem 2 gives the first FPT algorithm for the problem parameterized by $k + p$.

$T$-cycle. In the $T$-Cycle problem, we are given a graph $G$ and a set $T \subseteq V(G)$ of terminals. The task is to decide whether there is a cycle passing through all terminals $[4, 28, 45]$. By the celebrated result of Bjorklund, Husfeldt, and Taslaman $[4]$, $T$-Cycle is solvable in time $2^{O(T)}n^O(1)$, and their algorithm in fact returns the shortest such cycle. To solve $T$-Cycle as an application of Theorem 1, we do the following. We pick a terminal vertex $t \in T$, create a twin vertex $s$ of $t$ (i.e., a vertex $s$ with $N(s) = N(t)$), and color $s$ and $t$ with color 1. We then color all non-terminal vertices of $G$ with color 1 too. The remaining terminal vertices $T \setminus \{t\}$ we color in $|T| - 1$ colors from 2 to $|T|$, such that no color repeats twice. Then $G$ has a $T$-cycle if and only if there is a $|T|$-colored $(\{s\}, \{t\})$-linkage of order 1. Therefore, using the algorithm of Theorem 1, we can also find the shortest $T$-cycle in time $2^{O(T)}n^O(1)$. One could use Theorem 1 to generalize the algorithmic result of Bjorklund, Husfeldt, and Taslaman in different settings. For example, instead of a cycle passing through all terminal vertices, we can ask for a cycle containing at least $k$ terminals from a set $T$ of unbounded size, in time $2^k n^{O(1)}$.

Another generalization of $T$-Cycle comes from covering terminal vertices by at most $p$ disjoint cycles. For example, in the basic VRP (vehicle routing problem) one wants to route $p$ vehicles, one route per vehicle, starting and finishing at the depot so that all the customers are supplied with their demands and the total travel cost is minimized $[10]$. In the simplified situation when the clients are viewed as terminal vertices $T$ of a graph and routes in VRP are required to be disjoint, this problem turns into the problem of finding a “$p$-flower” of minimum total length containing all vertices of $T$. By $p$-flower we mean a family of $p$ cycles that intersect only in one (depot) vertex $s$. To see this problem as a problem of finding a colored $(S,T)$-linkage, we replace the depot $s$ by a set $S$ of $2p$ vertices whose neighbors are identical to the neighbors of $s$. Then similar to $T$-Cycle, this variant of VRP reduces to computing a minimum length $(|T| + 1)$-colored $(S,S)$-linkage of order $p$; thus it is solvable in time $2^{O(T)}p n^{O(1)}$ by Theorem 1.

Colored paths and cycles. The problems of finding a path, cycle, or another specific subgraph in a colored graph with the maximum or the minimum number of different colors appear in different subfields of algorithms, graph theory, optimization, and operations research $[6, 8, 11, 12, 25, 31, 32, 42, 47]$. In particular, the seminal color-coding technique of Alon, Yuster, and Zwick $[1]$, builds on an algorithm finding a colorful path in a $k$-colored graph, that is, a path of $k$ vertices and $k$ colors, in time $O(2^k n)$.

In the Maximum Colored $(s,t)$-Path problem, we are given a graph $G$ with a coloring $c : V(G) \to [n]$ and integer $k$. The task is to identify whether $G$ contains a $k$-colored $(s,t)$-path, i.e., an $(s,t)$-path with at least $k$ different colors. In the literature, this problem is also known as Maximum Labeled Path $[12]$ and Maximum Tropical Path $[11]$. Theorem 1 yields the first FPT algorithm for Maximum Colored $(s,t)$-Path, as well as for Maximum Colored Cycle (decide whether $G$ contains a $k$-colored cycle). It is also the first FPT algorithm for the even more restricted variant of deciding if a given $k$-colored graph contains any $k$-colored path. A recent paper of Cohen et al. $[11]$ claims a $O(2^k n^2)$ time deterministic algorithm for computing a shortest $k$-colored path in a given $k$-colored graph. Unfortunately, a closer inspection of the algorithm of Cohen et al. reveals that it computes a $k$-colored walk instead of a $k$-colored path.$^2$

$^2$The error in $[11]$ occurs on p. 478. It is claimed that if $P$ is a shortest $(u,v)$-path that uses the set $C$ of colors
It is interesting to note that the minimization version of the colored \((s,t)\)-path, i.e., to decide whether there is an \((s,t)\)-path containing at most \(k\) different colors, is \(W[1]\)-hard even on very restricted classes of graphs [18].

**Beyond graphs: frameworks.** Frameworks provide a natural generalization of colored graphs. Following Lovász [35], we say that a pair \((G, M)\), where \(G\) is a graph and \(M = (V(G), \mathcal{I})\) is a matroid on the vertex set of \(G\), is a framework. Then we seek for a path, cycle, or \((S,T)\)-linkage in \(G\) maximizing the rank function of \(M\). Note that frameworks \((G, M)\) where \(M\) is a partition matroid generalize colored graphs. Indeed, the universe \(V(G)\) of \(M\) is partitioned into color classes \(L_1, \ldots, L_n\) and a set \(I\) is independent if \(|I \cap L_i| \leq 1\) for every color \(i \in [n]\). However, by plugging different types of matroids into the definition of the framework, we obtain problems that cannot be captured by colored graphs.

Frameworks, under the name pregeometric graphs, were used by Lovász in his influential work on representative families of linear matroids [34]. The problem of computing maximum matching in frameworks is strongly related to the matroid, the matroid parity, and polymatroid matching problems. See the Matching Theory book of Lovász and Plummer [36] for an overview. In their book, Lovász and Plummer use the term matroid graph for frameworks. In his most recent monograph, [35], Lovász introduces the term frameworks, and this is the term we adopt in our work. More generally, the problems of computing specific subgraphs of large ranks in a framework, belong to the broad class of problems about submodular function optimization under combinatorial constraints [7, 9, 40].

Let \((G, M)\) be a framework and let \(r: 2^{V(G)} \to \mathbb{Z}_{\geq 0}\) be the rank function of the matroid \(M\). The rank of a subgraph \(H\) of \(G\) is \(r(V(H))\) and we denote it by \(r(H)\). We say that an \((S,T)\)-linkage \(\mathcal{P}\) in a framework \((G, M)\) is \(k\)-ranked if the rank of \(\mathcal{P}\), that is the rank in \(M\) of the elements corresponding to the vertices of the paths of \(\mathcal{P}\), is at least \(k\). With additional work involving (lossy) randomized truncation of the matroid, it is possible to extend Theorem 1 from colored graphs to frameworks over a general class of representable matroids.

**Theorem 3.** There is a randomized algorithm that, given a framework \((G, M)\), where \(G\) is an \(n\)-vertex graph and \(M\) is represented as a matrix over a finite field of order \(q\), sets of vertices \(S,T \subseteq V(G)\), and an integer \(k\), in time \(2^p + O(k^2 \log(q+k))n^{O(1)}\) either finds a \(k\)-ranked \((S,T)\)-linkage of order \(p\) and of minimum total length, or determines that \((G, M)\) has no \(k\)-ranked \((S,T)\)-linkage of order \(p\).

With minor adjustments, Theorem 3 can be adapted for frameworks with matroids that are in general not representable over a finite field of small order. For example, uniform matroids, and more generally transversal matroids, are representable over a finite field, but the field of representation must be large enough. Despite this, we can apply Theorem 3 to transversal matroids. Similarly, it is possible to apply Theorem 3 in the situation when \(M\) is represented by an integer matrix over rationals with entries bounded by \(n^{O(k)}\).

**Weighted extensions.** Theorem 1 can be extended into a weighted version in two different settings. The first setting is to have weights on edges that affect the length of the \((S,T)\)-linkage. It is easy to see that by subdividing edges, coloring the subdivision vertices with a new “dummy color”, and increasing \(k\) by one, all our algorithms work in the setting when the edges have polynomially-bounded positive integer weights.

The second weighted extension is more interesting. It is to have weights on vertices, and asking for an \((S,T)\)-linkage containing a combination of weights and colors in a specific sense. In this setting if \(P'\) is a \((w,t)\)-sub-path of \(P\) using colors \(C' \subseteq C\), then \(P'\) must be a shortest \((w,t)\)-path among all \((w,t)\)-paths using colors \(C'\). This claim is correct for walks but not for paths.
setting, we have in addition to the coloring $c : V(G) \to [n]$ a weight function $w : V(G) \to \mathbb{Z}_{\geq 1}$. For integers $k, w$, we say that an $(S,T)$-linkage $P$ is $(k, w)$-colored if its vertices $V(P)$ contain a set $X \subseteq V(P)$ so that $|X| = k$, all vertices of $X$ have different colors, and the total weight of $X$ is exactly $w(X) = \sum_{v \in X} w(v) = w$. This weighted version does not follow by direct reductions, but instead by a modification of Theorem 1 (in our main proof, we will directly prove Theorem 4 instead of Theorem 1).

**Theorem 4.** There is a randomized algorithm that, given as an input an $n$-vertex graph $G$, a coloring $c : V(G) \to [n]$ of $G$, a weight function $w : V(G) \to \mathbb{Z}_{\geq 1}$, two sets of vertices $S,T$, and three integers $p, k, w$, in time $2^{k+p+O(1)}w$ either returns a $(k, w)$-colored $(S,T)$-linkage of order $p$ and of minimum total length, or determines that no $(k, w)$-colored $(S,T)$-linkages of order $p$ exist.

Note that Theorem 4 implies Theorem 1 by setting all vertex weights to 1 and $w = k$. Theorem 4 allows to derive some applications of our technique that do not directly follow from Theorem 1, which we proceed to describe.

**Longest T-cycle.** Recall that in the $T$-Cycle problem the task is to find a cycle passing through a given set $T$ of terminal vertices. Both the algorithm of Björklund, Husfeldt, and Taslaman [4], and the application of the algorithm of Theorem 1 find in fact the shortest $T$-cycle. A natural generalization of the $T$-Cycle problem is the Longest $T$-Cycle problem, where in addition to the set $T$ we are given an integer $k$ and the task is to find a cycle of length at least $k$ passing through the terminals $T$. Theorem 4 can be used to solve Longest $T$-Cycle in time $2^{\max(|T|,k)p+O(1)}$ as follows. First, if $|T| \geq k$, any $T$-cycle has length at least $k$ and we just use the algorithm for $T$-Cycle. Otherwise, like in the reduction for $T$-Cycle, we first pick a terminal $t \in T$ and create a twin $s$ of it. Then, we color $s$ and $t$ with color 1, and all the other vertices with different colors from 2 to $n$. We also assign weight 3 to the terminal vertices $T$, weight 1 to the vertex $s$, and weight 2 to all other vertices. We invoke Theorem 4 to find an $(\{s\}, \{t\})$-linkage of order 1 that contains a set $X$ of vertices with distinct colors, size $|X| = k$, and weight $w(X) = 2k + |T|$. Any such set $X$ must be a superset of $T$ and not contain $s$, and therefore the found path must correspond to a cycle of length at least $k$ passing through the terminals $T$.

**Vehicle routing with profits.** With Theorem 4, we can give an algorithm for the vehicle routing problem in a more general setting. In particular, we consider the situation where the depot has $k$ parcels, $p$ vehicles, and for each vertex $v$ we know that we obtain a profit $w(v)$ for delivering a parcel to that vertex. We can use Theorem 4 with the same reduction as used for VRP earlier, but instead letting the coloring of the vertices to be a bijection, to obtain a $2^{k+p+O(1)}w$ time algorithm for determining the shortest routing by cycles intersecting only at the depot that yields a total profit of $w$.

**Longest k-colored (S,T)-linkage.** Theorem 4 can be also used to derive a longest path version of Theorem 1, in particular an algorithm that given a graph $G$, a coloring $c : V(G) \to [n]$, two sets of vertices $S,T \subseteq V(G)$, three integers $k, p, \ell$, in time $2^{p+\ell+kn+O(1)}$ outputs a $k$-colored $(S,T)$-linkage of order $p$ and length at least $\ell$. The reduction is as follows. First, if $p \geq \ell$, then any $(S,T)$-linkage of order $p$ has length at least $\ell$, so we can use Theorem 1. Otherwise, we are looking for a $k$-colored $(S,T)$-linkage that contains at least $\ell - p$ edges. We subdivide every edge, and for each created subdivision vertex we assign a new color and weight $2k$. For the original vertices we keep their colors and assign weight 1. Now, any $k$-colored $(S,T)$-linkage of order $p$ corresponds to an $(S,T)$-linkage of order $p$ that contains a set $X$ of vertices with distinct colors, size $|X| = k + \ell - p$, and weight exactly $w(X) = (\ell - p) \cdot 2k + k$ (note that here we use the property that we are looking for an exact weight instead of maximum weight).
**Weighted frameworks.** We consider a generalization of frameworks into weighted frameworks. In particular, we say that a triple \((G, M, \text{we})\), where \(G\) is a graph, \(M = (V(G), T)\) is a matroid, and \(\text{we} : V(G) \rightarrow \mathbb{Z}_{\geq 1}\) is a weight function, is a weighted framework. Now we can say that an \((S, T)\)-linkage \(P\) in a weighted framework \((G, M, \text{we})\) is \((k, w)\)-ranked if \(V(P)\) contains a set \(X\) of vertices with \(X \subseteq T\), size \(|X| = k\), and weight \(\text{we}(X) = w\). By using the same reduction as from Theorem 1 to Theorem 3, we obtain the following theorem.

**Theorem 5.** There is a randomized algorithm that given a weighted framework \((G, M, \text{we})\), where \(G\) is an \(n\)-vertex graph and \(M\) is represented as a matrix over a finite field of order \(q\), sets of vertices \(S, T \subseteq V(G)\), and integers \(p, k, w\), in time \(2^p \cdot O(k^3 \log(q+k)) n^{O(1)} w\) either finds a \((k, w)\)-ranked \((S, T)\)-linkage of order \(p\) and of minimum total length, or determines that \((G, M, \text{we})\) has no \((k, w)\)-ranked \((S, T)\)-linkages of order \(p\).

Note that Theorem 5 implies Theorem 3 by setting all vertex weights to 1 and \(w = k\).

Finally, we remark that even though the correctness argument of our algorithm is technical, the algorithm itself is simple and practical, consisting of only simple dynamic programming over walks in the graph. In particular, the observed practicality of the algorithm of Björklund, Husfeldt, and Taslaman [4] for \(T\)-Cycle on graphs with thousands of vertices holds also for our algorithm.

**Organization of the paper.** The rest of the paper is organized as follows. In Section 2 we overview our techniques and outline our algorithms. In Section 3 we recall definitions and preliminary results. In Section 4 we prove the main result, i.e., Theorem 4 (recall that Theorem 4 implies Theorem 1). In Section 5 we give the extensions of our results from colored graphs to frameworks, i.e., Theorem 5. In Section 6 we prove Theorem 2. Finally, we conclude in Section 7.

## 2 Techniques and outline

The techniques behind Theorem 1 build on the idea of exploiting cancellation of monomials, a fundamental tool in the area [2, 3, 4, 5, 29, 30, 33, 42, 46]. In particular, we build on the cycle-reversal-based cancellation for \(T\)-Cycle introduced by Björklund, Husfeldt, and Taslaman [4], and on the bijective labeling-based cancellation introduced by Björklund [2] (see also [3]). The algorithm of Theorem 2 builds on color-coding [1], generalizing ideas that appeared in [19] for finding an \((s, t)\)-cycle of length at least \(k\).

In Section 2.1 we explain the new ideas of the techniques behind Theorem 1 in comparison to the earlier works, in Section 2.2 we give a more detailed outline of the proof of Theorem 1, and in Section 2.3 we give an outline of the proof of Theorem 2.

### 2.1 New techniques for Theorem 1

Let us first focus on the single path case of Theorem 1, i.e., \(p = |S| = |T| = 1\), corresponding to the question of finding a \(k\)-colored \((s, t)\)-path. Our algorithm is analogous to the algorithm of Björklund, Husfeldt, and Taslaman [4] for \(T\)-Cycle, but instead of having the “interesting set” of vertices \(T\) fixed in advance, our algorithm can choose any interesting set \(X \subseteq V(G)\) of vertices of size \(|X| = k\) included in the path “on the fly” in the dynamic programming over the walks. In particular, our dynamic programming over walks can choose whether it gives a label to a vertex or not. This is the crucial difference to the earlier works where there would be a set of vertices \(Y \subseteq V(G)\) fixed in advance so that a vertex of \(Y\) would always be given a label if encountered in the walk and the vertices \(V(G) \setminus Y\) would never be given labels [2, 3, 4, 42]. This would impose a limitation that because these algorithms work in time exponential in the number of labels used
(i.e. \(2^k\), where \(k\) is the number of labels), the intersection of the found path with the set \(Y\) would have to be bounded in the parameter. This explains why the previous techniques could not yield an FPT-algorithm for \textsc{Maximum Colored Path}, as no such suitable set \(Y\) can be fixed in advance.

Our on the fly labeling of vertices allows our algorithm to find paths that visit the same color multiple times, while still making sure that at least \(k\) different colors are visited. In particular, the interesting set \(X \subseteq V(G)\) of vertices in the path that we want to label is any set of size \(k\) that contains \(k\) different colors. While our dynamic programming is still a straightforward dynamic programming over walks, the main difficulty over previous works is the argument that if no solution exists, then the polynomial that we compute is zero, i.e., all unwanted walks cancel out.

First, the argument of cancellation in the case when two vertices of the same color are given a label is a now-standard application of the bijective labeling based cancellation of Björklund [2]. Therefore, our main focus is on a cancellation argument for walks where \(k\) vertices of different colors have been labeled. Here, our starting point is the cycle reversal based cancellation argument for \(T\)-\textsc{Cycle} [4], but in our case significantly more arguments are needed. In particular, the main difference to earlier works caused by the introduction of the on the fly labeling is that a vertex can occur in a walk as both labeled and unlabeled. Very much oversimplified, this case is handled by a new label-swap cancellation argument, where a label is moved from a labeled occurrence of a vertex into an unlabeled occurrence of the vertex. While in isolation this argument is simple, it causes significant complications when combining with the cycle reversal based cancellation, in particular because of the “no labeled digons” property we have to impose to the labeled walk. However, we manage to combine these two arguments into a one very technical cancellation argument.

Then, let us move from one \((s,t)\)-path to an \((S,T)\)-linkage. This generalization of using cancellation of monomials to find multiple paths is foreshadowed by an algorithm for minimum cost flow by Lingas and Persson [33]. However, their arguments are considerably simpler due to not having labels on the walks.

To find \((S,T)\)-linkages, we use a similar dynamic programming to the one path case, extending the set of walks from \(S\) to \(T\) one walk at a time. Here, we must introduce a new cancellation argument for the case when two different walks intersect. This argument is again simple in isolation: take the intersection point of the two intersecting walks and swap the suffixes of them starting from this point. First, to make sure that this operation does anything we need to make sure that the suffixes are not equal. We do this by enforcing that the ending vertices of the walks are different already in the dynamic programming, which adds the extra \(2^p\) factor to the time complexity. The second complication is that again, this suffix swap operation does not play well together with the other cancellation arguments, and we need to again significantly increase the complexity of the combination of the three cancellation arguments. In the end, we have to consider 18 different cases in our cancellation argument, see Definition 3.

The extension from Theorem 1 to the weighted setting of Theorem 4 is a simple modification of the dynamic programming so that also the weight of the labeled vertices \(X\) is stored. Interestingly, this argument could be extended to look for paths containing a set of vertices \(X\) with any property of \(X\) that could be efficiently evaluated in dynamic programming.

2.2 Outline of Theorem 1

We first give the outline of the algorithm for the single path case of finding a \(k\)-colored \((s,t)\)-path, and then discuss the generalization to \((S,T)\)-linkage.

Superficially, our approach follows the one of Björklund, Husfeldt, and Taslaman [4] developed for the \(T\)-\textsc{cycle} problem. Similar to Björklund et al., for every length \(\ell \geq 1\), we define a certain family of walks \(C_\ell\) and a polynomial \(f(C_\ell)\) so that over a finite field of characteristic 2, the polynomial
\( f(G) \) is non-zero if the graph contains a \( k \)-colored \((s, t)\)-path of length \( \ell \), and the polynomial \( f(G) \) is the identically zero polynomial if the graph does not contain any \( k \)-colored \((s, t)\)-path of length \( \leq \ell \). Then by making use of the DeMillo-Lipton-Schwartz-Zippel lemma [44, 50], finding a \( k \)-colored \((s, t)\)-path of minimum length boils down to evaluating the polynomial \( f(G) \) at a random point for increasing values of \( \ell \).

With \( k \)-colored path, the role similar to the role of terminal vertices in \( T \)-cycle is played by a subset \( X \) of \( k \) vertices of the path with \( k \) different colors. However, a priori we do not know this set \( X \), and there could be \( n^k \) candidates so we cannot enumerate them. Because of that, we define the polynomial \( f \) on families of labeled \((s, t)\)-walks in the graph \( G \). A labeled \((s, t)\)-walk of length \( \ell \) is a pair of sequences \( W = ( (v_1, \ldots, v_\ell), (r_1, \ldots, r_\ell)) \), where \( v_1, \ldots, v_\ell \) is an \((s, t)\)-walk of length \( \ell \), and \( r_1, \ldots, r_\ell \) is a sequence of numbers from \([0, k]\) indicating a labeling. The interpretation of the labeling is that \( r_i = 0 \) indicates that the index \( i \) of the walk is not labeled, and \( r_i \geq 1 \) indicates that the index \( i \) is labeled with the label \( r_i \), with the interpretation that the vertex \( v_i \) at this index is selected to the set \( X \).

Next we present the definition of the polynomial \( f \). The polynomial \( f \) is over \( \text{GF}(2^{3+[\log_2 n]}) \), which is a field of characteristic 2 and order \( \geq 8n \). With every edge \( uv \in E(G) \) we associate a variable \( f_e(uv) \), with every vertex \( v \in V(G) \) we associate a variable \( f_v(v) \), and with every color-label pair \((x, y) \in [n] \times [k] \) we associate a variable \( f_c(x, y) \). For a labeled walk \( W = ( (v_1, \ldots, v_\ell), (r_1, \ldots, r_\ell)) \) we associate the monomial

\[
    f(W) = \prod_{i=1}^{\ell-1} f_e(v_i v_{i+1}) \cdot \prod_{i \in [\ell]|r_i \neq 0} f_v(v_i) \cdot f_c(c(v_i), r_i).
\]

For the family of walks \( C_\ell \), which we will define immediately, we are interested in the polynomial

\[
    f(\mathcal{C}_\ell) = \sum_{W \in \mathcal{C}_\ell} f(W).
\]

For vertices \( s, t \) and integers \( k, \ell \), the family \( \mathcal{C}_\ell \) is the family of all labeled \((s, t)\)-walks

\[
    W = ( (v_1 = s, v_2, \ldots, v_\ell = t), (r_1, \ldots, r_\ell))
\]

of length \( \ell \) that satisfy the following two properties. The first property is that the labeling \((r_1, \ldots, r_\ell) \) is bijective, meaning that every label from \([k]\) is used exactly once. Note that this implies that every monomial of \( f(\mathcal{C}_\ell) \) has degree \( \ell - 1 + 2k \), being a product of \( \ell - 1 \) edge variables, \( k \) vertex variables, and \( k \) color-label pair variables. The second property is that the labeled walk \( W \) has no labeled digons. By that we mean that \( W \) cannot have a subwalk \( v_{i-1} v_i v_{i+1} \) with \( v_i \) being a labeled vertex (with \( r_i \neq 0 \)) and \( v_{i-1} = v_{i+1} \). It is not immediately clear that having no labeled digons is useful, but this will turn out to be crucial similarly to the property of having no \( T \)-digons in the algorithm for \( T \)-cycle [4].

It is not difficult to prove that when a graph has a \( k \)-colored \((s, t)\)-path of length \( \ell \), then \( f(\mathcal{C}_\ell) \) is a non-zero polynomial. Indeed, a path has no repeated vertices and thus has no labeled digons, so if we take a \( k \)-colored \((s, t)\)-path \( v_1, \ldots, v_\ell \) and let the labels \( r_1, \ldots, r_\ell \) take the values from \([k]\) on \( k \) vertices with \( k \) different colors, then the labeled walk \( W = ( (v_1, \ldots, v_\ell), (r_1, \ldots, r_\ell)) \) appears in \( \mathcal{C}_\ell \), and thus a corresponding monomial \( f(W) \) appears in \( f(\mathcal{C}_\ell) \). Because \( v_1, \ldots, v_\ell \) is a path and the labeled vertices have different colors, we can recover the labeled walk \( W \) uniquely from the monomial \( f(W) \), and therefore the monomial \( f(W) \) must occur exactly once in the polynomial \( f(\mathcal{C}_\ell) \) (i.e. with coefficient 1), and therefore \( f(\mathcal{C}_\ell) \) is non-zero.

The proof of the opposite statement—absence of a \( k \)-colored \((s, t)\)-path of length \( \leq \ell \) implies that \( f(\mathcal{C}_\ell) \) is zero—is more complicated. We have to show that in this case each monomial \( f(W) \) for
labeled walks \( W \in \mathcal{C}_\ell \) occurs an even number of times in the polynomial \( f(\mathcal{C}_\ell) \), in particular that there is an even number of labeled walks \( W \in \mathcal{C}_\ell \) for every monomial \( f(W) \). The proof is based on constructing an \( f \)-invariant fixed-point-free involution \( \phi \) on \( \mathcal{C}_\ell \), that is a function \( \phi : \mathcal{C}_\ell \to \mathcal{C}_\ell \) such that for every \( W \in \mathcal{C}_\ell \) it holds that \( f(W) = f(\phi(W)), \phi(W) \neq W, \) and \( \phi(\phi(W)) = W \).

Let us start with the easy part of the proof, that is, constructing such \( \phi \) for labeled walks where two vertices of the same color are labeled (which could be two different occurrences of the same vertex). In this case, let \( 1 \leq i < j \leq \ell \) be the lexicographically smallest pair of indices so that \( c(v_i) = c(v_j), r_i \neq 0, \) and \( r_j \neq 0. \) The function \( \phi \) works by swapping \( r_i \) with \( r_j \). Because each label from \( [k] \) occurs in \( r_1, \ldots, r_\ell \) exactly once, this results in a different labeled walk \( \phi(W) \) with the same monomial \( f(\phi(W)) = f(W), \) and moreover \( W = \phi(\phi(W)) \) holds. After this argument, we can let \( \mathcal{C}_\ell^* \subseteq \mathcal{C}_\ell \) be the family of labeled walks in \( \mathcal{C}_\ell \) where all labeled vertices have different colors, and we know that \( f(\mathcal{C}_\ell^*) = f(\mathcal{C}_\ell) \). Therefore, we can focus only to constructing \( \phi : \mathcal{C}_\ell^* \to \mathcal{C}_\ell^* \).

Now, the first approach (which does not work) would be to adapt the strategy of Björklund, Husfeldt, and Taslaman for our purposes. The essence of their strategy is the following. Since \( \phi(W) \) is that a labeled vertex could also occur several times in a walk as unlabeled. Because of that, reversing a subwalk can result in a walk with a labeled digon, and thus \( \phi \) could map \( W \) outside the family \( \mathcal{C}_\ell^* \). For example, for a walk \( abcd\hat{a}b \) (here \( \hat{a} \) is a labeled vertex), reversing \( \hat{a}bcd\hat{a}b \) results in walk \( adcb\hat{a}b \) with labeled digon \( b\hat{a}b \). A natural “patch” for that type of walks is to not reverse but to apply a new type of operation of swapping a label from one occurrence of a vertex to another occurrence of it. For example, swapping a label for \( abcd\hat{a}b \) would result in \( \hat{a}bcdab \). This results in a different labeled walk contributing the same monomial \( f(W) \) to the polynomial. See Figure 1 for an illustration of the above examples.

![Figure 1](image_url)  
Figure 1: An illustration of the walks \( abcd\hat{a}b, \hat{a}bcd\hat{a}b = adcb\hat{a}b, \) and \( \hat{a}bcdab. \) The grey bags correspond to vertices of the graph. The squares are copies of the corresponding bag-vertex and these together with the red path illustrate the order the vertices appear in the walk. Red squares correspond to labeled vertices.

However, the new operation of swapping a label brings us new problems. First of all, swapping a label could again result in a labeled digon. For example, swapping a label for walk \( \hat{a}bc\hat{a}c \) results in walk \( ab\hat{b}c\hat{a}c \) with labeled digon \( c\hat{a}c. \) An attempt to “patch” this by using a “mixed” strategy—when possible, swap a label, otherwise reverse—does not work either. For example, for walk \( W = \hat{a}bc\hat{a}c \) we cannot label swap (that will result in a labeled digon \( c\hat{a}c), \) hence we reverse. Thus we obtain walk \( W' = \phi(W) = \hat{a}bc\hat{a}c = \hat{a}bc\hat{a}c. \) For \( W', \) swapping a label for \( a \) is a valid operation, thus \( \phi(W') = abc\hat{a}c, \) but then we would have that \( \phi(\phi(W)) \neq W. \) See Figure 2 for an illustration of the above example.

At this moment, the situation becomes desperate: the more patches we introduce, the more
issues appear and the whole construction falls apart. Moreover, on top of that, one has to define
the mapping \( \phi \) recursively in order to deal with palindromic loops, making the situation even more
complicated. We find it a bit surprising, that in the end a combination of label swaps and reverses
allows the construction of the required mapping \( \phi \). To make it happen, we use quite an involved
strategy to identify what labels can be swapped and what subwalks could be reversed, and in
what order. Whole Section 4.4 is devoted to defining this strategy (for the more general setting of
\((S,T)\)-linkages) and to the proof of its correctness.

To evaluate the polynomial \( f(C_\ell) \), we apply quite standard dynamic programming techniques.
In particular, the polynomial can be evaluated in \( 2^k \cdot n^{O(1)} \) time by dynamic programming over
walks, where we store the length of the walk, the last two vertices of the walk, the subset of labels
used so far (causing the \( 2^k \) factor), and whether the last vertex is labeled. This is similar to the
dynamic programming for \( T\text{-cycle} \) [4], with the difference only in that it is chosen in the dynamic
programming which vertices of the walk are labeled, and that instead of a subset of \( T \) we store the
subset of the labels.

To extend the algorithm from a single \((s,t)\)-path to an \((S,T)\)-linkage of order \( p \), we define a
family \( C_\ell \) of labeled walkages and a polynomial \( f(C_\ell) \) over them. We note that by a simple reduction
we can assume that \(|S| = |T| = p\), and that \( S \) and \( T \) are disjoint. A labeled walkage of order \( p \) and
total length \( \ell \) is a \( p \)-tuple \( W = (W^1, \ldots, W^p) \) of labeled walks \( W^i \), whose sum of the lengths is \( \ell \).
The family \( C_\ell \) contains labeled walkages \( W \) with the following properties: They have order \( p \), total
length \( \ell \), the starting vertices are ordered according to a total order on \( V(G) \), ending vertices are
distinct (each vertex in \( T \) is an ending vertex of exactly one walk in \( W \)), the labeling is bijective
(each label from \([k]\) is used exactly once), and no walk in \( W \) contains a labeled digon.

The monomial \( f(W) \) is then defined as
\[
f(W) = \prod_{i=1}^{p} f(W^i),
\]
and the polynomial \( f(C_\ell) \) as
\[
f(C_\ell) = \sum_{W \in C_\ell} f(W).
\]

The definitions are analogous to the single path case, in particular we recover the previously
explained single path case by setting \( p = 1 \). The proof that if there exists a \( k \)-colored \((S,T)\)-linkage
of order \( p \) and total length \( \ell \) then \( f(C_\ell) \) is non-zero is directly analogous to the one path case. Also
the proof that we can consider the smaller family \( C_\ell^* \subseteq C_\ell \) where all labeled vertices have different
colors is analogous.
However, to prove that if there is no $k$-colored $(S,T)$-linkages of order $p$ and total length $\leq \ell$ then $f(C_p^\ell)$ is identically zero we need new cancellation arguments beyond the previous cycle reversal and label swap arguments. In particular, none of the previously considered arguments can be applied if we have a labeled walkage $W = (W^1, W^2)$ of order two, where both $W^1$ and $W^2$ are labeled paths that intersect. In this case, the new argument is that we could swap the suffixes of $W^1$ and $W^2$ starting from the intersection point. For example, for a walkage $W = (abcde, stcuv)$, we define $\phi(W) = (abcuv, stcde)$. The property that the walks in $W$ have different ending vertices is crucial here to ensure that $\phi(W) \neq W$.

However, also with this suffix swap cancellation argument we run into problems. In particular, the first challenge is that the suffix swap could create labeled digons, for example when $W = (ab\hat{c}de, stcbu)$ both of the walks are paths, but swapping the suffix after $c$ would create a labeled digon. In this situation we can instead use the label swap operation on $c$, from the first walk to the second, but of course this will add again even more complications. In the end, we manage to extend the strategy of $\phi$ from paths to linkages, but it makes the definition of $\phi$ even more complicated (see Definition 3, the path case uses the case groups A and C, while the linkage case needs the addition of case groups B and D).

The dynamic programming for $(S,T)$-linkage is similar to the $(s,t)$-path, extending the walks in the walkage one walk at the time. It requires two new fields to store, the index of the walk that we are currently extending, and the subset of the ending vertices $T$ that have been already used. Storing the used ending vertices causes the additional factor $2^p$ in the time complexity (as we can assume that $|T| = p$).

### 2.3 Outline of Theorem 2

Recall that the main difference to Theorem 1 is that Theorem 2 provides a deterministic algorithm that, moreover, works on directed graphs. The price is, however, that this algorithm is only suitable for the special case of finding an $(S,T)$-linkage of length at least $k$, and the time complexity as a function of $k$ and $p$ is higher. Theorem 2 thus requires a completely different toolbox: the algorithm is based on ideas of random separation. Our result can be seen as a generalization of earlier works on finding paths and cycles of length at least $k$, the closest one being the result of Fomin et al. [19] on finding an $(s,t)$-cycle of length at least $k$. Note that their result is stated for undirected graphs, and that the problem of finding an $(S,T)$-linkage of order 2 and length at least $k$ is equivalent to the problem of finding an $(s,t)$-cycle of length at least $k$ (up to increasing $k$ by 2) on undirected graphs.

Similarly to the earlier results, the case where the target $(S,T)$-linkage is of length close to $k$ can be covered by a standard application of color-coding [1]. The difficulty is that the length of the $(S,T)$-linkage can be arbitrarily larger than $k$. While because of that it would be intractable to highlight the target $(S,T)$-linkage as a whole, it is still possible to apply random separation to give distinct colors to $k$-length segments at the end of each path in the $(S,T)$-linkage. The main hurdle is then to argue that at least in one color we can pick a finishing segment as an arbitrary shortest path of length $k$, without intersecting any other path in the optimal solution. Afterwards, finding the desired $(S,T)$-linkage is easy, as the length requirement is already satisfied; one only needs to find a suitable connection to complete the $(S,T)$-linkage, which exists as witnessed by the optimal solution.

Lemma 15 encapsulates the novel combinatorial result allowing the approach above, strongly generalizing a similar basic idea that appeared in [19] for two undirected paths. To give an intuition behind the lemma (see also Figure 3), observe first that the problem of finding an $(S,T)$-linkage of order $p$ and of total length at least $k$ is equivalent to the problem of finding an $(s,t)$-linkage of order
p and of total length at least \( k + 2 \), where an \((s, t)\)-linkage of order \( p \) consists of \( p \) internally-disjoint \((s, t)\)-paths, for some \( s, t \in V(G) \). Now let \((s, t)\)-paths \( P_1, \ldots, P_q \) come from the shortest solution, an \((s, t)\)-linkage of order \( p \) and the smallest total length which is at least \( k \). Let the sets \( C_1, \ldots, C_q \) be the result of random separation applied to \( k \)-length suffixes of \( P_1 - \{t\}, \ldots, P_q - \{t\} \), i.e., for each \( i \in [q] \), the \( k \)-length suffix of \( P_i - \{t\} \) is contained in \( C_i \). The algorithm of Theorem 2 seeks to find a solution where for some \( i \in [q] \), \( v_i \) is the \( k \)-th vertex of \( P_i \) from \( t \), and \( Q_i \) is a \( k \)-length shortest path from \( v_i \) to \( t \) inside \( C_i \), by guessing \( v_i \in C_i \) and taking an arbitrary path \( Q_i \) of the

![Figure 3: Illustration to the proof of Lemma 16. Large empty squares mark tokens in the current state; those with red filling are moved by the next rule.](image-url)
form above. The solution is then any collection of an \((s, v_i)\)-path and \(p - 1\) \((s, t)\)-paths that do not intersect each other and \(Q_i\), together with \(Q_i\). If \(Q_i\) does not intersect the \((s, v_j)\)-prefix of \(P_j\), for each \(j \in [q]\), then the paths \(P_1, \ldots, P_q\) certify that the desired collection of paths exists. Now comes Lemma 15: it claims, roughly, that if this is not the case for all \(i \in [q]\), then there is a shorter \((s, t)\)-linkage given by prefixes of \(P_1, \ldots, P_q\) and suffixes of \(Q_1, \ldots, Q_q\) (introducing another color to the random separation makes sure that the total length of the prefixes is still at least \(k\)), which is a contradiction.

The proof of Lemma 15 can be imagined as the following token sliding game. First, we put a token on each \(P_i\), at the first place of intersection with some \(Q_j\). Then we move the tokens by applying two rules, \textbf{Push} and \textbf{Clear}. If two tokens end up on the same \(Q_j\) for some \(j \in [q]\), we move the farthest of them from \(t\) further along its path \(P_i\), until it hits another \(Q_{j'}\); this is called \textbf{Push}. As for the \textbf{Clear}, if at any step \(h\) the current token \(t_i^h\) of the path \(P_i\) reaches the vertex \(v_i\), we forfeit this path: the token is moved to \(t\), which corresponds to the \(i\)-th path of the shorter solution being exactly \(P_i\), and all other tokens on \(Q_i\) are moved next along their paths similarly to the rule \textbf{Push}. Moreover, every future application of any rule will not place a token on \(Q_i\), skipping it to the next \(Q_j\) that is still active. Clearly, this game is finite, as tokens are only being slid further along their paths. The main claim of Lemma 15 is that when the game is over, there is at least one remaining active token, these tokens are one per a path in \(\{Q_j\}_{j \in [q]}\) (since \textbf{Push} is not applicable), and that all corresponding paths \(P_i\) can be simultaneously extended each along its own \(Q_j\) instead of taking their original routes, without intersections (since \textbf{Push} we always keep the closest token to \(t\)). This is a shorter solution since a token of \(P_i\), if active, is inside some \(Q_j\) at distance less than \(k\) from \(t\), and the prefix of \(P_i\) up to this token is shorter than the prefix of \(P_i\) up to \(v_i\).

Another challenge the proof of Theorem 2 faces, is that while the random separation approach is well-known, it is normally applied to separating two, rarely three (e.g. [19]), sets. We, on the other hand, need to apply random separation to \(p\) sets simultaneously, while making sure that it can be derandomized. To this end, in Lemma 16 we devise in a deterministic way a family of functions that models random separation of \(p\) sets of size at most \(k\) each. The size of this family is bounded by \(p^{O(kp)} \log n\), which matches the inverse probability (up to the \(\log n\) factor) of coloring the universe in \(p\) colors uniformly at random so that each set receives its own color. The construction is based on perfect hash families [39].

3 Preliminaries

In this section, we introduce basic notation and state some auxiliary results.

3.1 Basic definitions and preliminary results

We use \(\mathbb{Z}_{\geq 1}\) to denote the set of positive integers and \(\mathbb{Z}_{\geq 0}\) the set of non-negative integers. Also, given integers \(p, q\) such that \(p < q\), we use \([p, q]\) to denote the set \(\{p, p + 1, \ldots, q\}\) and, if \(p \geq 1\), we use \([p]\) to denote the set \(\{1, \ldots, p\}\).

Parameterized Complexity. We refer to the book of Cygan et al. [14] for introduction to the area. Here we only briefly mention the notions that are most important to state our results. A parameterized problem is a language \(L \subseteq \Sigma^* \times \mathbb{N}\), where \(\Sigma^*\) is a set of strings over a finite alphabet \(\Sigma\). An input of a parameterized problem is a pair \((x, k)\), where \(x\) is a string over \(\Sigma\) and \(k \in \mathbb{N}\) is a parameter. A parameterized problem is fixed-parameter tractable (or FPT) if it can be solved
in time \( f(k) \cdot |x|^{O(1)} \) for some computable function \( f \). The complexity class FPT contains all fixed-parameter tractable parameterized problems.

**Graphs.** We use standard graph-theoretic terminology and refer to the textbook of Diestel [17] for missing notions. We consider only finite graphs, and the considered graphs are assumed to be undirected if it is not explicitly said to be otherwise. For a graph \( G \), \( V(G) \) and \( E(G) \) are used to denote its vertex and edge sets, respectively. Throughout the paper we use \( n = |V(G)| = |G| \) and \( m = |E(G)| \) if this does not create confusion. For a graph \( G \) and a subset \( X \subseteq V(G) \) of vertices, we write \( G[X] \) to denote the subgraph of \( G \) induced by \( X \). For a vertex \( v \), we denote by \( N_G(v) \) the (open) neighborhood of \( v \), i.e., the set of vertices that are adjacent to \( v \) in \( G \). For \( X \subseteq V(G) \), \( N_G(X) = \left( \bigcup_{v \in X} N_G(v) \right) \setminus X \). The degree of a vertex \( v \) is \( d_G(v) = |N_G(v)| \). If \( G \) is a digraph, \( N^+_G(v) \) denotes the out-neighborhood of \( v \), i.e., the set of vertices that are adjacent to \( v \) in \( G \) via an arc from \( v \), and \( N^-_G(v) \) is the in-neighborhood, defined symmetrically for arcs going to \( v \). We may omit subscripts if the considered graph is clear from a context.

A walk \( W \) of length \( \ell \) in \( G \) is a sequence of vertices \( v_1, v_2, \ldots, v_\ell \), where \( v_i v_{i+1} \in E(G) \) for all \( 1 \leq i < \ell \). The vertices \( v_1 \) and \( v_\ell \) are the endpoints of \( W \) and the vertices \( v_2, \ldots, v_{\ell-1} \) are the internal vertices of \( W \). A path is a walk where no vertex is repeated. For a path \( P \) with endpoints \( s \) and \( t \), we say that \( P \) is an \((s,t)\)-path. A cycle is a path with the additional property that \( v_1 v_\ell \in E(G) \) and \( \ell \geq 3 \).

**DeMillo-Lipton-Schwartz-Zippel lemma.** Our strategy involves the use of the DeMillo-Lipton-Schwartz-Zippel lemma for randomized polynomial identity testing.

**Lemma 1 ([44, 50]).** Let \( p(x_1, \ldots, x_n) \) be a non-zero polynomial of total degree \( d \) over a field \( \mathbb{F} \), and let \( S \) be a subset of \( \mathbb{F} \). If each \( x_i \) is independently assigned an uniformly random value from \( S \), then \( p(x_1, \ldots, x_n) = 0 \) with probability at most \( d/|S| \).

### 3.2 Hardness results

We conclude this section by showing the NP-hardness of finding a \( k \)-colored \((s,t)\)-path on directed graphs, for any \( k \geq 2 \), and the optimality of the time complexity of Theorem 1 assuming the Set Cover Conjecture (SeCoCo) of Cygan et al. [13].

We start with the hardness for directed graphs.

**Proposition 1.** For any integers \( k, \ell \geq 2 \), it is NP-complete to decide, given a directed graph \( G \), a coloring \( c : V(G) \to \ell \), and two vertices \( s \) and \( t \), whether \( G \) has a \( k \)-colored \((s,t)\)-path.

**Proof.** We show the claim for \( k = \ell = 2 \) as it is straightforward to generalize the proof for other values of \( k \) and \( \ell \). We reduce from the DISJOINT PATHS problem on directed graphs. The task of this problem is, given a (directed) graph \( G \) and \( k \) pairs of terminal vertices \( (s_i, t_i) \) for \( i \in \{1, \ldots, k\} \), decide whether \( G \) has vertex-disjoint \((s_i, t_i)\)-paths for \( i \in \{1, \ldots, k\} \). This problem is well-known to be NP-complete on directed graphs even if \( k = 2 \) [23]. Consider an instance \((G, (s_1, t_1), (s_2, t_2))\) of DISJOINT PATHS, where \( G \) is a directed graph. We assume that the terminal vertices are pairwise distinct. We construct the directed graph \( G' \) from \( G \) by adding a vertex \( w \) and arcs \((t_1, w)\) and \((w, s_2)\). Note that \( G \) has vertex-disjoint \((s_1, t_1)\) and \((s_2, t_2)\)-paths if and only if \( G' \) has an \((s_1, t_2)\)-path containing \( w \). We define the coloring \( c \) by setting \( c(w) = 1 \) and defining \( c(v) = 2 \) for all \( v \in V(G') \setminus \{w\} \). Clearly, \( G' \) has a 2-colored \((s_1, t_2)\)-path if and only if \( G' \) has an \((s_1, t_2)\)-path containing \( w \). This immediately implies NP-hardness.

\(\square\)

Then, we show that Theorem 1 is optimal assuming the Set Cover Conjecture.
Proposition 2. If there is a \((2 - \varepsilon)^{k_1 n}O(1)\) time algorithm for finding a \(k\)-colored path in a \(k\)-colored graph for some \(\varepsilon > 0\), then there is a \((2 - \varepsilon)^n (mn)O(1)\) time algorithm for Set Cover.

Proof. In the Set Cover problem, we are given a universe \(U\) of \(n\) elements, a collection \(S\) of \(m\) subsets of \(U\), and an integer \(t\) and we ask whether there is a collection \(S' \subseteq S\) of size \(t\) such that for every \(u \in U\), there is a set \(S \in S'\) such that \(u \in S\).

Given an instance \((U, S, t)\) of Set Cover, where \(|U| = n\) and \(S = \{S_1, \ldots, S_m\}\), we construct a graph \(G\) as follows. We first construct the graph \(H\) by considering two vertices \(a\) and \(b\) and adding \(m\) internally vertex-disjoint \((a, b)\)-paths \(P_{S_1}, \ldots, P_{S_m}\), where for every \(i \in [m]\), the vertices in \(P_{S_i}\) are bijectively mapped to the elements of \(S_i\). We call \(a\) the source of \(H\) and \(b\) the sink of \(H\). We finally construct a graph \(G\) that is obtained by considering \(t\) copies \(H_1, \ldots, H_t\) of \(H\), for each \(i \in [t-1]\), identifying the sink \(b_i\) of \(H_i\) with the source \(a_{i+1}\) of \(H_{i+1}\), and adding two new vertices \(v\) and \(v'\) of degree one, adjacent to \(a_1\) and \(b_t\) respectively. See Figure 4 for an illustration of the construction of graph \(G\). Note that \(t \leq m\) and \(|V(G)| = (mn)O(1)\).

Assuming an ordering \(u_1, \ldots, u_n\) of \(U\), for each \(i \in [n]\), we assign color \(i\) to all vertices of \(G\) that correspond to \(u_i\), color \(n + 1\) and \(n + 2\) to \(v\) and \(v'\), and color \(n + 3\) to all vertices in \(V(G) \setminus \{v, v'\}\) that do not correspond to members of \(U\). Observe that \((U, S, t)\) is a yes-instance of Set Cover if and only if there is an \(n + 3\)-colored path in \(G\). Therefore, a \((2 - \varepsilon)^{k_1 n}O(1)\) time algorithm for finding a \(k\)-colored path in a \(k\)-colored \(n\)-vertex graph implies the existence of a \((2 - \varepsilon)^n (mn)O(1)\) time algorithm for finding a set cover of size \(t\) in a universe \(U\) of size \(n\) with a collection \(S\) of \(m\) subsets of \(U\).

\[\square\]

4 Randomized algorithm for colored \((S, T)\)-linkages

In this section we prove the main result, i.e., Theorem 4. Recall that Theorem 1 is a special case of Theorem 4.

Let \(G\) be an \(n\)-vertex graph, \(p\) an integer, and \(S, T \subseteq V(G)\). An \((S, T)\)-linkage of order \(p\) is a set \(P\) of \(p = |P|\) vertex-disjoint paths between \(S\) and \(T\). We denote by \(V(P)\) the vertices in the paths of \(P\). The length of an \((S, T)\)-linkage is the total number \(|V(P)|\) of vertices in the paths. Let \(c : V(G) \to [n]\) an arbitrary coloring of \(G\), and \(\mathbf{w} : V(G) \to \mathbb{Z}_{\geq 1}\) a weight function. For positive integers \(k\) and \(w\), we say that an \((S, T)\)-linkage \(P\) is \((k, w)\)-colored if there exists a set \(X \subseteq V(P)\) with \(|X| = k\), all vertices of \(X\) have different colors, and \(\mathbf{w}(X) = \sum_{v \in X} \mathbf{w}(v) = w\). We give a \(2^{p+k_n}O(1)\)-time algorithm for the problem of finding a minimum length \((k, w)\)-colored \((S, T)\)-linkage of order \(p\) (Theorem 4).

We will assume that \(|S| = |T| = p\), and \(S\) and \(T\) are disjoint, as the general case can be reduced to this case by the following reduction: We add \(p\) vertices \(s_1, \ldots, s_p\) with \(N(s_i) = S\) and \(p\) vertices \(t_1, \ldots, t_p\) with \(N(t_i) = T\), all with the same new color and weight equal to \(k \cdot \max_{v \in V(G)} \mathbf{w}(v) + 1\). Then, we can set \(S = \{s_1, \ldots, s_p\}\) and \(T = \{t_1, \ldots, t_p\}\), and solve the problem with \(k\) increased by one and \(w\) increased by \(k \cdot \max_{v \in V(G)} \mathbf{w}(v) + 1\). Because we can assume that the original weights
are at most \( w + 1 \), this increases the target weight \( w \) by a factor \( O(k) \), and therefore does not increase the time complexity of the algorithm.

### 4.1 Labeled walks and walkages

In this subsection we define labeled walks and labeled walkages.

**Labeled walks.** Let \( \ell \) be an integer. A walk of length \( \ell \) in \( G \) is a sequence of vertices \( v_1, \ldots, v_\ell \) of \( G \), where \( v_iv_{i+1} \in E(G) \) for all \( 1 \leq i < \ell \). A labeled walk of length \( \ell \) is a pair of sequences \( W = ((v_1, v_2, \ldots, v_\ell), (r_1, r_2, \ldots, r_\ell)) \), where \( v_1, \ldots, v_\ell \) is a walk of length \( \ell \), and \( r_1, \ldots, r_\ell \) is a sequence of integers from \([0,k]\), indicating a labeling. The interpretation of the labeling is that \( r_i = 0 \) indicates that the index \( i \) is unlabeled and \( r_i \neq 0 \) indicates that the index \( i \) is labeled with the label \( r_i \in [k] \). A labeled walk is injective if each label from \([k]\) appears in it at most once. Most of the labeled walks that we treat in the algorithm have length at least one, but the definition allows also an empty labeled walk of length zero. The set of vertices collected by \( W \) is \( R(W) = \{v_i \mid r_i \neq 0\} \), i.e., the set of vertices that occur at labeled indices. The set of edges of \( W \) is \( E(W) = \{v_iv_{i+1} \mid 1 \leq i < \ell\} \). An index \( i \) in a labeled walk of length \( \ell \) is a digon if \( 1 < i < \ell \) and \( v_{i-1} = v_{i+1} \) (see Figure 5 for an illustration). An index \( i \) in a labeled walk is a labeled digon if it is a digon and \( r_i \neq 0 \).

![Figure 5: An example of a labeled walk \( W \) with a digon \( i \).](image)

**Labeled walkages.** A labeled walkage of order \( p \) is a tuple \( W = (W^1, \ldots, W^p) \), where each \( W^i = ((v^i_1, \ldots, v^i_\ell), (r^i_1, \ldots, r^i_\ell)) \) is a labeled walk of length \( \ell_i \geq 1 \). The length of \( W \) is \( \sum_{i=1}^{p} \ell_i \). The set of edges of \( W \) is \( E(W) = \bigcup_{i=1}^{p} E(W^i) \). The set of vertices collected by \( W \) is \( R(W) = \bigcup_{i=1}^{p} R(W^i) \). The weight \( \text{we}(W) \) of \( W \) is the sum of the weights of the labeled vertices, i.e., \( \text{we}(W) = \sum_{i=1}^{p} \sum_{j \in [\ell_i]} [v^i_j] \neq 0 \text{we}(v^i_j) \). Note that the weight of a vertex can be counted more than once if the vertex occurs as labeled more than once. A labeled walkage is injective if each label from \([k]\) appears in it at most once, and bijective if each label from \([k]\) appears in it exactly once. Note that every labeled walk in an injective labeled walkage is injective.

The set of ending vertices of a labeled walkage \( W \) of order \( p \) is \( T(W) = \{v^i_{\ell_i} \mid i \in [p]\} \). The tuple of starting vertices of \( W \) is \( \text{start}(W) = (v^1_1, \ldots, v^p_1) \). Let \( < \) be a total order on \( V(G) \). A labeled walkage is ordered if \( \text{start}(W) \) is ordered according to \( < \), i.e., \( v^i_1 < v^{i+1}_j \) holds for all \( 1 \leq i < p \). The asymmetry that the starting vertices are an ordered tuple while the ending vertices are an unordered set is essential for our algorithm. A labeled linkage is a labeled walkage where every vertex occurs at most once, i.e., the walks are vertex-disjoint paths.

We also define semiproper and proper labeled walkages. The intuition here is that, in Section 4.2, we define a polynomial over semiproper walkages (see also Definition 1). Then, walkages that are semiproper but not proper will be handled by using standard techniques and therefore we can focus on proper walkages. Dealing with proper walkages will be the most technical part of the proof. A labeled walkage is semiproper if it is injective, no walk in it contains labeled digons, and the ending vertices of the walkage are distinct, i.e., \( v^i_{\ell_i} \neq v^j_{\ell_j} \) for \( i \neq j \). A labeled walkage \( W \) is
proper if it is semiproper and all of its labeled indices correspond to vertices of different colors, i.e., \(|c(R(W))| = |\{(i, j) \mid r^c_{ij} \neq 0\}|\). Note that being proper implies that no vertex is labeled twice, and note that if \(W\) is bijective and proper then \(|c(R(W))| = k\).

4.2 Algorithm

We assume that there is a total order \(<\) on \(V(G)\), and for a set \(S \subseteq V(G)\) we denote by \(\text{ordv}(S)\) the tuple containing the elements of \(S\) ordered according to \(<\). Note that \(G\) contains a \((k, w)\)-colored \((S, T)\)-linkage of order \(p\) and length \(\ell\) if and only if there is a bijective proper ordered labeled linkage \(W\) with order \(p\), length \(\ell\), weight \(\text{we}(W) = w\), tuple of starting vertices \(\text{start}(W) = \text{ordv}(S)\), and set of ending vertices \(\text{T}(W) = T\). We define a family of labeled walkages that includes all such labeled linkages, but relaxes the condition of being a linkage to walkage, and the condition of being proper to semiproper.

For each integer \(\ell\), we define a family of labeled walkages \(C_\ell\) of length \(\ell\).

**Definition 1 (Family \(C_\ell\)).** Let \(\ell\) a positive integer. The family \(C_\ell\) consists of the bijective semiproper ordered labeled walkages \(W\) with order \(p\), length \(\ell\), weight \(\text{we}(W) = w\), tuple of starting vertices \(\text{start}(W) = \text{ordv}(S)\), and set of ending vertices \(\text{T}(W) = T\).

**Definition of the polynomial.** Let \(q = 2^{3+[\log_2 n]}\) and keep in mind that \(\text{GF}(q)\) is a finite field of characteristic 2 and order \(q \geq 8n\). Next, we define a polynomial over \(\text{GF}(q)\) that will be evaluated at a random point by our algorithm. For each edge \(uv \in E(G)\) we associate a variable \(f_e(uv)\), for each vertex \(v \in V(G)\) we associate a variable \(f_v(v)\), and for each color-label-pair \((x, y) \in [n] \times [k]\) we associate a variable \(f_c(x, y)\). For a labeled walk \(W = ((v_1, \ldots, v_\ell), (r_1, \ldots, r_\ell))\) we associate the monomial

\[
f(W) = \prod_{i=1}^{\ell-1} f_e(v_i v_{i+1}) \cdot \prod_{i \in [\ell] | r_i \neq 0} f_v(v_i) \cdot f_c(c(v_i), r_i).
\]

For a labeled walkage \(W = (W_1, \ldots, W^p)\) we associate the monomial

\[
f(W) = \prod_{i=1}^p f(W^i).
\]

For a family \(\mathcal{F}\) of labeled walkages we associate the polynomial

\[
f(\mathcal{F}) = \sum_{W \in \mathcal{F}} f(W).
\]

In particular, because the walkages in \(C_\ell\) are bijective, every monomial in the polynomial \(f(C_\ell)\) has degree \(\ell - p + 2k\), being a product of \(\ell - p\) variables corresponding to the edges of the walkage, \(k\) variables corresponding to the labeled vertices, and \(k\) variables corresponding to the color-label-pairs.

**Algorithm for finding a \((k, w)\)-colored \((S, T)\)-linkage.** Our algorithm for finding a \((k, w)\)-colored \((S, T)\)-linkage of order \(p\) works as follows. Starting with \(\ell = p\), we evaluate the polynomial \(f(C_\ell)\) at a random point \(x\) over \(\text{GF}(q)\), for increasing values of \(\ell\). If \(f(C_\ell)(x) \neq 0\), we return that \(G\) contains a \((k, w)\)-colored \((S, T)\)-linkage of order \(p\), and moreover that the shortest \((k, w)\)-colored \((S, T)\)-linkage of order \(p\) has length \(\ell\). Otherwise, we continue increasing \(\ell\) until \(\ell = n + 1\) in which case we return that \(G\) does not contain a \((k, w)\)-colored \((S, T)\)-linkage of order \(p\).
For the proof of correctness of the algorithm, in Section 4.3 we show that with probability at least 1/2 this algorithm returns the length of the shortest \((k, w)\)-colored \((S, T)\)-linkage of order \(p\), and never returns a length shorter than the shortest \((k, w)\)-colored \((S, T)\)-linkage of order \(p\).

**Proof of time complexity of the algorithm.** Next we prove the time complexity of the algorithm. The evaluation of the polynomial is done using dynamic programming. This is a standard application of dynamic programming over walks while keeping track of the set of labels used so far, the weight of the labeled vertices, and the set of ending vertices used. We prove that it can be performed in time \(2^{p+k}n^{O(1)}w\).

**Lemma 2.** Let \(S, T\) be disjoint subsets of \(V(G)\) of size \(|S| = |T| = p\), \(c: V(G) \rightarrow [n]\) a coloring of \(G\), \(w: V(G) \rightarrow Z_{\geq 1}\) a weight function, \(\ell \leq n\) an integer, \(k, w\) integers, and \(q = 2^{3+\lceil \log_2 n \rceil}\). The polynomial \(f(C_t)\) can be evaluated at a random point over GF\((q)\) in time \(2^{p+k}n^{O(1)}w\).

**Proof.** We associate random values over GF\((q)\) to all variables \(f_v, f_{uv}, f_{x,y}\) and from now denote by \(f_v, f_{vu}, f_{x,y}\) these associated values, and by extension for a walkage \(\mathcal{W}\) denote by \(f(\mathcal{W})\) the value associated to the monomial \(f(\mathcal{W})\) and for a family of walkages \(\mathcal{F}\) denote by \(f(\mathcal{F})\) the value associated to the polynomial \(f(\mathcal{F})\). Now, the task is to compute \(f(C_t)\).

Informally, we will compute \(f(C_t)\) by dynamic programming over partial walkages, growing the walkages one labeled walk at a time in the order specified by \(\text{ordv}(S)\).

Denote \(\text{ordv}(S) = (s_1, s_2, \ldots, s_p)\) and for any \(t \in [p]\) denote by \(\text{pre}_t(S)\) the length-\(t\) prefix of \(\text{ordv}(S)\). For every integer \(t \in [p]\), integer \(l \in [\ell]\), set \(L \subseteq [k]\) of labels, set \(T' \subseteq T\) of ending vertices, weight \(w' \in [0, w]\), vertices \(x, y \in V(G)\), and integer \(o \in \{0, 1\}\), we define

\[
D(t, l, L, T', w', x, y, o) = f(\mathcal{F}(t, l, L, T', w', x, y, o)),
\]

where we define \(\mathcal{F}(t, l, L, T', w', x, y, o)\) to be the family of labeled walkages \(\mathcal{W} = (W^1, \ldots, W_t)\), where for each \(i \in [l]\), \(W^i = ((v^i_1, \ldots, v^i_{t_i}), (r^i_1, \ldots, r^i_{t_i}))\), that satisfy the following properties:

1. Each labeled walk \(W^i\) in \(\mathcal{W}\) has length at least 2 and does not contain labeled digons,
2. \(\mathcal{W}\) has order \(t\) and ordered tuple of starting vertices \(\text{start}(\mathcal{W}) = \text{pre}_t(S)\),
3. \(\mathcal{W}\) has length \(t\),
4. \(\mathcal{W}\) is injective and the set of used labels is \(L\),
5. the set of ending vertices of all but the last walk in \(\mathcal{W}\) is \(T((W^1, \ldots, W^{t-1})) = T'\),
6. \(\mathcal{W}\) has weight \(w(\mathcal{W}) = w'\),
7. the last vertex of the last walk in \(\mathcal{W}\) is \(v^t_{t_t} = x\),
8. the second last vertex of the last walk in \(\mathcal{W}\) is \(v^t_{t_t-1} = y\), and
9. if \(o = 0\), then \(r_t, t_t = 0\), otherwise \(r_t, t_t \neq 0\).

In other words, \(t\) specifies the number of walks, \(l\) specifies the length, \(L\) specifies the used labels, \(T'\) specifies the used ending vertices, \(w'\) specifies the weight, \(x\) specifies the last vertex of the last walk, \(y\) specifies the second last vertex of the last walk, and \(o\) specifies whether the last vertex of the last walk is labeled. Note that it can be without loss of generality assumed that each walk has length at least 2 because \(S\) and \(T\) are disjoint.
Then, we define also a shorthand that for \( t \in [p], l \in [\ell], L \subseteq [k], T' \subseteq T, \) and \( w' \in [0,w], \)

\[
D(t, l, L, T', w') = \sum_{x \in T'} \sum_{y \in N(x)} \sum_{o \in \{0,1\}} D(t, l, L\setminus\{x\}, w', x, y, o),
\]

which intuitively denotes the polynomial corresponding to a “completed” walkage of \( t \) walks with length \( l \), used labels \( L \), used ending vertices \( T' \), and weight \( w' \).

Now it holds that \( f(C_\ell) = D(p, \ell, [k], T, w) \), and therefore computing \( f(C_\ell) \) can be done by computing all of the values \( D(t, l, L, T' \setminus \{x\}, w', x, y, o) \) by dynamic programming.

Next we specify this computation by dynamic programming. All values that we do not specify here will be set to zero. First, to initialize, we define a special value \( D(0, 0, \emptyset, \emptyset, 0) = 1 \) corresponding to a family of walkages containing one empty walkage.

Next, we describe computing the states where \( o = 0 \), i.e., the last vertex is not labeled, for all \( t \in [p], l \in [\ell], L \subseteq [k], T' \subseteq T, w' \in [0, w] \), \( x \in V(G) \), and \( y \in N(x) \), assuming that all the states with smaller \( l \) have already been computed. There are four cases, corresponding to the four lines of Equation (1). In the first case the walk \( W_t \) has length at least three, its second last vertex \( y \) is not labeled, and we are extending the walkage by adding one not labeled vertex \( x \) to \( W_t \). Second case is the same, but the second last vertex \( y \) is labeled and thus we have to ensure to not create a labeled digon. Third case is the case that we are extending the walkage by adding one more labeled walk, consisting of two vertices \( y, x \), where \( y = s_t \), neither of them labeled. Fourth case is like the third, but the first vertex \( y = s_t \) of the new walk is labeled. Recall the notation that \( [y = s_t] = 1 \) if \( y = s_t \) holds, and 0 otherwise.

\[
D(t, l, L, T', w', x, y, 0) = f_v(xy)
\]

\[
\cdot \left( \sum_{z \in V(G)} D(t, l - 1, L, T', w', y, z, 0) + \sum_{z \in V(G) \setminus \{x\}} D(t, l - 1, L, T', w', y, z, 1) + [y = s_t] \cdot D(t - 1, l - 2, L, T', w') + [y = s_t] \cdot \sum_{r \in L} f_v(y) \cdot f_c(c(y), r) \cdot D(t - 1, l - 2, L \setminus \{r\}, T', w' - \text{we}(y)) \right).
\]

Then, we describe computing the states where \( o = 1 \), i.e., the last vertex is labeled, for all \( t \in [p], l \in [\ell], L \subseteq [k], T' \subseteq T, w' \in [0, w] \), \( x \in V(G) \), and \( y \in N(x) \), assuming that all of the states with smaller \( l \) have already been computed. There are again four cases, analogously to Equation (1).
Lemma 3. If challenge. It is based on the technical Lemma 5, whose proof is postponed to Section 4.4. The remaining part of the proof of (b) is much more complicated and is the main technical argument is based on the now-standard technique of bijective labeling based cancellation introduced | set of vertices with | f |

Proof. Consider a (k, w)-colored (S, T)-linkage \( C \) of order \( p \) and length \( \ell \) and

\( f(C) \) is non-zero if \( |V(G)| \) contains a \( 2k \)-colored \( (S, T) \)-linkage of order \( p \) and length \( \ell \), then evaluating \( f(C) \) at a random point of \( GF(q) \) has probability at least 1/2 to be non-zero. From (b) it follows that if \( G \) does not contain a \( (k, w) \)-colored \( (S, T) \)-linkage of order \( p \) and length \( \leq \ell \), then evaluating \( f(C) \) at a random point is guaranteed to be zero. This establishes that the algorithm is correct with probability at least 1/2, with one-sided error.

The part (a) is relatively easy to prove (Lemma 3). To prove (b), we first show that the monomials in \( f(C) \) corresponding to non-proper labeled walkages cancel out (Lemma 4). This argument is based on the now-standard technique of bijective labeling based cancellation introduced in [2]. The remaining part of the proof of (b) is much more complicated and is the main technical challenge. It is based on the technical Lemma 5, whose proof is postponed to Section 4.4.

We start with (a).

Lemma 3. If \( G \) has a \( (k, w) \)-colored \( (S, T) \)-linkage of order \( p \) and length \( \ell \), then \( f(C) \) is non-zero.

Proof. Consider a \( (k, w) \)-colored \( (S, T) \)-linkage \( P \) of order \( p \) and length \( \ell \). Let \( X \subseteq V(P) \) be the set of vertices with \( |X| = k \), different colors, and weight \( w_e(X) = w \). We can turn \( P \) into a proper labeled linkage \( W \) of order \( p \), length \( \ell \), weight \( w \), where \( \text{start}(W) = \text{ordv}(S) \) and \( T(W) = T \), by

\[
D(t, l, L, T', w', x, y, 1) = \sum_{r \in L} f_r(x) \cdot f_c(c(x), r) \cdot f_e(xy)
\]

\[
\cdot \left( \sum_{z \in V(G)} D(t, l - 1, L \setminus \{ r \}, T', w' - we(x), y, z, 0) + \sum_{z \in V(G) \setminus \{ x \}} D(t, l - 1, L \setminus \{ r \}, T', w' - we(x), y, z, 1) + [y = s_t] \cdot D(t - 1, l - 2, L \setminus \{ r \}, T', w' - we(x)) \right)
\]

This completes the description of the dynamic programming, which shows that each of the states \( D(t, l, L, T', w', x, y, o) \) can be computed in \( n^{O(1)} \) time given the values of the states with smaller \( l \). As there are \( p \cdot \ell \cdot 2^k \cdot 2^p \cdot (w + 1) \cdot n \cdot n \cdot 2 = O(p2^{p+k}n^3w) \) states, the algorithm works in time \( 2^{p+k}n^{O(1)}w \). □

As the algorithm can be implemented by \( O(n) \) applications of Lemma 2, the algorithm has time complexity \( 2^{p+k}n^{O(1)}w \). Recovering the solution can be done by a factor of \( O(n^2) \) more applications.

4.3 Correctness

To prove the correctness of the algorithm, we show that

(a) the polynomial \( f(C) \) is non-zero if \( G \) contains a \( (k, w) \)-colored \( (S, T) \)-linkage of order \( p \) and length \( \ell \) and

(b) the polynomial \( f(C) \) is the identically zero polynomial if the graph does not contain a \( (k, w) \)-colored \( (S, T) \)-linkage of order \( p \) and length \( \leq \ell \).

Because \( f(C) \) has degree \( \ell - p + 2k \leq 3n \leq q/2 \), it follows from Lemma 1 and (a) that if \( G \) contains a \( (k, w) \)-colored \( (S, T) \)-linkage of order \( p \) and length \( \ell \), then evaluating \( f(C) \) at a random point of \( GF(q) \) has probability at least 1/2 to be non-zero. From (b) it follows that if \( G \) does not contain a \( (k, w) \)-colored \( (S, T) \)-linkage of order \( p \) and length \( \leq \ell \), then evaluating \( f(C) \) at a random point is guaranteed to be zero. This establishes that the algorithm is correct with probability at least 1/2, with one-sided error.

The part (a) is relatively easy to prove (Lemma 3). To prove (b), we first show that the monomials in \( f(C) \) corresponding to non-proper labeled walkages cancel out (Lemma 4). This argument is based on the now-standard technique of bijective labeling based cancellation introduced in [2]. The remaining part of the proof of (b) is much more complicated and is the main technical challenge. It is based on the technical Lemma 5, whose proof is postponed to Section 4.4.

We start with (a).
ordering the paths based on their starting vertices and assigning the labels \([k]\) arbitrarily to the vertices \(X\) when \(W\) intersects \(X\).

Therefore \(W \in C_\ell\), so it remains to prove that \(W\) is the only labeled walkage in \(C_\ell\) that corresponds to the monomial \(f(W)\), which then implies that the monomial \(f(W)\) occurs in the polynomial \(f(C_\ell)\) with coefficient 1, implying that \(f(C_\ell)\) is non-zero.

Notice that from \(f(W)\), from the edge variables \(f_e\) we can recover the edges \(E(W)\) of \(W\), from the vertex variables \(f_v\) we can recover the labeled vertices \(X\), and because vertices in \(X\) have different colors, from the color-label pair variables \(f_c\) we can recover how the labels correspond to the labeled vertices. Therefore as the ordering of the paths is fixed by \(\text{ordv}(S)\) and every vertex appears in \(W\) at most once, we have that \(W\) is the unique element of \(C_\ell\) that corresponds to the monomial \(f(W)\).

Then, we deal with non-proper walkages in \(C_\ell\). Let \(C_\ell^* \subseteq C_\ell\) denote the family of proper labeled walkages in \(C_\ell\), i.e., the labeled walkages in \(C_\ell\) where all labeled indices have vertices of different colors.

**Lemma 4.** It holds that \(f(C_\ell^*) = f(C_\ell)\).

**Proof.** We will show that there is a function \(\phi : C_\ell \setminus C_\ell^* \rightarrow C_\ell \setminus C_\ell^*\) that is an \(f\)-invariant fixed-point-free involution, i.e., for all \(W \in C_\ell \setminus C_\ell^*\) it holds that (1) \(f(\phi(W)) = f(W)\), (2) \(\phi(W) \neq W\), and (3) \(\phi(\phi(W)) = W\). This implies that the set \(C_\ell \setminus C_\ell^*\) can be partitioned into pairs \(\{W, \phi(W)\}\) with \(f(W) = f(\phi(W))\), and therefore every monomial corresponding to a labeled walkage in \(C_\ell \setminus C_\ell^*\) occurs in \(f(C_\ell)\) an even number of times, and therefore they cancel out because \(f\) is over a field of characteristic 2.

The function \(\phi\) is defined as follows. Let \(W = (W^1, \ldots, W^p)\) be a labeled walkage in \(C_\ell \setminus C_\ell^*\), where \(W^i = ((v^i_1, \ldots, v^i_{\ell^i}), (r^i_1, \ldots, r^i_{\ell^i}))\). Because \(W\) is semiproper but not proper, there exists two different labeled indices that have a vertex of the same color, i.e., pairs \((i, a)\) and \((j, b)\) with \(i, j \in [p], a \in [\ell_i], b \in [\ell_j], (i, a) \neq (j, b), c(v^i_a) = c(v^b_b), r^i_a \neq 0,\) and \(r^b_b \neq 0\). Let \((i, a), (j, b)\) be the lexicographically smallest such pair. We set \(\phi(W)\) to be the labeled walkage obtained from \(W\) after swapping \(r^i_a\) with \(r^b_b\).

First, we observe that \(\phi(W) \in C_\ell\). Indeed, it cannot make a bijective walkage into non-bijective, and as it does not change the sequence of vertices of \(W\) or which indices are labeled, it cannot make a semiproper walk into non-semi-proper, or change the order, the length, the weight, the tuple of starting vertices, or the set of ending vertices. Also \(\phi(W)\) is not proper, i.e., \(\phi(W) \in C_\ell \setminus C_\ell^*\), because the vertices \(v^i_a\) and \(v^j_b\) are still labeled and have the same color.

To see why \(f(\phi(W)) = f(W)\), note that, since \(\phi\) does not change the vertices, it also does not change the edge variables \(f_e\) of the monomial, it does not change which vertices are labeled so it does not change the vertex variables \(f_v\) of the monomial, and because the vertices \(v^i_a\) and \(v^j_b\) have the same color the color-label-pair variables \(f_c\) of the monomial are also not changed.

Also, we have that \(\phi(W) \neq W\), since the fact that \(W\) is bijective implies that \(r^i_a \neq r^j_b\). Also, \(\phi(\phi(W)) = W\) because the swapping does not change which indices are labeled, and therefore does not change the lexicographically smallest pair of labeled indices with the same colors.

As a result of Lemma 4, we can work with \(f(C_\ell^*)\) instead of \(f(C_\ell)\).

The most complicated part of the correctness proof will be to show part (b), that is, if there is no \((k, w)\)-colored \((S, T)\)-linkage of order \(p\) and length at most \(\ell\), then \(f(C_\ell^*)\) (and, thus by Lemma 4, \(f(C_\ell)\)) is an identically zero polynomial. Most of this proof will be presented in Section 4.4, but we introduce here the statement the lemma that we will prove in Section 4.4. For this, we define **barren labeled walkages**.
In other words, a labeled walkage $W$ of length $\ell$ is barren if its edges form a subgraph of $G$ where no labeled linkage $W'$ of length at most $\ell$ can have the same sets of starting vertices, ending vertices, and collected vertices as $W$. Intuitively, this means that the labeled walkage $W$ can not be “untangled” to give a corresponding labeled linkage. In particular, observe that because the “untangling” preserves the set of collected vertices, i.e., $R(W') = R(W)$, if no $(k,w)$-colored $(S,T)$-linkages of order $p$ and length at most $\ell$ exists, then all labeled walkages in $C_\ell^w$ are barren.

Next, we state the main technical lemma for establishing the correctness of our algorithm. Section 4.4 is devoted to its proof.

**Lemma 5.** Let $G$ be a graph and let $B$ be the set of all proper barren labeled walkages in $G$. There exists a function $\phi : B \to B$ so that for all $W \in B$, the function $\phi$ satisfies that

1. $\phi(\phi(W)) = W$ ($\phi$ is involution),
2. $\phi(W) \neq W$ ($\phi$ is fixed-point-free),
3. $f(\phi(W)) = f(W)$ ($\phi$ preserves the monomial),
4. $T(\phi(W)) = T(W)$ ($\phi$ preserves the set of ending vertices), and
5. $\text{start}(\phi(W)) = \text{start}(W)$ ($\phi$ preserves the ordered tuple of starting vertices).

The main reason for defining the function $\phi$ for all proper barren labeled walkages instead of just barren walkages in $C_\ell^w$ is that $\phi$ will be defined recursively, and in the recursion we will anyway need to handle all proper barren labeled walkages.

Now, the proof of (b) is an easy consequence of Lemma 5.

**Lemma 6.** If $G$ has no $(k,w)$-colored $(S,T)$-linkage of order $p$ and length $\leq \ell$, then $f(C_\ell^w)$ is an identically zero polynomial.

**Proof.** First, because $G$ has no $(k,w)$-colored $(S,T)$-linkage of order $p$ and length $\leq \ell$, all labeled walkages in $C_\ell^w$ are barren, i.e., $C_\ell^w \subseteq B$.

We show that if $W \in C_\ell^w$, then $\phi(W) \in C_\ell^w$. By definition, $\phi(W)$ is proper. By (3), $\phi$ preserves the set of labeled vertices and moreover because the labeled vertices have different colors it preserves also the label-vertex mapping, and therefore $\phi(W)$ is bijective and has weight $w$. By (4) and (5), $\phi$ preserves the set of ending vertices and the ordered tuple of starting vertices. By (3), $\phi$ also preserves the length $\ell$, as the order of $W$ is preserved by (5). Therefore the restriction $\phi|_{C_\ell^w}$ is a function $\phi|_{C_\ell^w} : C_\ell^w \to C_\ell^w$.

Then, by (1-3), $\phi|_{C_\ell^w}$ is an $f$-invariant fixed-point-free involution on $C_\ell^w$, implying that the set $C_\ell^w$ can be partitioned into pairs $\{W, \phi(W)\}$ with $f(W) = f(\phi(W))$, and therefore for every monomial $f(W)$, there is an even number of labeled walkages $W \in C_\ell^w$ corresponding to it, and therefore because $f(C_\ell^w)$ is over a field of characteristic 2, it is identically zero.

**4.4 Proof of Lemma 5**

In this subsection we prove Lemma 5 by explicitly defining the function $\phi$ and then showing that it has all of the required properties.
In order to define $\phi$ we first introduce some notation for manipulating labeled walks and labeled walkages. Let $W = ((v_1, \ldots, v_ℓ), (r_1, \ldots, r_ℓ))$ be a labeled walk. For indices $a, b$ with $1 \leq a \leq b \leq ℓ$, we denote by $W[a, b]$ the labeled subwalk between $a$ and $b$, inclusive, i.e., the labeled walk $W[a, b] = ((v_a, \ldots, v_b), (r_a, \ldots, r_b))$. If $a > b$, then $W[a, b]$ denotes an empty labeled walk.

The involution $\phi$ will use three types of operations: reversing a subwalk, swapping a label from one occurrence of a vertex to another occurrence of it (possibly in a different walk), and swapping suffixes of two walks.

The subwalk reversal operation is defined as follows. Let $W$ be a labeled walk of length $ℓ$ and $a, b$ indices with $1 \leq a < b \leq ℓ$. The walk obtained from $W$ by reversing the subwalk between $a$ and $b$, inclusive, including the labels, is denoted by $W[\overleftarrow{a, b}]$. For example, if $W = ((v_1, v_2, v_3, v_4), (0, 1, 0, 2))$, then $W[\overleftarrow{2, 3}] = ((v_1, v_3, v_2, v_4), (0, 0, 1, 2))$. A labeled walk $W$ is a palindrome if $W = W[\overleftarrow{1, ℓ}]$ holds, i.e., the labeled walk is the same in reverse. Note that $W[\overleftarrow{a, b}] = W$ holds if and only if $W[a, b]$ is a palindrome and that a subwalk $W[a, b]$ can be a palindrome only if its length is odd or if it is the empty walk. We will use the following lemma about palindromic subwalks of labeled walks, and in particular the reason to forbid labeled digons is to make this lemma true. Recall that any labeled walk in a proper labeled walkage is injective and does not contain labeled digons. Recall also that $R(W[a + 1, b − 1]) = \emptyset$ if and only if $W$ has no labels in the subwalk $W[a + 1, b − 1]$.

**Lemma 7.** Let $W = ((v_1, \ldots, v_ℓ), (r_1, \ldots, r_ℓ))$ be an injective labeled walk of length $ℓ$ that does not contain labeled digons, and let $a, b \in [ℓ]$. If $v_a = v_b$ and $W[a + 1, b − 1]$ is a palindrome, then $R(W[a + 1, b − 1]) = \emptyset$.

**Proof.** First, because $W[a + 1, b − 1]$ is injective and palindrome, the only vertex of $W[a + 1, b − 1]$ that can be labeled is the middle vertex. However, a label cannot occur at the middle vertex of a palindrome with more than one vertex because it would be a labeled digon. If $W[a + 1, b − 1]$ has exactly one vertex, then again this vertex cannot be labeled because $v_a = v_b$ and $W$ does not contain labeled digons.

The label swap operation is defined as follows. Let $W = (W^1, \ldots, W^p)$ be a labeled walkage of order $p$, where for each $i \in [p]$ the walkage $W^i$ is denoted by $((v_1^i, \ldots, v_{ℓ_i}^i), (r_1^i, \ldots, r_{ℓ_i}^i))$. Let $(i, a)$, $(j, b)$ be pairs with $i, j \in [p]$, $a \in [ℓ_i]$, $b \in [ℓ_j]$, $v_a^i = v_b^j$, and exactly one of $r_a^i$ and $r_b^j$ equal to zero (i.e., one of them unlabeled and one labeled). The labeled walkage obtained from $W$ by swapping $r_a^i$ with $r_b^j$ is denoted by $W \sim_{a,b}^{i,j}$. Note that because $r_a^i \neq r_b^j$, it holds that $W \sim_{a,b}^{i,j} \neq W$. See Figure 6 for an illustration of the label swap operation.

The suffix swap operation is defined as follows. Let $(i, a)$ and $(j, b)$ be pairs with $i, j \in [p]$, $a \in [ℓ_i + 1]$, $b \in [ℓ_j + 1]$, and $i \neq j$. The labeled walkage obtained from $W$ by swapping the suffix of $W^i$ starting at index $a$ with the suffix of $W^j$ starting at index $b$ is denoted by $W \leftrightarrow_{a,b}^{i,j}$. Note that here we allow that $a = ℓ_i + 1$ or $b = ℓ_j + 1$, with the interpretation that this corresponds to the empty suffix. Clearly, if both $a = ℓ_i + 1$ and $b = ℓ_j + 1$, then this operation does not do anything, but otherwise if $W$ is a proper labeled walkage, applying this operation will in fact always result in a different walkage because of the different ending vertices condition. See Figure 7 for an illustration of the suffix swap operation.

If $W^1$ and $W^2$ are labeled walks so that the last vertex of $W^1$ is adjacent to the first vertex of $W^2$, then $W^1 \circ W^2$ denotes the concatenation of $W^1$ and $W^2$. If $W = (W^1, \ldots, W^p)$ is a labeled walkage and $W$ is a labeled walk, then $W \circ W$ denotes the labeled walkage $(W \circ W^1, \ldots, W^p)$ and $W \sqcup W$ denotes the labeled walkage $(W, W^1, \ldots, W^p)$.

Next we define the function $\phi$ of Lemma 5. We will provide some intuition about $\phi$ right after the definition, and Figures 8 to 16 demonstrate different cases of it. The definition of $\phi$ will be recursive, using induction by the length of the walkage.
\[ ((v^1_1, \ldots, v^1_i), (r^1_1, \ldots, r^1_i)) \]
\[ \vdots \]
\[ ((v^i_a, \ldots, v^i_j), (r^i_a, \ldots, r^i_j)) \]
\[ \vdots \]
\[ ((v^p_a, \ldots, v^p_r), (r^p_a, \ldots, r^p_r)) \]
\[ W \]
\[ W \overset{i,j_{a,b}}{\sim} \]

**Figure 6:** An illustration of the label swap operation. On the left: a labeled walkage \( W = (W^1, \ldots, W^p) \), and pairs \((i, a), (j, b)\) with \(i, j \in [p]\), \(a \in [\ell_i]\), \(b \in [\ell_j]\), \(v^i_a = v^j_b\), and exactly one of \(r^i_a\) and \(r^j_b\) equal to zero. Note that we allow \(i = j\). On the right: the labeled walkage \( W \overset{i,j_{a,b}}{\sim} \).

\[ ((v^1_1, \ldots, v^1_i), (r^1_1, \ldots, r^1_i)) \]
\[ \vdots \]
\[ ((v^i_a, \ldots, v^i_j), (r^i_a, \ldots, r^i_j)) \]
\[ \vdots \]
\[ ((v^p_a, \ldots, v^p_r), (r^p_a, \ldots, r^p_r)) \]
\[ W \]
\[ W \overset{i,j_{a,b}}{\leftrightarrow} \]

**Figure 7:** An illustration of the suffix swap operation. On the left: a labeled walkage \( W = (W^1, \ldots, W^p) \), and pairs \((i, a), (j, b)\) with \(i, j \in [p]\), \(a \in [\ell_i + 1]\), \(b \in [\ell_j + 1]\), and \(i \neq j\). On the right: the labeled walkage \( W \overset{i,j_{a,b}}{\leftrightarrow} \).

**Definition 3** (The function \(\phi\)). Let \( W = (W^1, \ldots, W^p) \) be a proper barren labeled walkage of order \(p\). For each \(i \in [p]\), denote \( W^i = ((v^i_1, \ldots, v^i_i), (r^i_1, \ldots, r^i_i)) \). The value \(\phi(W)\) is defined, in some cases recursively, by selecting the first matching case from the following list:

**A.** if the vertex \(v^1_i\) occurs only once in \(W\):
1. if \(\ell_1 \geq 2\), then \(\phi(W) = W^1[1, 1] \circ \phi(W^1[2, \ell_1], W^2, \ldots, W^p)\).
2. otherwise (i.e., \(\ell_1 = 1\)), \(\phi(W) = W^1 \sqcup \phi(W^2, \ldots, W^p)\).

**B.** if the vertex \(v^1_i\) occurs in at least three different walks \(W^i\):
There must be at least two different walks \(W^i\) that contain \(v^1_i\) but do not contain it as labeled. Let \(i, j\) be the two smallest indices so that both \(W^i\) and \(W^j\) contain \(v^1_i\) but do not contain it as labeled. Let \(a\) be the index of the first occurrence of \(v^1_i\) in \(W^i\) and \(b\) be the index of the first occurrence of \(v^1_i\) in \(W^j\). Now, \(\phi(W) = W \overset{i,j_{a,b}}{\leftrightarrow} \).

**C.** if the vertex \(v^1_i\) occurs only in the walk \(W^1\):
By the case (A), the vertex \(v^1_i\) occurs multiple times in \(W^1\). Let \(b\) be the index of the last
occurrence of \(v_1^i\) in \(W^1\) and \(a\) be the index of the second last occurrence of \(v_1^i\) in \(W^1\). Note that \(a = 1\) if \(v_1^i\) occurs only twice in \(W^1\), and note also that \(1 \leq a \leq b - 2\).

1. if \(r_1^i = r_b^i = 0\):
   (a) if \(W^1[2, b - 1]\) is not a palindrome, then \(\phi(W) = (W^1[2, b - 1], W^2, \ldots, W^p)\).
   (b) otherwise, if \(b < \ell_1\), then \(\phi(W) = W^1[1, b] \circ \phi(W^1[b + 1, \ell_1], W^2, \ldots, W^p)\).
   (c) otherwise (i.e., \(b = \ell_1\)), \(\phi(W) = W^1 \sqcup \phi(W^2, \ldots, W^p)\).

2. if the index \(b\) is not a digon in \(W^1\), then \(\phi(W) = W \preceq_{1,b}^i\).
   Note: If neither case (1) nor (2) applies, then \(r_1^i \neq 0\).

3. if \(W^1[2, a - 1]\) is not a palindrome, then \(\phi(W) = (W^1[2, a - 1], W^2, \ldots, W^p)\).
   Note: If \(a = 1\), then \(W^1[2, a - 1]\) is the empty walk which is a palindrome.

4. if \(v_{a+1}^i = v_{b-1}^i\):
   (a) if \(W^1[a+1, b-1]\) is not a palindrome, then \(\phi(W) = (W^1[a+1, b-1], W^2, \ldots, W^p)\).
   (b) otherwise, \(\phi(W) = W^1[1, b] \circ \phi(W^1[b+1, \ell_1], W^2, \ldots, W^p)\).
      Note: Here \(W^1[b+1, \ell_1]\) cannot be an empty walk because by case (C.2) \(b\) is a digon in \(W^1\).

X. otherwise, \(\phi(W) = W^1[1, a] \circ \phi(W^1[a+1, \ell_1], W^2, \ldots, W^p)\).
   Note: The case C.X will form a “common case” with the case D.X.

D. if the vertex \(v_1^i\) occurs in exactly two different walks:
Let \(i\) be the index of the another walk \(W^i\) in which \(v_1^i\) occurs and let \(b\) be the index of the first occurrence of \(v_1^i\) in \(W^i\).

1. if \(r_1^i = r_b^i = 0\), then \(\phi(W) = W \leftrightarrow_{1,b}^{i,i}\).

2. if the index \(b\) is not a digon in \(W^i\), then \(\phi(W) = W \sim_{1,b}^{i,i}\).
   Note: If neither case (1) nor (2) applies, then \(r_1^i \neq 0\).

3. if \(v_1^i\) occurs at least twice in \(W^i\), then let \(c\) be the index of its second occurrence and \(\phi(W) = W \leftrightarrow_{2,c+1}^{i,i}\).
   Note: It can happen that one of the suffixes in this case is empty. However, both of them cannot be empty at the same time because \(W^1\) and \(W^i\) have different ending vertices because \(W\) is proper.
   Note: In the remaining cases, \(v_1^i\) occurs exactly once in \(W^i\), and this occurrence is a digon at index \(b\).
   Now, let \(a\) be the index of the last occurrence of \(v_1^i\) in \(W^1\) (if \(v_1^i\) occurs only once in \(W^1\), then \(a = 1\)).

4. if \(W^1[2, a - 1]\) is not a palindrome, then \(\phi(W) = (W^1[2, a - 1], W^2, \ldots, W^p)\).
   Note: If \(a = 1\), then \(W^1[2, a - 1]\) is the empty walk which is a palindrome.

5. if \(a = \ell_1\), then \(\phi(W) = W^1 \sqcup \phi(W^2, \ldots, W^p)\).

6. if \(v_{a+1}^i = v_{b+1}^i\), then \(\phi(W) = W \leftrightarrow_{a+1,b+1}^{i,i}\).
   Note: By case (5) it holds that \(a < \ell_1\) and by case (2) it holds that \(b < \ell_i\).

X. otherwise, \(\phi(W) = W^1[1, a] \circ \phi(W^1[a+1, \ell_1], W^2, \ldots, W^p)\).
   Note: The case D.X will form a “common case” with the case C.X.
Intuition for $\phi$. Before laboriously proving that $\phi$ indeed is a function from proper barren labeled walkages to proper barren labeled walkages satisfying the required properties, let us give some rough outline of ideas behind it. First, the general idea is that if the walkage $W$ goes to a certain case, then the walkage $\phi(W)$ goes again to the same case, which then maps it back to $W$. The only exception is that the cases C.X and D.X could map to each other.

Then, let us consider the cases relevant for a single walk, i.e., the case A.1 and the cases under C. Here, the intuition of case A.1 is to just move forward in the walk: we don’t care much about what $\phi$ does to the rest of the walk because it must preserve the vertex right after $v^1_1$, and attaching $v^1_1$ to the front will not create a digon because $v^1_1$ occurs only at one index. Then, case C.1.a is the standard loop reversal case, which is safe because neither index 1 nor $b$ is labeled. The case C.1.b (and C.1.c) corresponds to ignoring a palindromic subwalk, which can be safely done by Lemma 7. Then, case C.2 is the standard label swap case, which is safe because the index $b$ is a not digon (note that the index 1 is never a digon). The cases C.1–C.2 are in some sense the “easy cases”, while the cases C.3–C.X require more analysis of the remaining situation and quite unintuitive design. First, if neither C.1 nor C.2 applies, we know that the index 1 is labeled and the index $b$ is digon. The purpose of case C.3 is to, in some sense reduce to a situation where we pretend that the vertex $v^1_1$ occurs only at indices 1, $a$, and $b$, as the walk between 1 and $a$ is an irrelevant palindromic loop. Then, case C.4 handles a corner condition which would prevent case C.X from working. The case C.X ignores the palindromic loop between 1 and $a$, leaving the only occurrence of the vertex $v^1_1$ in the rest of the walk to be at the digon $b$, which in some sense makes it “harmless” in that the recursive calls will never need to analyse the vertex $v^b_1$ again as the first vertex.

The intuition for the case of multiple walks is as follows. First, the case A.2 is just an analogue of A.1 when the first walk has length 1. Then, if the vertex $v^1_1$ occurs multiple times, we consider three different cases: $v^1_1$ occurs in at least three walks, $v^1_1$ occurs in one walk, and $v^1_1$ occurs in two walks. Here, the three walks case B is quite easy, as we can just consider two of the walks where $v^1_1$ is not labeled, circumventing all issues with labeled digons. When $v^1_1$ occurs in only one walk we go to the one walk case C. Then, when $v^1_1$ occurs in two walks $W^1$ and $W^i$, the intuition of cases under D is that we concatenate $W^1$ with reversed $W^i$, with some special marker in between, and then apply the single walk cases under C for this concatenation. Here, in the case D.X this can change whether $v^1_1$ occurs in two walks or a single walk, and therefore it is necessary to have the
common case of C.X and D.X, moreover taking care in the proof that moving back from C.X to D.X will be handled correctly.

**Correctness proof for \( \phi \).** We will then proceed to first show that \( \phi \) is well-defined, then that \( \phi \) maps proper barren labeled walkages to proper barren labeled walkages, and then that \( \phi \) satisfies all of the properties stated in Lemma 5, with \( \phi(\phi(W)) = W \) being the most complicated of them to prove. The proof is long because we have to analyze most of the 18 cases one by one. However, most of the arguments in these proofs are relatively easy once the definition of \( \phi \) is set. The main challenge in the proof was to come up with the right definition of \( \phi \).

**Well-definedness of \( \phi \).** In Definition 3, in several cases, namely A.1, A.2, C.1.b, C.1.c, C.4.b, C.X, D.5, and D.X, the function \( \phi \) is defined recursively. A priori it is not even clear why the syntactic value \( \phi(W) \) is even well-defined in these cases. It requires proof that in these cases the recursive argument is in the domain of \( \phi \), in particular that it is also a proper barren labeled walkage.

Next we show that the syntactic value \( \phi(W) \) for proper barren labeled walkages \( W \) is well-defined. We remark that Lemma 8 does not yet show that \( \phi(W) \) is a proper barren labeled walkage; it will require more efforts to prove (see Lemmas 9 to 11).

**Lemma 8.** In case A.1 of Definition 3 it holds that \( (W^1[2, \ell_1], W^2, \ldots, W^p) \) is a proper barren labeled walkage, in cases A.2, C.1.c, and D.5 it holds that \( (W^2, \ldots, W^p) \) is a proper barren labeled walkage, in cases C.1.b, and C.4.b it holds that \( (W^1[b+1, \ell_1], W^2, \ldots, W^p) \) is a proper barren labeled walkage, and in cases C.X and D.X it holds that \( (W^1[a + 1, \ell_1], W^2, \ldots, W^p) \) is a proper barren labeled walkage.

**Proof.** In all cases, the labeled walkage used as the recursive argument is obtained from \( W \) by removing either the walk \( W^1 \) or a prefix of \( W^1 \). First we need to argue that the recursive argument is a labeled walkage. For this, the only thing to argue is that (1) the recursive argument contains at least one walk (i.e., \( p \geq 2 \) in cases A.2, C.1.c, and D.5) and that (2) all walks in the recursive
argument are non-empty (i.e. \( \ell_1 \geq 2 \) in case A.1, \( b < \ell_1 \) in cases C.1.b and C.4.b, and \( a < \ell_1 \) in cases C.X and D.X). The other properties of labeled walkages are clearly satisfied when removing either \( W^1 \) or a prefix of \( W^1 \).

The above conditions are satisfied directly by definition in cases A.1 and C.1.b. In case C.4.b, \( b < \ell_1 \) holds by the fact that (due to case C.2) the index \( b \) is a digon in \( W^1 \). In case C.X, recall that \( a \) is the index of the second last occurrence of \( v_1^1 \) in \( W^1 \), so \( a < \ell_1 \). In case D.X, we have that \( a < \ell_1 \) by case D.5. For the remaining cases A.2, C.1.c, and D.5, observe the following. If \( p = 1 \) would hold, then \( \mathcal{W} = (W^1) \). Then, since in all these three cases \( v_{b_2}^1 = v_1^1 \) and \( W^1[2, \ell_1] \) cannot contain labels (in A.2 trivially, in C.1.c by \( r_1^1 = r_b^0 = 0 \) and Lemma 7, and in D.5 by cases D.1, D.2, and D.4 combined with Lemma 7), it should hold that \( (W^1[1, 1]) \) is a labeled linkage consisting only of one walk with one vertex that would contradict the fact that \( \mathcal{W} \) is barren by Definition 2.

It is clear by definition of a proper labeled walkage that removing a walk or a prefix of a walk maintains that the walkage is proper. To complete the proof, it remains to show case by case that the labeled walkages used as recursive arguments are barren.

In all of the cases the proofs will follow the same template: For the sake of contradiction we suppose that the labeled walkage \( \mathcal{W} \) used as a recursive argument is not barren, then consider the labeled linkage \( \mathcal{W}' \) that witnesses that \( \mathcal{W}' \) is not barren, and then use \( \mathcal{W}' \) to construct a labeled linkage that shows that \( \mathcal{W} \) is not barren, obtaining a contradiction. We spell out these steps in detail for the case A.1, and in less detail for subsequent cases.

**Case A.1.** For the sake of contradiction, suppose that \( \mathcal{W}' = (W^1[2, \ell_1], W^2, \ldots, W^p) \) is not barren. Then by the definition of barren, there exists a labeled linkage \( \mathcal{W}'' \) with \( \text{start}(\mathcal{W}'') = \text{start}(\mathcal{W}') \), \( \mathcal{T}(\mathcal{W}'') = \mathcal{T}(\mathcal{W}') \), \( \mathcal{R}(\mathcal{W}'') = \mathcal{R}(\mathcal{W}') \), length \( \leq \ell - 1 \), and edges \( E(\mathcal{W}'') \subseteq E(\mathcal{W}') \). By the assumptions of case A.1, the labeled walkage \( \mathcal{W}' \) does not contain \( v_1^1 \), so the labeled linkage \( \mathcal{W}'' \) cannot contain
v^1_1 because the edge property \( E(W') \subseteq E(W) \) ensures that \( v^1_1 \) cannot occur in a walk of length more than one, and the start vertex property \( \text{start}(W') = \text{start}(W) \) ensures that \( v^1_1 \) cannot occur in a walk of length one. By the start vertex property, it holds that the first vertex of the first walk in \( W'' \) is \( v^1_2 \). Therefore, \( W'[1,1] \circ W'' \) is a labeled linkage. Because \( v^1_1 v^2_2 \in E(W) \) and \( E(W') \subseteq E(W) \subseteq E(W) \), we have that \( E(W'[1,1] \circ W'') \subseteq E(W) \). Also, observe that because of \( R(W'') = R(W) \), it holds that \( R(W'[1,1] \circ W'') = R(W) \). Similarly, we observe that \( \text{start}(W'[1,1] \circ W'') = \text{start}(W) \), \( T(W'[1,1] \circ W'') = T(W) \), and the length of \( W'' \) is at most \( \ell \). Therefore, \( W'[1,1] \circ W'' \) is a labeled linkage that according to Definition 2 contradicts the fact that \( W \) is barren.

**Case A.2.** Again, suppose that \( W' = (W^2, \ldots, W^p) \) is not barren, and consider the witness \( W'' \). Because \( v^1_1 \) does not occur in \( W'' \), it holds that \( W^1 \sqcup W'' \) is a labeled linkage that contradicts that \( W \) is barren.

**Case C.1.b.** Suppose that \( W' = (W^1[b+1,\ell_1], W^2, \ldots, W^p) \) is not barren and consider the witness \( W'' \). As, by definition of \( b \) in case C, \( v^1_1 \) occurs in \( W \) only in the subwalk \( W^1[1, b] \), it cannot occur in \( W'' \). Therefore \( W[1,1] \circ W'' \) is a labeled linkage. It is easy to observe that \( \text{start}(W[1,1] \circ W'') = \text{start}(W) \), \( T(W[1,1] \circ W'') = T(W) \), and \( E(W[1,1] \circ W'') \subseteq E(W) \). Also, Lemma 7 implies that \( R(W[1,1]) = \emptyset \) and therefore \( R(W[1,1] \circ W'') = R(W) \), therefore contradicting that \( W \) is barren.

**Case C.1.c.** This case is similar as the previous, in particular, Lemma 7 implies that \( R(W^1) = \emptyset \). Therefore, if we assume that \( W' = (W^2, \ldots, W^p) \) is not barren and we take the labeled linkage \( W'' \) that witnesses that \( W' \) is not barren, we can construct a labeled linkage \( W'[1,1] \sqcup W'' \) that contradicts the fact that \( W \) is barren.

**Case C.4.b.** Assume that \( W' = (W^1[b+1,\ell_1], W^2, \ldots, W^p) \) is not barren and take the labeled linkage \( W'' \) that witnesses that \( W' \) is not barren. As \( v^1_1 \) occurs in \( W \) only in the subwalk \( W^1[1, b] \), it cannot occur in \( W'' \). Therefore \( W[1,1] \circ W'' \) is a labeled linkage. Note that in this case \( W[2,a-1] \) is a palindrome, the index \( a \) of the walk \( W^1 \) is not labeled because the index 1 is
Figure 16: Example of cases D.1-D.6 of the definition of \( \phi \). In each grey bag, all vertices inside
the corresponding bag are the same vertex of the graph. Labeled vertices are depicted in red and
green. White vertices correspond to unlabeled vertices and black vertices can be either labeled or
unlabeled.

labeled and \( \mathcal{W} \) is proper, and \( W_1[a+1, b-1] \) is a palindrome, and the index \( b \) of \( W_1 \) is not labeled.
Therefore, by Lemma 7, \( R(W_1[1, b]) = \{ v_1 \} \), and therefore \( R(W_1[1, 1] \circ \mathcal{W}''') = R(\mathcal{W}) \), and therefore
we contradict the fact that \( \mathcal{W} \) is barren.

Case C.X. In this case, the argument is less apparent because \( v_1 \) indeed occurs in \( W_1[a+1, \ell_1] \).
Again, we start by assuming that \( \mathcal{W}'' = (W_1[a+1, \ell_1], W_2, \ldots, W_p) \) is not barren, and take the
labeled linkage \( \mathcal{W}''' \) that witnesses that \( \mathcal{W}''' \) is not barren. Now, note that \( v_1 \) occurs in \( \mathcal{W}'' \) only as a
single digon in \( W_1[a+1, \ell_1] \). Therefore, \( v_1 \) cannot occur as a starting or ending vertex in \( \mathcal{W}''' \). Also,
there is only one edge in \( E(\mathcal{W}) \) incident to \( v_1 \), which then prevents \( v_1 \) occurring at any position in \( \mathcal{W}''' \),
because any position containing \( v_1 \) would have to a digon, but \( \mathcal{W}''' \) is labeled linkage and thus
does not contain digons. Therefore, we construct a labeled linkage \( W_1[1, 1] \circ \mathcal{W}''' \) and use the fact
that \( W_1[2, a-1] \) is palindrome with Lemma 7 to conclude that \( R(W_1[1, 1] \circ \mathcal{W}''') = R(\mathcal{W}) \), and
to finally observe that \( W_1[1, 1] \circ \mathcal{W}''' \) satisfies all the other needed properties to contradict the
fact that \( \mathcal{W} \) is barren.

Case D.5. Note that here we have that \( r_1^1 \neq 0 \), \( v_1^1 = v_{\ell_1}^1 \), and \( W_1[2, \ell_1 - 1] \) is a palindrome. The
arguments are similar to case C.X: The vertex \( v_1^1 \) does occur in the walkage \( (W_2, \ldots, W_p) \), but it
occurs in it only a single time, which is a digon in the walk \( W_1 \). So suppose that \( \mathcal{W}'' = (W_2, \ldots, W_p) \)
is not barren, and consider the witness \( \mathcal{W}''' \). By similar arguments as in case C.X, \( v_1^1 \) cannot occur
in \(W''\). Therefore, we construct a labeled linkage \(W^1[1,1] \sqcup W''\) and use the fact that \(W^1[2, a - 1]\) is palindrome with Lemma 7 to conclude that \(R(W^1[1,1] \sqcup W'') = R(W)\), and to finally observe that \(W^1[1,1] \sqcup W''\) satisfies all the other needed properties to contradict the fact that \(W\) is barren.

**Case D.X.** Note that here again, we have that \(r_1^1 \neq 0\) and \(W^1[2, a - 1]\) is a palindrome, where \(a\) is the last occurrence of \(v_1^1\) in \(W^1\). The arguments are similar to case C.X: The vertex \(v_1^1\) does occur in the walkage \((W^1[a + 1, \ell_1], W^2, \ldots, W^p)\), but it occurs in it only a single time, which is a digon in the walk \(W^1\). So again suppose that \(W' = (W^1[a + 1, \ell_1], W^2, \ldots, W^p)\) is not barren, and consider the witness \(W''\). Again by arguments of C.X we have that \(v_1^1\) cannot occur in \(W''\). Therefore, we again construct a labeled linkage \(W^1[1,1] \circ W''\) and use the fact that \(W^1[2, a - 1]\) is palindrome with Lemma 7 to conclude that \(R(W^1[1,1] \circ W'') = R(W)\), and to finally observe that \(W^1[1,1] \circ W''\) satisfies also all the other needed properties to contradict that \(W\) is barren. \(\square\)

The next three lemmas establish that \(\phi(W)\) is a proper barren labeled walkage. In addition, Lemma 9 shows that \(\phi\) satisfies the properties \(f(\phi(W)) = f(W)\), \(T(\phi(W)) = T(W)\), and \(\text{start}(\phi(W)) = \text{start}(W)\).

**Lemma 9.** Let \(W\) be a proper barren labeled walkage. It holds that \(\phi(W)\) is a labeled walkage, \(\text{start}(\phi(W)) = \text{start}(W)\), \(T(\phi(W)) = T(W)\), and \(f(\phi(W)) = f(W)\).

**Proof.** We prove the lemma by induction on the length of the walkage \(W\). Here, all of the cases should be easy to verify, so the arguments we provide will be terse.

Cases A.1 works directly by induction, in particular, we can use the induction assumptions that

- \(\phi(W^1[2, \ell_1], W^2, \ldots, W^p)\) is a labeled walkage,
- \(\text{start}(\phi(W^1[2, \ell_1], W^2, \ldots, W^p)) = \text{start}(W^1[2, \ell_1], W^2, \ldots, W^p))\),
- \(T(\phi(W^1[2, \ell_1], W^2, \ldots, W^p)) = T(W^1[2, \ell_1], W^2, \ldots, W^p))\), and
- \(f(\phi(W^1[2, \ell_1], W^2, \ldots, W^p)) = f(W^1[2, \ell_1], W^2, \ldots, W^p))\),

to prove the same properties for \(W\). In particular, we use the start vertex property to ensure that the first vertex of the first walk of \(\phi(W^1[2, \ell_1], W^2, \ldots, W^p)\) is \(v_2^1\), and therefore the first edge of the first walk of \(W^1[1,1] \circ \phi(W^1[2, \ell_1], W^2, \ldots, W^p)\) is \(v_1^1v_2^1\).

Case A.2 works similarly to A.1. Case B works by the property that \(v_1^b = v_2^b\), in particular observing that if both of the suffixes are non-empty, the suffix swap operation preserves the set of ending vertices \(T(W)\). Case C.1.a works because \(v_1^1 = v_2^1\), and cases C.1.b and C.1.c work by similar induction as A.1. Case C.2 works because \(v_1^1 = v_2^1\), in particular, even though the index of the label in the walk changes, the vertex variable or the label-color pair variable do not change because the vertex does not change. Case C.3 works because \(v_1^1 = v_2^1\). Case C.4.a works because \(v_1^a = v_2^b\) and case C.4.b works by similar induction as A.1. Case C.X works again by induction.

Case D.1 works because \(v_1^1 = v_2^1\). Case D.2 works because \(v_1^1 = v_2^1\), by the same argument as case C.2.

In case D.3, all other conditions work directly by \(v_1^1 = v_2^i\), but we should pay attention to the ending vertices condition \(T(\phi(W)) = T(W)\), because it can happen that one of the suffixes is empty. As observed already in the definition, observe that at most one of the suffixes can be empty because \(v_1^1 = v_2^i\) and \(W^1\) and \(W^i\) have different ending vertices because \(W\) is proper. First, if \(\ell_1 = 1\), then the ending vertex of \(W^i\) becomes \(v_{i}^c = v_1^i = v_{\ell_1}^i\), and the ending vertex of \(W^1\) becomes \(v_{\ell_1}^1\), so the condition holds. Second, if \(\ell_i = c\), then the ending vertex of \(W^1\) becomes \(v_1^1 = v_c^i = v_{\ell_i}^i\), and the ending vertex of \(W^i\) becomes \(v_{\ell_1}^1\), so the condition holds.
Case D.4 works because \( v_1^1 = v_2^1 \). Case D.5 works by induction. Case D.6 works because \( v_{a+1}^1 = v_{b+1}^r \) and the suffixes are guaranteed to be non-empty. Case D.X works by induction. \( \square \)

Note that because \( f(W) \) and \( \text{start}(W) \) determine \( R(W), E(W), \) and the length of \( W \) uniquely, Lemma 9 implies that \( \phi(W) \) is a barren walkage (because \( W \) is barren). It remains to prove that \( \phi(W) \) is proper, and to prove that \( \phi(W) \) is proper the only remaining thing to prove is that \( \phi(W) \) does not contain labeled digons (Lemma 11). In particular, the property \( T(\phi(W)) = T(W) \) guarantees that the ending vertices of \( \phi(W) \) are distinct, and \( f(\phi(W)) = f(W) \), which implies \( R(\phi(W)) = R(W) \) guarantees that the labeled vertices of \( \phi(W) \) have different colors (because \( W \) is proper).

We will make use of the following lemma that follows directly from Lemma 9.

**Lemma 10.** If a vertex occurs exactly once in \( W \) and this occurrence is a digon, then this vertex also occurs exactly once in \( \phi(W) \) and this occurrence is also a digon with the same adjacent vertices.

**Proof.** Suppose that a vertex \( v \) occurs exactly once in \( W \) and this occurrence is a digon. Therefore, it cannot be a starting or ending vertex in \( W \). By Lemma 9, it holds that \( \text{start}(\phi(W)) = \text{start}(W) \) and \( T(\phi(W)) = T(W) \) and therefore \( v \) cannot be a starting or ending vertex neither in \( \phi(W) \). Then, since by Lemma 9, we have that \( f(\phi(W)) = f(W) \), implying \( E(\phi(W)) = E(W) \), the vertex \( v \) must have the exactly same adjacent vertices in \( \phi(W) \) as in \( W \). \( \square \)

Then we prove that \( \phi(W) \) has no labeled digons.

**Lemma 11.** Let \( W \) be a proper barren labeled walkage. The labeled walkage \( \phi(W) \) has no labeled digons.

**Proof.** We prove the lemma by induction on the length of the walkage \( W \).

**Case A.1.** In this case, as \( \phi(W^1[2, \ell_1], W^2, \ldots, W^p) \) has no labeled digons by induction, the only potential place for a labeled digon could be index 2 of \( W^1 \). However, because \( v_1^1 \) occurs only once in \( W \) and therefore does not occur in \( \phi(W^1[2, \ell_1], W^2, \ldots, W^p) \), we have that the index 2 of \( W^1 \) cannot become a digon.

**Case A.2.** Trivially by induction.

**Case B.** Here, by the definition of \( v_a^i \) and \( v_b^j \) in case B, both \( v_a^i \) and \( v_b^j \) are unlabeled and \( v_a^i = v_b^j \) holds, so if \( W \leftrightarrow_{a,b} \) would contain a labeled digon then also \( W \) would.

**Case C.1.a.** Here, the indices 1 and \( b \) of \( W^1 \) are not labeled so they cannot become labeled digons. For indices in \([2, b - 1]\), note that if \( i \in [2, b - 1] \) would be a labeled digon in \( W^1[2, b - 1] \), then \( b + 1 - i \) would have been a labeled digon in \( W^1 \).

**Case C.1.b.** Potential places for labeled digons here are indices \( b \) and \( b + 1 \) at \( W^1 \). However, \( b \) is not labeled so no labeled digon can be at \( b \), and because \( v_1^j \) does not occur in \((W^1[b + 1, \ell_1], W^2, \ldots, W^p)\), it cannot occur in \( \phi(W^1[b + 1, \ell_1], W^2, \ldots, W^p) \) and therefore \( b + 1 \) cannot become labeled digon.

**Case C.1.c.** Trivially by induction.

**Case C.2.** The index 1 of \( W^1 \) is not digon by definition of digon, and the index \( b \) is not digon by definition of this case, so no labeled digons are created.

**Case C.3.** Here, the index 1 cannot be a digon by definition, and the index \( a \) of \( W^1 \) has \( r_a = 0 \) by case C.2, so they cannot become labeled digons. For indices in \([2, a - 1]\), the same argument as in case C.1.a applies.

**Case C.4.a.** Neither index \( a \) nor \( b \) is labeled so they cannot become labeled digons, and for indices in \([a + 1, b - 1]\) the same argument as in case C.1.a applies.
Case C.4.b. Same argument as in C.1.b, and using that by case C.2, it holds that \( r_1^i = 0 \).

Case C.X. Here, the index a of \( W^1 \) cannot become a labeled digon because it is not labeled. For the index \( a + 1 \) the argument is more complicated: First note that the vertex \( v_1^a = v_1^b (= v_1^i) \) occurs only once in \( (W^1[a + 1, \ell_1], W^2, \ldots, W^p) \) (as \( v_1^b \)). Also, by case C.2, b is a digon in \( W^1 \). Therefore, by Lemma 10, the vertex \( v_1^b = v_1^a \) occurs in \( \phi(W^1[a + 1, \ell_1], W^2, \ldots, W^p) \) only once, and this occurrence is a digon adjacent with vertex \( v_0^b = v_0^a \). Because by Lemma 9 \( \phi \) preserves the starting vertices, and by case C.4 it holds that \( v_0^a \neq v_0^b \), it holds that the occurrence of \( v_1^b \) in \( \phi(W^1[a + 1, \ell_1], W^2, \ldots, W^p) \) cannot be as the second vertex of the first walk. Therefore, the index \( a + 1 \) of the first walk cannot be a labeled digon in \( W^1[1, a] \circ \phi(W^1[a + 1, \ell_1], W^2, \ldots, W^p) \).

Case D.1. As \( r_1^i = r_1^a = 0 \) and \( v_1^i = v_1^a \), it holds that if \( W \leftrightarrow_{1, i}^1 \) would contain a labeled digon then also \( W \) would.

Case D.2. The index 1 of \( W^1 \) cannot become a digon by definition of digon. The index b of \( W^i \) cannot become a digon by definition of this case.

Case D.3. Because \( v_1^a = v_1^i \), in this case if \( W \leftrightarrow_{1, c+1}^{1, i} \) would have a labeled digon at index 2 of \( W^1 \) or at index \( c + 1 \) of \( W^r \), then this same labeled digon would have existed also in \( W \). Then, no labeled digon can be created to index 1 of \( W^1 \) by definition of digon or to index c of \( W^i \) because \( r_1^c = 0 \) and therefore \( r_1^i = 0 \) by case D.2.

Case D.4. The index 1 of \( W^1 \) cannot become a digon by definition, and the index a cannot become a labeled digon because it is not labeled because the index 1 is labeled. For indices in \( [2, a - 1] \), the same argument as in case C.1.a applies.

Case D.5. Trivially by induction.

Case D.6. In this case, by cases D.1 and D.2 we have that \( r_1^a \neq 0 \) and \( r_1^i = 0 \). The former implies that \( r_1^b = 0 \). Therefore, the index a of \( W^1 \) nor the index b of \( W^i \) cannot become labeled digons because they cannot become labeled. Then, if the index \( a + 1 \) of \( W^1 \) would be a labeled digon in \( W \leftrightarrow_{1, a+1}^{1, i} \), then the index \( b + 1 \) of \( W^i \) would have been a labeled digon in \( W \) because \( v_1^a = v_1^b \) (and symmetrically for \( b + 1 \) of \( W^i \)).

Case D.X. Here, a similar argument as in C.X works: By Lemma 10 we know that \( v_1^b \) occurs only as a digon surrounded by \( v_1^{b-1} = v_1^{i+1} \) in \( \phi(W^1[a + 1, \ell_1], W^2, \ldots, W^p) \). Therefore, because \( \phi \) maintains the starting vertices and \( v_1^{a+1} \neq v_1^{b+1} \), it holds that the index \( a + 1 \) of \( W^1 \) cannot be a digon in \( W^1[1, a] \circ \phi(W^1[a + 1, \ell_1], W^2, \ldots, W^p) \). The index a of \( W^1 \) cannot become a labeled digon because it is not labeled.

The function \( \phi \) is an involution. Now we have shown that \( \phi \) is a function \( \phi: B \rightarrow B \), where \( B \) is the set of all barren proper labeled walkages, and that \( f(\phi(W)) = f(W) \), \( \mathcal{T}(\phi(W)) = \mathcal{T}(W) \), and \( \text{start}(\phi(W)) = \text{start}(W) \) hold. Next we show that \( \phi \) is an involution on \( B \), i.e., \( \phi(\phi(W)) = W \) holds.

Lemma 12. For any proper barren labeled walkage \( W \) it holds that \( \phi(\phi(W)) = W \).

Proof. We use induction on the length of walkage \( W \). The structure of the proof is to show that in all cases except C.X and D.X, the walk \( \phi(W) \) goes to the same case of Definition 3 as \( W \). Then, the cases C.X and D.X are treated together.

Case A.1. In both \( W \) and \( \phi(W) \) it holds that the \( v_1^1 \) occurs only once in the walkage and \( \phi \) does not change the first vertex, so \( \phi(\phi(W)) = W \) holds by induction.

Case A.2. In this case the first walk of \( W \) is the same as the first walk of \( \phi(W) \), so \( \phi(\phi(W)) = W \) holds by induction.
Case B. In this case, the function $\phi$ preserves the set of walks in which $v_1$ occurs, and moreover preserves the walk in which $v_1$ occurs as labeled (if it occurs as labeled in any walk). Therefore the walkage $\phi(W)$ also goes to case B, and in case B the indices $i, j$ selected for $\phi(W)$ are the same as selected for $W$. The suffix swap operation also does not change the indices of the first occurrences of $v_1$ in $W^i$ and $W^j$, so the indices $a$ and $b$ selected are the same. Then, the lemma follows by observing that $W \leftrightarrow_{i,j} W = W$.

At this point, let us observe that $W$ goes to cases C.1-4 if and only if $\phi(W)$ goes to cases C.1-4. This is because all of these cases maintain that $v_1$ occurs only in the walk $W^1$, but multiple times in the walk $W^1$. In particular, in the recursive cases this is maintained by the fact that $v_1$ does not occur in the recursive argument.

We also observe that $W$ goes to cases D.1-6 if and only if $\phi(W)$ goes to cases D.1-6. This is because all of these cases maintain that $v_1$ occurs in exactly two different walks. In the cases D.1-4 and D.6 this is easy to observe since these are not recursive, and in the case D.5 this follows from the fact that the walk $W^1$ is not changed and that in this case $v_1$ occurs exactly once in $(W^2, \ldots, W^p)$.

Case C.1. Note that going to the cases C.1.a, C.1.b, and C.1.c depends only on the last occurrence $b$ of $v_1$, and the labels $r_1$ and $r_1$. None of these cases change these, so $\phi(\phi(W)) = W$ holds in case C.1.a because $W^1[2, b-1] = W^1$ and reversing a subwalk does not change whether it is a palindrome, and by induction in cases C.1.b and C.1.c.

Case C.2. The order of the vertices in the walks and the fact that exactly one of $r_1$ and $r_b$ is labeled is maintained, so if $W$ goes to case C.2 then also $\phi(W)$ goes to C.2. Then, $\phi(\phi(W)) = W$ because $W \sim_{1,b} \sim_{1,b} W = W$.

Case C.3. It holds that $a < b$, so therefore reversing $W^1[2, a - 1]$ does not change the fact that $b$ is a digon in $W^1$. It also does not change the index $a$ of the second last occurrence of $v_1$, nor the index $b$ of the last occurrence of $v_1$, nor the fact that $r_1 \neq 0$, nor the fact that $W^1[2, a - 1]$ is not a palindrome.

Case C.4.a. Because $v_{a+1} = v_{b-1}$, reversing $W^1[a + 1, b - 1]$ does not change the fact that $b$ is a digon in $W^1$. Reversing $W^1[a + 1, b - 1]$ also does not change the index $a$ of the second last occurrence of $v_1$, nor the index $b$ of the last occurrence of $v_1$, nor the fact that $r_1 \neq 0$, nor the fact that $W^1[2, a - 1]$ is a palindrome.

Case C.4.b. Going to the case C.4.b depends only on the subwalk $W^1[1, b]$ and on the vertex with index $b + 1$ in $W^1$ (whether the index $b$ is a digon). Clearly, $\phi$ does not change the subwalk $W^1[1, b]$ in this case. The vertex with index $b + 1$ is not changed because by Lemma 9 the starting vertices are preserved by $\phi(W^1[b + 1, \ell_1], W^2, \ldots, W^p)$, so the lemma holds by induction.

Before moving to the cases C.X and D.X, we handle the cases D.1-D.5.

Case D.1. In this case, the operation $W \leftrightarrow_{1,b} W$ does not change the two walks in which $v_1$ occurs, nor it changes the fact that the first occurrence of $v_1$ in $W^i$ is at index $b$, nor that $r_1 = r_b = 0$. The lemma follows from the fact that $W \leftrightarrow_{1,b} W = W$.

Case D.2. This case does not change the vertices of the walks, so it is maintained that $v_1$ occurs only in $W^1$ and $W^i$. It also does not change the fact that exactly one of $r_1$ and $r_b$ is labeled or the fact the index $b$ is not a digon in $W^i$, so the lemma follows from the fact that $W \sim_{1,b} W = W$.

Case D.3. This case does not change the index $b$ of the first occurrence of $v_1$ in $W^i$ or the index $c$ of the second occurrence of $v_1$ in $W^i$, and neither does it change the fact that $r_1 \neq 0$. Because $c > b + 1$, it also does not change that the index $b$ is a digon in $W^i$. The lemma follows from the fact that $W \leftrightarrow_{2,c+1} W = W$. 

34
Case D.4. This case does not change the index b of the first occurrence of \( v^1 \) in \( W^i \), nor that \( r^i_1 \neq 0 \), nor that \( b \) is a digon in \( W^i \), nor that \( v^1 \) occurs only once in \( W^i \). It also does not change the index \( a \) of the last occurrence of \( v^1 \) in \( W^i \), or the fact that \( W^1[2, a - 1] \) is a palindrome so the lemma holds.

Case D.5. The vertex \( v^1 \) occurs exactly once in \((W^2, \ldots, W^p)\), so it is also maintained that \( v^1 \) occurs in exactly two walks. The walk \( W^1 \) is not changed, so it is maintained that \( r^1 \neq 0 \) and therefore \( \phi(W) \) does not go to case D.1. By Lemma 10 we have that \( \phi(W) \) does not go to case D.2, and again because \( v^1 \) occurs only once in \((W^2, \ldots, W^p)\) we have that \( \phi(W) \) does not go to case D.3. As \( W^1 \) is not changed we have that \( \phi(W) \) does not go to case D.4. Then, as case D.5 does not change the walk \( W^1 \), it is maintained that \( a = \ell_1 \), so \( \phi(W) \) goes to case D.5.

Case D.6. This case does not change that \( r^1 \neq 0 \), so \( \phi(W) \) does not go to case D.1. It also does not change the index b of the first occurrence of \( v^1 \) in \( W^i \), and because \( v^1_{a+1} = v^1_{b+1} \), it does not change that \( b \) is a digon in \( W^i \), and therefore \( \phi(W) \) does not go to case D.2. Because \( a \) is the last occurrence of \( v^1 \) in \( W^i \), it also does not change that \( v^1 \) occurs in \( W^i \) only once, so \( \phi(W) \) does not go to case D.3. It also does not change the index a of the last occurrence of \( v^1 \) in \( W^1 \) or the subwalk \( W^1[2, a - 1] \), so \( \phi(W) \) does not go to cases D.4. or D.5. Therefore, \( \phi(W) \) goes to case D.6 with the same values of \( a, b, \) and \( i \), so \( \phi(\phi(W)) = W \) holds because \( W \phi^{1,1}_{a+1, b+1} \phi^{1,1}_{a+1, b+1} = W \).

Case C.X. We aim to prove that if \( W \) goes to case C.X, then \( \phi(W) \) also goes to case C.X or to the case D.X, with the same value of the index \( a \), and therefore \( \phi(\phi(W)) = W \) will hold by induction, as in both cases \( \phi \) is defined as \( \phi(W) = W^1[1, a] \circ \phi(W^1[a+1, \ell_1]), W^2, \ldots, W^p \). Therefore, \( \phi(W) \) does not go to case A or B.

Suppose that \( v^1 \) occurs in \( \phi(W) \) only in the walk \( W^1 \), i.e., goes to case C. We will show that \( \phi(W) \) goes to case C.X. It is maintained that \( r^1 \neq 0 \), so \( \phi(W) \) does not go to case C.1. Then, by Lemma 10 it is maintained that the last occurrence of \( v^1 \) must be a digon so \( \phi(W) \) does not go to case C.2. Also, this case does not change the index a of the second last occurrence of \( v^1 \) nor the walk \( W^1[2, a - 1] \), so it is maintained that \( W^1[2, a - 1] \) is a palindrome and therefore \( \phi(W) \) does not go to case C.3. Then, to argue that \( \phi(W) \) does not go to case C.4, observe that because \( \phi \) maintains the starting vertex, the vertex at the position \( a + 1 \) is maintained. The vertices around the digon at the last occurrence of \( v^1 \) are maintained by Lemma 10, so therefore if \( W \) does not go to case C.4 then also \( \phi(W) \) does not go to case C.4. Therefore \( \phi(W) \) goes to case C.X, and as the walk \( W^1[1, a] \) is maintained, it goes to this case with the same value of \( a \), so the lemma holds by induction.

Then, suppose that \( v^1 \) occurs in \( \phi(W) \) in two walks \( W^1 \) and \( W^i \), i.e., goes to case D. We will show that \( \phi(W) \) goes to case D.X. It is maintained that \( r^1 \neq 0 \), so \( \phi(W) \) does not go to case D.1. Then, by Lemma 10 it must be that \( v^1 \) occurs in \( W^i \) only once and as a digon, and therefore \( \phi(W) \) does not go to case D.2 or D.3. Now, it will hold that the index a of the last occurrence of \( v^1 \) in the walk \( W^1 \) of \( \phi(W) \) is the same as the index a of the last occurrence of \( v^1 \) in the walk \( W^1 \) of \( W \). Therefore, the subwalk \( W^1[1, a] \) will be the same in \( W \) and \( \phi(W) \), and therefore \( \phi(W) \) will not go to case D.4 because \( W \) did not go to case C.3. Then, because \( \phi \) cannot turn a non-empty walk into an empty walk, it is maintained that the length of \( W^1 \) is more than \( a \), so \( \phi(W) \) cannot go to case D.5. Then, for case D.6 we again note that \( \phi \) maintains the vertex at position \( a + 1 \), and that the digon around the occurrence of \( v^1 \) outside of \( W^1[1, a] \) is maintained by Lemma 10. Therefore, \( \phi(W) \) goes to case D.X with the same value of \( a \), so the lemma holds by induction.

Case D.X. We will show that if \( W \) goes to case D.X, then \( \phi(W) \) also goes to case D.X or to the case C.X, with the same value of the index \( a \), and therefore \( \phi(\phi(W)) = W \) will hold by induction, as in both cases \( \phi \) is defined as \( \phi(W) = W^1[1, a] \circ \phi(W^1[a+1, \ell_1]), W^2, \ldots, W^p \). First, it is maintained
that $v_1^1$ occurs more than once, so therefore $\phi(W)$ does not go to case A. Then, as $v_1^1$ occurs only once in $(W^1[α + 1, ℓ_1], W^2, \ldots, W^p)$, it can occur in at most two walks in $\phi(W)$, so $\phi(W)$ cannot go to case B.

Suppose that $v_1^1$ occurs in $\phi(W)$ only in the walk $W^1$, i.e., goes to case C. We will show that $\phi(W)$ goes to case C.X. It is maintained that $r_1^1 \neq 0$, so $\phi(W)$ does not go to case C.1. Then, because $W^1[α, α] \mid W$ contains all other occurrences of $v_1^1$ in $W$ except the occurrence in $W^i$, it must be that now the last occurrence of $v_1^1$ in $W^1$ of $\phi(W)$ corresponds to the occurrence of $v_1^1$ in $W^i$ of $W$, in particular, by Lemma 10 the last occurrence of $v_1^1$ in $W^1$ of $\phi(W)$ must be a digon, and therefore $\phi(W)$ does not go to case C.2. By the same reasoning, it also must be that the the index $α$ of the last occurrence of $v_1^1$ in $W^1$ of $W$ is the same as the index $α$ of the second last occurrence of $v_1^1$ in $W^1$ of $\phi(W)$, and therefore the subwalk $W^1[α, α]$ of $\phi(W)$ in case C is the same as the subwalk $W^1[α, α]$ of $W$ in case D. Then, it follows that because $W$ did not go to case D.4, $\phi(W)$ does not go to case C.3. Then, as the case D.X does not change the vertex at the index $α + 1$ of $W^1$, it holds that the vertex at the index $α + 1$ of $W^1$ is the same in $W$ and $\phi(W)$. Also, by Lemma 10 the vertices around the digon of the last occurrence of $v_1^1$ are the same in $W$ and $\phi(W)$, so $\phi(W)$ does not go to case C.4 because $W$ did not go to case D.5. Therefore, $\phi(W)$ must go to case C.X, and we already reasoned that the index $α$ is the same as for $W$ in the case D.X, so the lemma holds by induction.

Suppose that $v_1^1$ occurs in $\phi(W)$ in two different walks, i.e., goes to case D. We will show that $\phi(W)$ goes to case D.X. First, it is maintained that $r_1^1 \neq 0$, so $\phi(W)$ does not go to case D.1. Then, we note that the walk $W^i$ that contains the other occurrence of $v_1^1$ may be different for $\phi(W)$ than $W$. However, it is maintained that $v_1^1$ occurs only once outside of $W^1$, and by Lemma 10 that the other occurrence is a digon and the vertices around this digon are maintained. Therefore, $\phi(W)$ does not go to case D.2, nor to the case D.3. Now, the index $α$ of the last occurrence of $v_1^1$ in $W^1$ will be the same for $\phi(W)$ and $W$ because $\phi(W)$ does not change the subwalk $W^1[α, α]$. Therefore, it is maintained that $W^1[2, α − 1]$ is a palindrome, and therefore $\phi(W)$ does not go to case D.4. Then, because $\phi$ cannot turn a non-empty walk into an empty walk, it is maintained that the length of $W^1$ is more than $α$, so $\phi(W)$ cannot go to case D.5. Then, by the start vertex property of $\phi$, the vertex at index $α + 1$ of $W^1$ is also maintained, and by Lemma 10 the vertices around the digon of the other occurrence of $v_1^1$ are maintained, so $\phi(W)$ does not go to case D.6. Therefore, $\phi(W)$ goes to the case D.X, with the same value of $α$, and therefore the lemma holds by induction.

Finally, we show that $\phi$ is fixed-point-free.

**Lemma 13.** For any proper barren labeled walkage $W$ it holds that $\phi(W) \neq W$.

*Proof.* We prove this by induction on the length $ℓ$ of the walkage. In the recursive cases A.1, A.2, C.1.b, C.1.c, C.4.b, C.X, D.5, and D.X this holds directly by induction. In cases B, D.1, D.3, and D.6, $\phi$ changes the suffixes of two walks, and at least one of the suffixes is non-empty. Because $W$ is proper, the ending vertices of all walks in $W$ are different, so the ending vertex of at least one of the walks involved in the suffix swap is changed (in fact the ending vertices of both of the walks change, but it is not necessary for this proof). In cases C.1.a, C.3, C.4.a, and D.4, $\phi$ reverses a non-palindromic subwalk so $\phi(W) \neq W$. In cases C.2 and D.2 $\phi$ changes a label from one position to another, so $\phi(W) \neq W$.

This completes the proof of Lemma 5.
5 From colored graphs to frameworks

In this section, we extend our results from weighted colored graphs to weighted frameworks, in particular, we prove Theorem 5 (recall that Theorem 5 implies Theorem 3), and then discuss even further extensions to frameworks where the matroid is not necessarily represented over a finite field.

5.1 Frameworks

We recall definitions related to frameworks.

**Matroids.** We refer to the textbook of Oxley [41] for the introduction to Matroid Theory.

**Definition 4.** A pair \( M = (V, \mathcal{I}) \), where \( V \) is a ground set and \( \mathcal{I} \) is a family of subsets of \( V \), called independent sets of \( M \), is a matroid if it satisfies the following conditions, called independence axioms:

(I1) \( \emptyset \in \mathcal{I} \),

(I2) if \( X \subseteq Y \) and \( Y \in \mathcal{I} \) then \( X \in \mathcal{I} \),

(I3) if \( X, Y \in \mathcal{I} \) and \( |X| < |Y| \), then there is \( v \in Y \setminus X \) such that \( X \cup \{v\} \in \mathcal{I} \).

An inclusion maximal set of \( \mathcal{I} \) is called a base. We use \( V(M) \) and \( \mathcal{I}(M) \) to denote the ground set and the family of independent sets of \( M \), respectively.

Let \( M = (V, \mathcal{I}) \) be a matroid. We use \( 2^V \) to denote the set of all subsets of \( V \). A function \( r: 2^V \to \mathbb{Z}_{\geq 0} \) such that for every \( X \subseteq V \),

\[
r(X) = \max\{|Y|: Y \subseteq X \text{ and } Y \in \mathcal{I}\}
\]

is called the rank function of \( M \). The rank of \( M \), denoted \( r(M) \), is \( r(V) \); equivalently, the rank of \( M \) is the size of any base of \( M \). A matroid \( M' = (V, \mathcal{I}') \) is a \( k \)-truncation of \( M = (V, \mathcal{I}) \) if for every \( X \subseteq V \), \( X \in \mathcal{I}' \) if and only if \( X \in \mathcal{I} \) and \( |X| \leq k \).

We work with several particular types of matroids. A uniform matroid is defined by the ground set \( V \) and its rank \( r \); every subset \( S \) of \( V \) of size at most \( r \) is independent. Partition matroids are the matroids that can be written as disjoint sums of uniform matroids. Transversal matroids arise from graphs. For a bipartite graph \( G = (U \cup B, E) \) with all edges between \( U \) and \( B \), we can define a matroid \( M = (V, \mathcal{I}) \) such that a set \( S \subseteq V \) is independent if there exists a matching in \( G \) such that every vertex in \( S \) is an endpoint of a matching edge.

**Matroid representations.** Let \( M = (V, \mathcal{I}) \) be a matroid and let \( \mathbb{F} \) be a field. An \( r \times n \)-matrix \( A \) is a representation of \( M \) over \( \mathbb{F} \) if there is a bijective correspondence \( f \) between \( V \) and the set of columns of \( A \) such that for every \( X \subseteq V \), \( X \in \mathcal{I} \) if and only if the set of columns \( f(X) \) consists of linearly independent vectors of \( \mathbb{F}^r \). Equivalently, \( A \) is a representation of \( M \) if \( M \) is isomorphic to the column matroid of \( A \), that is, the matroid whose ground set is the set of columns of the matrix and the independence of a set of columns is defined as the linear independence. If \( M \) has a such a representation, then \( M \) is representable over \( \mathbb{F} \) and it is also said \( M \) is a linear (or \( \mathbb{F} \)-linear) matroid. We can assume that the number of rows \( r = r(M) \) for a matrix representing \( M \) [37].

Whenever we consider a linear matroid, it is assumed that its representation is given and the size of \( M \) is \( \|M\| = \|A\| \), that is, the bit-length of the representation matrix. Notice that given a representation of a matroid, deciding whether a set is independent demands a polynomial number of field operations. In particular, if the considered field is a finite or is the field of rationals, we can verify independence in time that is a polynomial in \( \|M\| \).
Frameworks. A framework is a pair \((G, M)\), where \(M = (V, \mathcal{I})\) is a matroid whose ground set is the set of vertices of \(G\), i.e., \(V(M) = V(G)\). A weighted framework is a triple \((G, M, \omega)\), where \((G, M)\) is a framework and \(\omega : V(G) \to \mathbb{Z}_{\geq 1}\) is a weight function. An \((S, T)\)-linkage \(P\) in a weighted framework \((G, M, \omega)\) is \((k, w)\)-ranked if \(V(P)\) contains a set \(X \subseteq V(P)\) with \(X \in \mathcal{I}\), size \(|X| = k\), and weight \(\omega(X) = w\). When discussing algorithms for (weighted) frameworks, we explicitly specify how \(M\) is represented.

5.2 From colored graphs to frameworks

In this section we prove Theorem 5. We reduce the more general cases of matroids to the case of

\[ \text{Theorem 4.} \]

We start by giving our algorithm for the special case when the rank of \(M\) is bounded by \(k\), in particular when \(M\) is represented as a \(k \times n\) matrix.

\[ \text{Lemma 14. There is a randomized algorithm, that given a weighted framework \((G, M, \omega)\), where} \]

\[ \text{G is an} n\text{-vertex graph and \(M\) is represented as a} k \times n\text{matrix over a finite field of order} q, \text{sets of} \]

\[ \text{vertices} S, T \subseteq V(G), \text{and integers} p, k, w, \text{in time} 2^{p+O(k^2 \log q)} n^{O(1)} w \text{either finds a} (k, w)\text{-ranked} \]

\[ \text{\((S, T)\)-linkage of order} p \text{and of minimum total length, or determines that} (G, M, \omega) \text{has} \]

\[ \text{no} (k, w)\text{-ranked} (S, T)\text{-linkages of order} p. \]

\[ \text{Proof.} \] The matrix has at most \(q^k\) distinct column vectors so we can guess the \(k\) column vectors forming the independent set \(X\) of size \(k\) that we are looking for with at most \(q^{k^2}\) guesses. By inserting \(|S|\) new vertices with neighborhoods equal to \(S\), all-zero column vector, and weight 1, we can assume that the vectors of the starting vertices \(S\) will never correspond to the guessed basis. Then, we \(k + 1\)-color the graph, assigning the color \(k + 1\) to the vertices of the set \(S\) and other vertices whose column vectors are not in the guessed basis, and the colors \([k]\) to the other vertices according to which of the \(k\) guessed column vectors they correspond to. We also assign the weight of all vertices whose column vector is not in the guessed basis to be 1.

Then, \((G, M, \omega)\) has a \((k, w)\)-ranked \((S, T)\)-linkage of order \(p\) if and only if it has a \((k+1, w+1)\)-colored \((S, T)\)-linkage of order \(p\). In particular, the extra color \(k + 1\) contributes weight one and one color more, and the selected set \(X \subseteq V(P)\) without the extra color must correspond to an independent set of \(M\). Therefore, we get an algorithm with time complexity \(q^k 2^{p+k n^{O(1)} w} = 2^{p+O(k^2 \log q)} n^{O(1)} w\).

By extending Lemma 14 to matrices with a large number of rows by using randomized lossy truncation, we prove Theorem 5, which we restate here.

\[ \text{Theorem 5. There is a randomized algorithm that given a weighted framework \((G, M, \omega)\), where} \]

\[ \text{G is an} n\text{-vertex graph and \(M\) is represented as a} k \times n\text{matrix over a finite field of order} q, \text{sets of} \]

\[ \text{vertices} S, T \subseteq V(G), \text{and integers} p, k, w, \text{in time} 2^{p+O(k^2 \log (q+k))} n^{O(1)} w \text{either finds a} (k, w)\text{-ranked} \]

\[ \text{\((S, T)\)-linkage of order} p \text{and of minimum total length, or determines that} (G, M, \omega) \text{has} \]

\[ \text{no} (k, w)\text{-ranked} (S, T)\text{-linkages of order} p. \]

\[ \text{Proof.} \] Let \(A\) be a \(r \times n\) matrix representing \(M\). Our goal is to obtain a “lossy” representation of the \(k\)-truncation of \(M\) as a \(k \times n\) matrix over a field of order \(O(q+k^2)\). In particular, a representation so that any independent set of \(M\) of size \(k\) is independent in the representation with probability \(\geq \frac{1}{2}\), and any dependent set of \(M\) is dependent in the representation. Then, we obtain the algorithm by applying the algorithm of Lemma 14. Note that \(2^{p+O(k^2 \log (q+k^2))} n^{O(1)} w = 2^{p+O(k^2 \log (q+k))} n^{O(1)} w\).

We use two techniques from [37], increasing the order of the field and truncation. First, we make sure that the order of the field is at least \(2k\) by choosing the least integer \(i\) such that \(q^i \geq 2k\),
and going to the field of order \( q^i \), as detailed in Proposition 3.2 of [37]. Now, we can assume that \( A \) is over a field of order at least \( 2k \) and at most \( O(q + k^2) \).

Then, we truncate the matroid by multiplying the matrix \( A \) by a random \( k \times r \) matrix \( R \), in particular we claim that the \( k \times n \) matrix \( B = RA \) is now the desired representation of the \( k \)-truncation of \( M \). The analysis here is the same as in Proposition 3.7 of [37], but with a smaller field. In particular, let us consider a subset \( U \) of the ground set of \( M \), and let \( A_0 \) be the \( r \times |U| \) submatrix of \( A \) corresponding to \( S \). Now, in \( B \), the \( k \times |U| \) submatrix corresponding to \( S \) will be the matrix \( B_0 = RA_0 \). The rank of \( B_0 \) is at most the rank of \( A_0 \), so if \( U \) is dependent in \( M \) it will be dependent in the representation by \( B \). Then, assume that \( U \) is an independent set of \( M \) and \( |U| = k \). Now, \( det RA_0 \) can be considered as a degree-\( k \) non-zero polynomial whose variables are the \( kr \) random entries of \( R \). Therefore, by Lemma 1, the probability that \( det RA_0 = 0 \) is at most \( k/2k \).

With minor adjustments, Theorem 5 can be adapted for frameworks with matroids that are in general not representable over a finite field of small order. For example, uniform matroids, and more generally transversal matroids, are representable over a finite field, but the field of representation must be large enough. We first show how Theorem 1 can be applied in the case of transversal matroids.

**Theorem 6.** There is a randomized algorithm that given a weighted framework \((G, M, \mathbf{w})\), where \( G \) is an \( n \)-vertex graph and \( M \) is a transversal matroid represented by the corresponding bipartite graph, sets of vertices \( S, T \subseteq V(G) \), and integers \( p, k, w \), in time \( 2^{p+O(k^2 \log k)} n^{O(1)} w \) either finds a \((k, w)\)-ranked \((S, T)\)-linkage of order \( p \) and of minimum total length, or determines that \((G, M, \mathbf{w})\) has no \((k, w)\)-ranked \((S, T)\)-linkages of order \( p \).

**Proof.** We will construct a representation of the transversal matroid as a linear matroid over a finite field of order \( O(k) \), so that any independent set of \( M \) of size \( k \) is independent in the representation with probability \( \geq 1/2 \), and any dependent set of \( M \) is dependent in the representation. This yields the algorithm by then using Theorem 5.

Our construction is the same as the construction of [37], except by using a smaller field. We choose the least prime \( p \) with \( p \geq 2k \) and work in the field of order \( p \). Let the bipartition of the vertices of the bipartite graph be \((A, B)\). We construct an \(|B| \times |A|\) matrix, so that an entry of the matrix is a random element of the field if it corresponds to an edge, and zero otherwise. Now, the determinant of a submatrix is guaranteed to be zero if there is no corresponding matching, so any dependent set of \( M \) is dependent in the representation. Otherwise, the determinant of a \( k \times k \) submatrix can be seen as a non-zero degree-\( k \) polynomial that was evaluated at a random point. Therefore, as \( p \geq 2k \), by Lemma 1, the probability that it is non-zero is at least \( 1/2 \).

It is also possible to apply Theorem 5 in the situation when \( M \) is represented by an integer matrix over rationals with entries bounded by \( n^{O(k)} \).

**Theorem 7.** There is a randomized algorithm that given a weighted framework \((G, M, \mathbf{w})\), where \( G \) is an \( n \)-vertex graph and \( M \) is represented as an integer matrix over rationals with entries bounded by \( n^{O(k)} \), sets of vertices \( S, T \subseteq V(G) \), and integers \( p, k, w \), in time \( 2^{p+O(k^2 \log k)} n^{O(1)} w \) either finds a \((k, w)\)-ranked \((S, T)\)-linkage of order \( p \) and of minimum total length, or determines that \((G, M, \mathbf{w})\) has no \((k, w)\)-ranked \((S, T)\)-linkages of order \( p \).

**Proof.** Let \( c \) be a constant so that the entries of the matrix are bounded by \( n^{ck} \). We pick a random prime \( p \) among the first \( 2 \log_2(k!n^{ck}) \) primes, go to the finite field of order \( p \) by taking every entry modulo \( p \), and then apply the algorithm of Theorem 1.
We first analyze the time complexity and then the correctness. By the prime number theorem, the prime \( p \) is bounded by
\[
p = \mathcal{O}(\log k! n^{ck^2} \cdot \log \log k! n^{ck^2}) = \mathcal{O}(k^3 \log n \log \log n).
\]
We can find such random prime in \( n^{O(1)} \) time by elementary methods. Then, the time complexity by using Theorem 1 will be \( 2^{\mathcal{O}(p + k^2 \log(k + k^3 \log n \log \log n))} n^{O(1)} \). Denote \( t(n) = \log n \log \log n \) and consider two cases. First, if \( t(n) \leq k^5 \), then the time complexity is \( 2^{\mathcal{O}(p + k^2 \log(k^8))} n^{O(1)} = 2^{\mathcal{O}(p + k^2 \log k)} n^{O(1)} \). Second, if \( t(n) > k^5 \), then the time complexity is \( 2^{\mathcal{O}(p + k^2 \log k^3 t(n))} n^{O(1)} = 2^{\mathcal{O}(p + (t(n)^{1/2} \log t(n)) n^{O(1)} = 2^{\mathcal{O}(p) \cdot 2^{\mathcal{O}(\log^{1/2} n \log^{O(1)} \log n)}} = 2^{\mathcal{O}(p) n^{O(1)}} \).

Then, for the correctness we show that any dependent set of \( M \) is dependent in the representation and any independent set of \( M \) is independent in the representation with probability \( \geq 1/2 \). Let \( A \) be a square submatrix of the original representation and \( A_p \) the corresponding submatrix in the presentation modulo \( p \). Now, \( \det A_p = \det A \mod p \). Therefore, all dependent sets stay dependent. Then, assume that \( A \) is a \( k \times k \) submatrix corresponding to an independent set, i.e., \( \det A \neq 0 \). Now, the independent set can change into dependent only if \( \det A \) is divisible by \( p \). The value \( \det A \) is bounded by \( k! n^{ck^2} \), so there are at most \( \log_2(k! n^{ck^2}) \) primes dividing it. We chose \( p \) randomly among the first \( 2 \log_2(k! n^{ck^2}) \) primes, so with probability \( \geq 1/2 \) the prime \( p \) does not divide \( \det A \).

\[\square\]

6 Deterministic algorithm for longest \((S,T)\)-linkage

This section is dedicated to the proof of Theorem 2, which we restate next for convenience.

**Theorem 2.** There is a deterministic algorithm that, given an \( n \)-vertex digraph \( G \), two sets of vertices \( S, T \subseteq V(G) \), an integer \( p \), and an integer \( k \), in time \( p^{O(kp)} n^{O(1)} \) either returns a directed \((S,T)\)-linkage of order \( p \) and of total length at least \( k \), or determines that \( G \) has no directed \((S,T)\)-linkage of order \( p \) and total length at least \( k \).

We start with showing the main combinatorial lemma behind the theorem. The lemma is illustrated in Figure 3.

**Lemma 15.** Let \( G \) be a digraph and let \( C_1, \ldots, C_q \) be disjoint sets in \( V(G) \). For \( s, t \in V(G) \) let \( P_1, \ldots, P_q \) be internally-disjoint \((s,t)\)-paths. For each \( i \in [q] \), let \( v_i \in V(P_i) \) be such that the suffix of \( P_i \) starting from \( v_i \) lies inside \( C_i \), except for \( t \). For \( i \in [q] \), let \( Q_i \) be a path from \( v_i \) to \( t \) with all internal vertices in \( C_i \). Then there exist internally-disjoint \((s,t)\)-paths \( P'_1, \ldots, P'_q \) such that \( P'_i \) is either (i) \( P_i \) or (ii) a composition of a prefix of \( P_i \) not containing any vertices of \( P_i \) beyond \( v_i \), and a suffix of \( Q_j \) for some \( j \in [q] \), and there is at least one path of type (ii) among \( P'_1, \ldots, P'_q \).

**Proof.** For each \( i \in [q] \), denote the subpath of \( P_i \) from \( s \) to \( v_i \) by \( P_i^{-} \), and from \( v_i \) to \( t \) by \( P_i^{\rightarrow} \). First, assume there exists \( i \in [q] \) such that \( Q_i \) does not share a common internal vertex with any \( P_j^{-} \), \( j \in [q] \). In this case, the solution is immediate: for each \( j \neq i \), set \( P'_j = P_j \), and set \( P'_i = P_i^{\rightarrow} \circ Q_i \). The path \( P_i^{\rightarrow} \) does not intersect any other \( P'_j \) internally since \( P_1, \ldots, P_q \) are internally-disjoint, and by the assumption \( Q_i \) internally intersects neither \( P_j^{-} \) nor any other \( P'_j \). So for the remaining part of the proof we assume that for each \( Q_i \) there exists \( j \in [q] \) such that \( Q_i \) and \( P_j^{\rightarrow} \) share a common internal vertex.

We now show the statement by analyzing a certain token sliding game. Intuitively, we put a token on each of the paths \( P_1, \ldots, P_q \), originally on the place of the first intersection between \( P_i \) and some \( Q_j \) (see Figure 3b). Then we slide the tokens further along the paths according to certain
rules, until the tokens reach a state where no rules can be applied (Figures 3c and 3d). Our goal
is to show that in this case we obtain the desired paths \( P'_1, \ldots, P'_q \).

More formally, we define a state \( S \) as a tuple \((t_1, \ldots, t_q)\), where \( t_i \in V(P_i) \) for each \( i \in [q] \). The
original state \( S^1 = (t_1^1, \ldots, t_q^1) \) is defined as follows: for each \( i \in [q] \), \( t_i^1 \) is the closest to \( s \) vertex
along \( P_i \) that belongs to \( Q_j \) for some \( j \in [q] \). The game then proceeds iteratively, constructing the state \( S^{h+1} \) from \( S^h \) for each \( h \) starting from \( h = 1 \) by applying one of the following rules. For \( j \in [q] \), we shall refer to a path \( Q_j \) as active if \( t_j^h \neq t \).

**Clear** Let \( t_i^h = v_i \) for some \( i \in [q] \). Then set \( t_i^{h+1} = t \), and for each \( j \in [q] \) such that \( t_j^h \in V(Q_j) \setminus \{t\} \),
set \( t_j^h \) to be the next vertex along the path \( P_j \) that belongs to an active \( Q_{j'} \) for some \( j' \in [q] \),
here \( Q_i \) is not considered active. For all remaining \( j \in [q] \), set \( t_j^{h+1} = t_j^h \).

**Push** Let \( i \) and \( i', i \neq i' \in [q] \), be such that both \( t_i^h \) and \( t_i^h \) belong to \( Q_j \setminus \{t\} \) for some \( j \in [q] \);
additionally, let \( t_i^h \) be the farthest of two from \( t \) along \( Q_j \). Set \( t_i^h \) to be the next vertex along
the path \( P_i \) that belongs to an active \( Q_{j'} \) for some \( j' \in [q] \). For all \( i'' \in [q] \), \( i'' \neq i \), set \( t_i^{h+1} = t_i^{h'} \).

As long as there is a possibility, a Clear rule is applied; Push is only applied if no Clear is
available. If there are several options for applying the same rule, ties are breaking arbitrarily. We
observe that every application of each rule moves at least one of the state vertices further along its
respective path, and these vertices are never moved back. Thus, after a finite number of steps we
reach a state where neither of the rules is applicable. Denote this state by \( S^T \), our goal is to show the following.

**Claim 6.1.** Let \( S^T = (t_1^T, \ldots, t_q^T) \). There is \( i \in [q] \) such that \( t_i^T \neq t \).

Before showing the proof of Claim 6.1 we make the following simple observation.

**Claim 6.2.** For a state \( S^h \), \( h \in [T] \), for each \( j \in [q] \), \( t_j^h \in V(P_j) \), and \( t_j^h \) is either \( t \) or belongs to
an active \( Q_i \) for some \( i \in [q] \). Moreover, all of \( t_j^h \) that are not \( t \), are distinct.

**Proof.** By construction, the first part of the statement holds for the starting state \( S^1 \). Both rules
either move vertices to an active \( Q_i \) further along its path, or directly to \( t \). The second part follows
immediately from the fact that the paths \( P_j \) are internally-disjoint.

We first explain how Claim 6.1 implies the claim in the lemma. By Claim 6.1, there is at least
one \( i \in [q] \) such that \( t_i^T \neq t \); let \( I \subset [q] \) be the set of all indices with this property. By Claim 6.2,
each vertex in \( \{t_i^T\}_{i \in I} \) lies on an active \( Q_j \) for some \( j \in [q] \), and these vertices are all distinct.
Moreover, since the rule Push is not applicable, no two of these vertices share the same \( Q_j \). Let
\( \pi : I \to [q] \) be the injection that maps \( i \in I \) to the index \( j \) such that \( t_i^T \in V(Q_j) \). We construct
the desired family of paths as follows: for \( i \in [q] \setminus I \), let \( P_i^t \) be \( P_i \), and for \( j \in I \), let \( P_j^t \) be a concatenation
of the subpath of \( P_i \) from \( s \) to \( t_j^T \) (denoted \( \hat{P}_i \)), and the subpath of \( Q_{\pi(i)} \) from \( t_j^T \) to \( t \) (denoted \( \hat{Q}_i \)). Observe also that for each \( i \in I \), \( t_i^T \) is not \( v_i \) since the rule Clear is not applicable.
Moreover, since \( I \) is non-empty, there is at least one path of type (ii) in the constructed family. It
only remains to show that the paths \( \{P_i^t\}_{i \in [q]} \) are internally-disjoint.

For \( i \in [q] \setminus I \), denote \( \hat{P}_i = P_i^+ \) and \( \hat{Q}_i = P_i^- \). For \( i \neq i' \in [q] \), \( s \) is the only intersection
between \( \hat{P}_i \) and \( \hat{P}_{i'} \) since \( \hat{P}_i \) and \( \hat{P}_{i'} \) are proper prefixes of \( P_i \) and \( P_{i'} \) respectively, and these paths
are internally-disjoint by the assumption of the lemma. Observe that for each \( i \in [q] \), the vertices
of \( \hat{Q}_i \) except \( t \) lie in the set \( C_j \), for some \( j \in [q] \), and this correspondence between \( \{\hat{Q}_i\}_{i \in [q]} \) and

41
\{C_j\}_{j\in[q]} is a bijection defined by \(\pi\) on \(I\) and by the identity permutation on \([q] \setminus I\). Since the sets \(\{C_j\}_{j\in[q]}\) are disjoint, for any \(i \neq i' \in [q]\), we get that the paths \(\hat{Q}_i\) and \(\hat{Q}_{i'}\) share the only common vertex \(t\).

It remains to verify that for each \(i \neq i' \in [q]\), \(\hat{P}_i\) shares no common vertices with \(\hat{Q}_{i'}\). For \(i' \in [q] \setminus I\), \(\hat{Q}_{i'} = P_{i'}^T\), which is a suffix of \(P_i\), and this path cannot intersect \(\hat{P}_i\) which is a prefix of \(P_i\); thus in the following we assume \(i' \in I\). Assume the contrary, then there is a vertex \(u\) on \(\hat{P}_i\) that belongs to \(Q_{\pi(i')}\), and is located on \(Q_{\pi(i')}\) closer to \(t\) than \(t_i^T\), which is the starting vertex of \(\hat{Q}_{i'}\). Observe that the rule \texttt{Clear} has never been applied to \(Q_{\pi(i')}\); otherwise \(Q_{\pi(i')}\) would not be active in \(S^T\). Thus, there is a state \(S^h\) where \(t_i^h = u\), since any application of the rules to \(t_i^h\) for any \(h' \in [T]\) either leaves this vertex in place or moves it to the next vertex of \(P_i\) belonging to an active \(Q_j\) for some \(j \in [q]\). Now observe that no application of the rule \texttt{Push} makes the closest vertex to \(t\) on \(Q_{\pi(i')} - \{t\}\) among \(\{t_j^h\}_{j\in[q]}\) farther, by definition of \texttt{Push}. However, we get that \(t_i^h\) is closer to \(t\) on \(Q_{\pi(i')}\) than \(t_i^T\), which is the only vertex of \(\{t_j^h\}_{j\in[q]}\) on \(Q_{\pi(i')} - \{t\}\). This is a contradiction to the assumption that \(\hat{P}_i\) and \(\hat{Q}_{i'}\) intersect.

**Proof of Claim 6.1.** Assume the contrary, that \(t_i^T = t\) for each \(i \in [q]\). Since the paths \(\{P_i^T\}_{i\in[q]}\) are non-empty, \(T > 1\). Thus, the state \(S^{T-1}\) is defined and \(S^T\) is obtained by applying a rule to \(S^{T-1}\). First, observe that this rule could not have been \texttt{Push}, as it assumes there exist distinct \(t_i^{T-1}\) and \(t_j^{T-1}\) on \(Q_j - \{t\}\) for some \(i, i', j \in [q]\), and only moves \(t_i^{T-1}\) away while keeping \(t_j^T\) on \(Q_j - \{t\}\). Therefore, \texttt{Clear} has been applied to \(S^{T-1}\), replacing \(t_i^{T-1} = v_j\) by \(t_j^T = t\), for some \(i, j \in [q]\). Now, we claim that there is another \(i'' \in [q]\), \(i'' \neq i\) such that \(t_i^{T-1} \in V(Q_j) \setminus \{t\}\). Indeed, by the starting assumption of the proof, there exists \(i''\) such that \(P_{i''}^T\) and \(Q_j\) share an internal vertex \(u\). Since \(Q_j\) is active until the last step, and since the application of any rule moves \(t_{i''}^h\) to the next vertex of \(P_{i''}\) intersecting some active \(Q_j\), there exists a step \(h \in [T - 1]\) where \(t_{i''}^h = u\). Before the step \(T - 1\), only the rule \texttt{Push} could have been applied to \(Q_j\), and an application of this rule never makes the intersection \(\{t_{i''}^{h'}\}_{i'' \in [q]} \cap (Q_j \setminus \{v_j, t\})\) empty. Therefore there exists \(i'' \in [q]\) such that \(t_i^{T-1} \in Q_j \setminus \{v_j, t\}\). This contradicts the assumption that \(t_i^T = t\) since the application of \texttt{Clear} to \(Q_j\) moves \(t_i^{T-1}\) to the next vertex on \(P_{i''}\) on an active \(Q_j\); this will not take \(t_i^T\) farther than \(v_{i''}\) along \(P_{i''}\). \qed

Before we move to the proof of the main theorem, we note that the basic idea of random separation is to exploit random colorings of the vertex set. We, on the other hand, are first and foremost looking for a deterministic algorithm; the standard approach would be to enumerate a sufficiently “expressive” set of colorings, instead of trying a pre-set number of random colorings. Unfortunately, the existing results on derandomization of random separation algorithms cannot be applied directly, as normally random separation is considered for constant number of sets; most often two. Thus in the next lemma we directly construct a suitable family of functions by using the standard tool of perfect hash families, given by the classical result of Naor, Schulman, and Srinivasan [39] (we refer to [14, Chapter 5] for the detailed introduction to the concept). For integers \(n\) and \(k\), an \((n, k)\)-perfect hash family \(\mathcal{F}\) is a family of functions from \([n]\) to \([k]\) such that for each set \(S \subset [n]\) of size \(k\) there exists \(f \in \mathcal{F}\) that acts on \(S\) injectively. We are now ready to state our derandomization lemma.

**Lemma 16.** For an \(n\)-element set \(U\) and \(q\) integers \(k_1, \ldots, k_q\), \(\sum_{i=1}^q k_i = \ell\) there exists a family of functions \(\mathcal{F}\) of size \(q^\Theta(\ell)\log n\) mapping \(U\) to \([1, \ldots, q]\) with the following property. For any disjoint
Proof. First, construct an \((n, \ell)\)-perfect hash family \(\mathcal{H}\) of size \(e^{\ell O(\log \ell)} \log n\) in time \(e^{\ell O(\log \ell)} n \log n\) by the result of Naor, Schulman, and Srinivasan [39]. For every \(h \in \mathcal{H}\) and a partition \([\ell] = I_1 \cup \ldots \cup I_q\) such that \(|I_i| = k_i\) for \(i \in [q]\), add a function \(f_{I_1, \ldots, I_q}^h\) to \(\mathcal{F}\). The function acts as follows: for any \(x \in U\), \(f_{I_1, \ldots, I_q}^h(x) = i\) if \(h(x) \in I_i\).

We now show that \(\mathcal{F}\) defined above satisfies the conditions of the lemma. Fix the subsets \(A_1, \ldots, A_q, A \subseteq U\), denote \(A = A_1 \cup \ldots \cup A_q\), \(|A| = \ell\). By the definition of an \((n, \ell)\)-perfect hash family, there exists \(h \in \mathcal{H}\) such that the images \(h(x)\) are distinct for all \(x \in A\). For each \(i \in [q]\), define \(I_i\) to be the set of indices that \(h\) assigns to \(A_i\). By definition, \(f_{I_1, \ldots, I_q}^h(x) = i\) if \(h(x) \in I_i\), and \(h(x) \in I_i\) if and only if \(x \in A_i\). It only remains to bound the number of partitions \(I_1, \ldots, I_q\).

**Claim 6.3.** The number of partitions of \([\ell]\) into disjoint subsets \(I_1, \ldots, I_q\) with \(|I_i| = k_i\) for \(i \in [q]\) is \(O(\ell \cdot q^\ell)\).

**Proof.** The number of partitions is equal to the multinomial coefficient

\[
\binom{\ell}{k_1, k_2, \ldots, k_q} = \frac{\ell!}{k_1! k_2! \cdots k_q!} \leq e^{\sqrt{\ell}} \left(\frac{\ell}{e}\right)^{k_1} \left(\frac{\ell}{e}\right)^{k_2} \cdots \left(\frac{\ell}{e}\right)^{k_q} = O\left(\ell \cdot \left(\frac{\ell}{e}\right)^{k_1 \cdot k_2 \cdots k_q}q^\ell\right),
\]

by Stirling’s formula. We now argue that \(k_1 \cdot k_2 \cdots k_q \geq \left(\frac{\ell}{q}\right)^\ell\), which immediately implies that the desired number of partitions is bounded by \(O\left(\ell \cdot \left(\frac{\ell}{e}\right)^{\ell}/q^\ell\right)\). For that, we observe that the function \(f(x) = x \log x\) is strictly convex on \(x \geq 1\), since \((x \log x)' = (1 + \log x)\), thus \(x \log x + y \log y \geq 2 \cdot \left(\frac{x+y}{2}\right) \cdot \log \left(\frac{x+y}{2}\right)\), and \(x^y \cdot y^x \geq (\frac{x+y}{2})^{\frac{x+y}{2}}\) for any \(x, y \geq 1\), where the equality only holds if \(x = y\). Now, consider the function \(h(x_1, x_2, \ldots, x_q) = x_1^x \cdot x_2^x \cdots x_q^x\) defined on the polytope \(K \subseteq \mathbb{R}^q\) bounded by \(x_1 \geq 1, \ldots, x_q \geq 1, \sum_{i=1}^q x_i = \ell\). Since \(h\) is continuous on \(K\) and \(K\) is compact, \(h\) attains its minimum in \(K\). Assume that \(h\) achieves its minimum on \(x_1, \ldots, x_q \in K\) with \(x_i \neq x_j\) for some \(i, j \in [q]\). Then by the above, \(h(x_1, x_2, \ldots, x_j, \ldots, x_q) > h(x_1, \ldots, x_i+x_j, \ldots, x_{i+1}, \ldots, x_q)\); the tuple on the right-hand side still belongs to \(K\). Thus, such a \(x_1, \ldots, x_q\) cannot achieve the minimum, and \(h\) is minimized at the only point with equal coordinates, \((\ell/q, \ldots, \ell/q) \in K\). Since \((k_1, \ldots, k_q) \in K\), the claim is done. \(\square\)

The bound on the size of \(\mathcal{F}\) now follows directly from **Claim 6.3**: \(|\mathcal{F}| = |\mathcal{H}| \cdot O(\ell \cdot q^\ell) = q^{O(\ell)} \cdot n^{O(1)}\).

Finally, with Lemma 15 and Lemma 16 at hand, we move to the proof of Theorem 2 itself.

**Proof of Theorem 2.** First, we observe that finding an \((S, T)\)-linkage of order \(p\) and total length at least \(k\) is equivalent to finding an \((s, t)\)-linkage of order \(p\) and total length at least \((k+2)\), where moreover \(s \neq t\) and \(s\) is not adjacent to \(t\). Indeed, consider the digraph \(G'\) that is a copy of \(G\) with two new vertices \(s\) and \(t\), where \(N_{G'}(s) = S\) and \(N_{G'}(t) = T\). Then, any directed \((s, t)\)-linkage of order \(p\) and length \(k+2\) in \(G'\) induces a directed \((S, T)\)-linkage of order \(p\) and length \(k\) in \(G\) by removing \(s\) and \(t\), and vice versa. Thus for the rest of the proof we assume that the task is to find an \((s, t)\)-linkage of order \(p\) and total size at least \(k\), \(s \neq t\), and \(s\) is not adjacent to \(t\). We now describe two separate subroutines of our algorithm, tailored for different cases of the maximum length of the path in the target \((s, t)\)-linkage. The **short case** succeeds if there is an \((s, t)\)-linkage.
where all paths have less than $2k$ internal vertices, and the **main case** succeeds otherwise (the proof of correctness follows after the description of the algorithm).

**Short case.** For each $i \in [p]$, we branch over the number of internal vertices $k_i$ of the $i$-th path in the target linkage, $1 \leq k_i < 2k$. If $\sum_{i=1}^{p} k_i < k - 2$, we disregard the choice of $\{k_i\}_{i=1}^{p}$ and proceed to the next branch. Otherwise, consider a function family $\mathcal{F}$ given by an invocation of Lemma 16 with $q = p$ and the current values of $k_1, \ldots, k_p$. Branch over the choice of $f \in \mathcal{F}$ and denote by $C_1, C_2, \ldots, C_p$ the vertices colored by the respective colors via $f$. For each $i \in [p]$, we use a deterministic algorithm for finding a directed $(s, t)$-path with exactly $k_i$ internal vertices in the graph $G[C_i \cup \{s, t\}]$ in time $2^{O(h(k_i))} \cdot n^{O(1)}$. The fastest-known such algorithm is the algorithm of Zehavi [48] running in time $O(2.597^{k_i}) \cdot n^{O(1)}$.

If for some choice of $\{k_i\}_{i=1}^{p}$ and $f$ the desired collection of paths is found, the algorithm returns it. If no branch succeeds, the algorithm reports a no-instance.

We now argue for correctness of the algorithm above. Since the paths are internally-disjoint by construction and $\sum_{i=1}^{p} k_i \geq k - 2$, if the algorithm returns a collection of paths, they clearly form a solution. In the other direction, fix a solution induced by directed $(s, t)$-paths $P_1^*, \ldots, P_p^*$, where for each $i$ the $i$-th path contains exactly $k_i < 2k$ internal vertices, and consider the respective branch of the algorithm above. If for each $i \in [p]$ the set $C_i$ contains the internal vertices of the $i$-th path, the algorithm succeeds, as $G[C_i \cup \{s, t\}]$ contains an $(s, t)$-path with exactly $k_i$ internal vertices. Denote by $A_i$ the set of internal vertices of $P_i^*$, for each $i \in [p]$. Lemma 16 guarantees that there exists $f \in \mathcal{F}$ that colors each $A_i$ in color $i$, concluding the proof of correctness in this case.

---

**Algorithm 1:** Main case of the algorithm in Theorem 2.

```plaintext
1 for $q' = 1, \ldots, p$ do
2    Invoke Lemma 16 with $q = p + 1$, $k_1 = \cdots = k_{q'} = k$, $k_{q'+1} = \cdots = k_p = 2k$, and $k_{p+1} = k$, to obtain the function family $\mathcal{F}$;
3    foreach $f \in \mathcal{F}$ do
4        Denote by $C_0, C_1, \ldots, C_p$ the vertices colored by the respective colors via $f$;
5        for $i = 1, \ldots, p$ do
6            foreach $v_i \in C_i$ at distance $k$ from $t$ in $G_i = G[C_i \cup \{t\}]$ do
7                Find a shortest $(v_i, t)$-path $Q_i$ in $G_i$;
8                Find paths $P_1, P_2, \ldots, P_p$ in $G - (V(Q_i) \setminus \{v_i, t\})$, where $P_i$ is an $(s, v_i)$-path and for each $j \neq i$, $P_j$ is an $(s, t)$-path, such that no two paths share a vertex except for $s$ and $t$, and $P_i$ does not contain $t$;
9                if such paths $P_1, P_2, \ldots, P_p$ exist then
10                   Set $P_i = P_i \circ Q_i$;
11                   return the paths $P_1, P_2, \ldots, P_p$;
12                end
13            end
14        end
15    end
16 end
```

**Main case.** The basic procedure is given in Algorithm 1. Observe that the task in Line 8 can be easily reduced to an instance of network flow, thus the whole procedure given in Lines 4

---

$^3$While the result in [48] is stated for finding an arbitrary path of certain length, it could be easily adjusted to finding an $(s, t)$-path.
to 14 runs in polynomial time. If no iteration returns a collection of paths, the algorithm reports a no-instance.

It is easy to observe that if Algorithm 1 returns a collection of paths, then these paths constitute a solution to the given instance. Indeed, by construction, $P_1, \ldots, P_p$ are internally-disjoint $(s, t)$-paths. The length of $Q_i$ is exactly $k$, since the path $P_i$ contains $Q_i$ as a subpath, the $(s, t)$-linkage given by the paths $P_1, \ldots, P_p$ is of order $p$ and length at least $k$. It remains to verify that if there exists an $(s, t)$-linkage of order $p$ where there is a path with at least $2k$ internal vertices, then our algorithm successfully returns a collection of paths for some choice of $f$. Denote the $p$-many $(s, t)$-paths that form a solution of minimum total length by $P_1^*, \ldots, P_p^*$, assuming that the paths are ordered from longest to shortest. In particular, $|V(P_i^*) \setminus \{s, t\}| \geq 2k$. Denote by $q'$ the maximum index such that $|V(P_q^*) \setminus \{s, t\}| \geq 2k$, $1 \leq q' \leq p$.

We say that the coloring $C_0, C_1, \ldots, C_p$ agrees with the paths $P_1^*, \ldots, P_p^*$ if the following holds:

(i) for each $i \in [q']$, the last $k$ internal vertices of $P_i^*$ belong to $C_i$,

(ii) the first $k$ vertices of $P_i^*$ belong to $C_0$,

(iii) for each $j \in [p] \setminus [q']$, all internal vertices of $P_j^*$ belong to $C_j$.

For $i \in [q']$, denote by $A_i$ the last $k$ internal vertices of $P_i^*$; by $A_{p+1}$ the first $k$ vertices of $P_1^*$; for $j \in [p] \setminus [q']$, denote by $A_j$ all internal vertices of $P_j^*$ plus arbitrary vertices not yet part of any $A_i$ so that $|A_j| = 2k$. By Lemma 16, there exists $f \in \mathcal{F}$ that induces a coloring $C_1, \ldots, C_{p+1}$ with $A_i \subseteq C_i$ for each $i \in [p + 1]$, meaning that this coloring agrees with the solution $P_1^*, \ldots, P_p^*$. In the remainder of the proof, we argue that for this choice of $f$ Algorithm 1 outputs a solution.

For $i \in [p]$, let $v_i$ be the vertex of $P_i^*$ at distance exactly $k$ from $t$ along the path. Since the coloring agrees with the solution, the last $k$ internal vertices of $P_i^*$ belong to $C_i$, including $v_i$. If $v_i$ is at distance $k$ from $t$ in $G_i$, denote by $Q_i$ the shortest $(v_i, t)$-path in $G_i$ that the algorithm finds on Line 7; otherwise denote by $Q_i$ an arbitrary shortest $(v_i, t)$-path in $G_i$. We now apply Lemma 15 to the paths $P_1^*, \ldots, P_p^*$ in the graph $G$ with selected disjoint vertex subsets $C_1, \ldots, C_q$, with selected vertices $v_1, \ldots, v_q$ and paths $Q_1, \ldots, Q_q$. By the lemma, there exist internally-disjoint $(s, t)$-paths $P_1', \ldots, P_q'$, such that for each $i \in [q]$, $P_i'$ is either $P_i^*$ or a concatenation of a prefix $\widehat{P}_i$ of $P_i^*$ not extending beyond $v_i$, and a suffix $\widehat{Q}_i$ of $Q_i$ for some $j \in [q]$: moreover, at least one of the paths is of the second type.

First, we claim that the paths $P_1', \ldots, P_q', P_{q+1}', \ldots, P_p'$ together form an $(\{s\}, \{t\})$-linkage of length at least $k$. Indeed, the paths $P_1', \ldots, P_p'$ are internally-disjoint from the beginning; thus for each $q < i < j \leq p$, $P_i'$ and $P_j'$ are immediately internally-disjoint in case $P_i' = P_i^*$. In case $P_i'$ is a concatenation of $\widehat{P}_i$ and $\widehat{Q}_i$, the path $\widehat{P}_i$ again does not share a vertex with $P_j'$ as a prefix of $P_j^*$, except for $s$. The suffix $\widehat{Q}_i - \{t\}$ on the other hand is fully contained in some $C_{i'}$, $i' \in [q]$, while $P_{i'}^* - \{s, t\}$ is contained in $C_j$ by the property (iii) of the coloring. Finally, for the length observe that the path $P_i'$ contains the first $k$ internal vertices of the path $P_i^*$ since they belong to the set $C_0$ disjoint from $C_i$ for any $i > 0$.

Consider now a path $P_i'$ that is not $P_i^*$ but a concatenation, $\widehat{P}_i \circ \widehat{Q}_i$. Since the paths $P_1^*, \ldots, P_p^*$ come from a solution of smallest total length, it cannot be that $P_i'$ is shorter than $P_i^*$. On the other hand, the length of $\widehat{Q}_i$ is at most $k$, which is the length of the suffix of $P_i^*$ from $v_i$ to $t$, and $\widehat{P}_i$ is at most as long as the prefix of $P_i^*$ from $s$ to $v_i$. Thus it has to be that $\widehat{P}_i$ is exactly the prefix of $P_i^*$ from $s$ to $v_i$, and $\widehat{Q}_i$ is $Q_i$; additionally, $Q_i$ is then of length exactly $k$, thus $v_i$ is at distance $k$ from $t$ in $G_i$, and $Q_i$ is the shortest path chosen by the algorithm in Line 7.
We now claim that the existence of \((s, t)\)-linkage \(P'_1, \ldots, P'_{q}, P'_{q+1}, \ldots, P'_p\) implies that Line 8 is executed successfully for the respective choice of \(v_i\). Indeed, denote \(P_i = \tilde{P}_i\), for each \(i' \in [q]\), \(i' \neq i\), denote \(P_{i'} = P'_{i'}\), and for each \(j \in [p] \setminus [q]\), denote \(P_j = P'_j\). This collection of paths is of form required by the conditions in Line 8, and thus the algorithm of Line 8 returns a suitable collection of paths as well (not necessarily the same). This concludes the proof of correctness for Main case.

Finally, observe that the running time of both cases is dominated by the invocation of Lemma 16 with \(q = p + 1\) and \(\ell \leq 2k(p + 1)\), resulting in the total running time of \(p^\Theta(kp) \cdot n^{\Theta(1)}\).

7 Conclusion

We conclude with several concrete open questions. The first question is about derandomizing Theorem 1, even for the case when \(p = 1\). The algorithm in Theorem 1 is based on DeMillo-Lipton-Schwartz-Zippel lemma for polynomial identity testing, and therefore we do not expect to derandomize it using similar techniques [27, 38]. However, similarly to Theorem 2, we do not exclude that other methods could result in (maybe slower) deterministic algorithms. We are not aware of any deterministic and FPT in \(k\) algorithm for Maximum Colored Path.

The second question is about the Disjoint Paths problem. Here for a given set of pairs of terminal vertices \((s_1, t_1), \ldots, (s_r, t_r)\), the problem is to decide whether there are vertex-disjoint \((s_i, t_i)\)-paths, \(i \in [r]\). The problem is FPT parameterized by \(r\) by the seminal algorithm of Robertson and Seymour [43]. A natural extension of this problem would be on colorful graphs, where we want the disjoint paths to collect at least \(k\) colors. We do not know whether the colored variant of the problem is FPT parameterized by \(k\) even for 2-Disjoint Paths, that is for \(r = 2\).

The third question concerns extending Theorem 3, where we demand matroid \(M\) to be represented as a matrix over a finite field of order \(q\). The natural question here is whether there is an FPT algorithm for \(k\)-ranked \((s, t)\)-path (and more generally, for \(k\)-ranked \((S, T)\)-linkage of order \(p\)) in frameworks \((G, M)\), where \(M\) is a linear matroid represented as a matrix over rationals. We also ask what is the complexity of this problem when \(M\) is given by an independence oracle.

As was shown by Jensen and Korte [26], various matroid problems have unconditional complexity lower bounds asserting that they admit no algorithms where the number of oracle calls is bounded by a polynomial on the size of the matroid ground set. For example, this concerns the classical Matroid Parity problem that can be solved in polynomial time on linear matroids as it was shown by Lovász (see, e.g., [36]). It is natural to ask whether such a lower bound can be shown for \(k\)-ranked \((s, t)\)-path.

The last concrete question is about Longest \((s, t)\)-Path and Longest Cycle. Our algorithm implies the first \(2^k n^{\Theta(1)}\) time algorithms for these problems, and the dependency on \(k\) in the time complexity of our algorithm is unlikely to be improved in the general colored case. However, it remains an interesting open problem whether Longest \((s, t)\)-Path or Longest Cycle could be solved in time \((2 - \varepsilon)^k n^{\Theta(1)}\) for some \(\varepsilon > 0\), especially keeping in mind that \(k\)-Path admits an \(1.66^k n^{\Theta(1)}\) time algorithm [3].

References


48


