Parameterized Complexity of the List Coloring Reconfiguration Problem with Graph Parameters*†

Tatsuhiko Hatanaka[1], Takehiro Ito[2], and Xiao Zhou[3]

1 **Graduate School of Information Sciences, Tohoku University, Sendai, Japan**
hatanaka@ecei.tohoku.ac.jp
2 **Graduate School of Information Sciences, Tohoku University, Sendai, Japan**
takehiro@ecei.tohoku.ac.jp
3 **Graduate School of Information Sciences, Tohoku University, Sendai, Japan**
zhou@ecei.tohoku.ac.jp

Abstract

Let G be a graph such that each vertex has its list of available colors, and assume that each list is a subset of the common set consisting of k colors. For two given list colorings of G, we study the problem of transforming one into the other by changing only one vertex color assignment at a time, while at all times maintaining a list coloring. This problem is known to be PSPACE-complete even for bounded bandwidth graphs and a fixed constant k. In this paper, we study the fixed-parameter tractability of the problem when parameterized by several graph parameters. We first give a fixed-parameter algorithm for the problem when parameterized by k and the modular-width of an input graph. We next give a fixed-parameter algorithm for the shortest variant which computes the length of a shortest transformation when parameterized by k and the size of a minimum vertex cover of an input graph. As corollaries, we show that the problem for cographs and the shortest variant for split graphs are fixed-parameter tractable even when only k is taken as a parameter. On the other hand, we prove that the problem is W[1]-hard when parameterized only by the size of a minimum vertex cover of an input graph.

1998 ACM Subject Classification G.2.2 Graph Theory

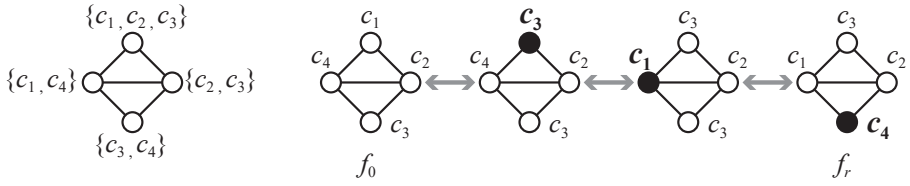
Keywords and phrases combinatorial reconfiguration, fixed-parameter tractability, graph algorithm, list coloring, W[1]-hardness

Digital Object Identifier 10.4230/LIPIcs.MFCS.2017.51

1 Introduction

Recently, the framework of *reconfiguration* [14] has been extensively studied in the field of theoretical computer science. This framework models several situations where we wish to find a step-by-step transformation between two feasible solutions of a combinatorial (search) problem such that all intermediate solutions are also feasible and each step respects a fixed reconfiguration rule. This reconfiguration framework has been applied to several well-studied combinatorial problems. (See a survey [18].)

* This work is partially supported by JST CREST Grant Number JPMJCR1402, and by JSPS KAKENHI Grant Numbers JP16J02175, JP16K00003, and JP16K00004.
† A full version of the paper is available at https://arxiv.org/abs/1705.07551.



(a) G, L (b)

Figure 1 A reconfiguration sequence between two L-colorings f_0 and f_t of G.

1.1 Our problem

In this paper, we study a reconfiguration problem for list (vertex) colorings in a graph, which was introduced by Bonsma and Cereceda [3].

Let $C = \{c_1, c_2, \ldots, c_k\}$ be the set of k colors, called the *color set*. A (proper) *k-coloring* of a graph $G = (V, E)$ is a mapping $f\colon V \to C$ such that $f(v) \neq f(w)$ for every edge $vw \in E$. In *list coloring*, each vertex $v \in V$ has a set $L(v) \subseteq C$ of colors, called the *list of* v; sometimes, the list assignment $L\colon V \to 2^C$ itself is called a *list*. Then, a k-coloring f of G is called an *L-coloring* of G if $f(v) \in L(v)$ holds for every vertex $v \in V$. Therefore, a k-coloring of G is simply an L-coloring of G when $L(v) = C$ holds for every vertex v of G, and hence L-coloring is a generalization of k-coloring. Figure 1(b) illustrates four L-colorings of the same graph G in Figure 1(a); the color assigned to each vertex is attached to the vertex.

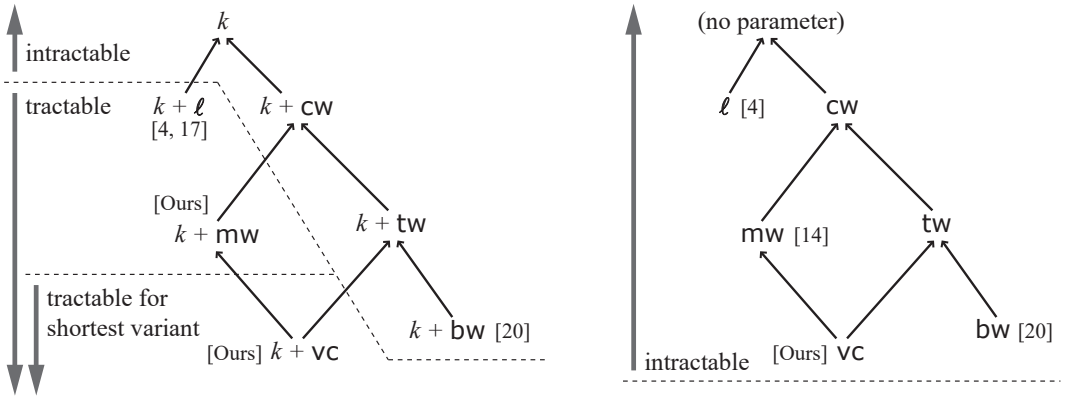
In the reconfiguration framework, two L-colorings f and f' of a graph $G = (V, E)$ are said to be *adjacent* if $|\{v \in V\colon f(v) \neq f'(v)\}| = 1$ holds, that is, f' can be obtained from f by recoloring exactly one vertex. A sequence $\langle f_0, f_1, \ldots, f_\ell \rangle$ of L-colorings of G is called a *reconfiguration sequence* between f_0 and f_ℓ (of *length* ℓ) if f_{i-1} and f_i are adjacent for each $i \in \{1, 2, \ldots, \ell\}$. Two L-colorings f and f' are *reconfigurable* if there exists a reconfiguration sequence between them. The LIST COLORING RECONFIGURATION problem is to determine whether two given L-colorings f_0 and f_t are reconfigurable, or not. Figure 1 shows an example of a yes-instance of LIST COLORING RECONFIGURATION, where the vertex whose color assignment was changed from the previous one is depicted by a black circle.

1.2 Known and related results

LIST COLORING RECONFIGURATION is one of the most well-studied reconfiguration problems, as well as COLORING RECONFIGURATION which is a special case of the problem such that $L(v) = \{c_1, c_2, \ldots, c_k\}$ holds for every vertex v. These problems have been studied intensively from various viewpoints [1, 2, 3, 4, 6, 7, 9, 13, 15, 19] including the generalizations [5, 20].

Bonsma and Cereceda [3] proved that COLORING RECONFIGURATION is PSPACE-complete even for bipartite graphs and any fixed constant $k \geq 4$. On the other hand, Cereceda et al. [7] gave a polynomial-time algorithm solving COLORING RECONFIGURATION for any graph and $k \leq 3$; the algorithm can be applied to LIST COLORING RECONFIGURATION, too. In particular, the former result implies that there is no fixed-parameter algorithm for COLORING RECONFIGURATION (and hence LIST COLORING RECONFIGURATION) when parameterized by only k under the assumption of P $\neq$ PSPACE.

Bonsma et al. [4] and Johnson et al. [15] independently developed a fixed-parameter algorithm to solve COLORING RECONFIGURATION when parameterized by $k + \ell$, where ℓ is the upper bound on the length of reconfiguration sequences, and again their algorithms can be applied to LIST COLORING RECONFIGURATION. In contrast, if COLORING RECONFIGURATION is parameterized only by ℓ, then it is W[1]-hard when k is an input [4] and does not admit a polynomial kernelization when k is fixed unless the polynomial hierarchy collapses [15].



(a) parameters with k (b) parameters without k

Figure 2 All results (including known ones) for LIST COLORING RECONFIGURATION from the viewpoint of parameterized complexity, where cw, tw, bw, mw, and vc are the upper bounds on the cliquewidth, treewidth, bandwidth, modular-width, and the size of a minimum vertex cover of an input graph, respectively.

Hatanaka et al. [13] proved that LIST COLORING RECONFIGURATION is PSPACE-complete even for complete split graphs, whose modular-width is zero. Wrochna [19] proved that LIST COLORING RECONFIGURATION is PSPACE-complete even when k and the bandwidth of an input graph are bounded by some constant; thus the treewidth and the cliquewidth of an input graph are also bounded.

1.3 Our contribution

To the best of our knowledge, known algorithmic results mostly employed the length ℓ of reconfiguration sequences as a parameter [4, 15], and no fixed-parameter algorithm is known when parameterized by graph parameters. Therefore, we study LIST COLORING RECONFIGURATION when parameterized by several graph parameters, and paint an interesting map of graph parameters which shows the boundary between fixed-parameter tractability and intractability. Our map is Figure 2 which shows both known and our results, where an arrow $\alpha \to \beta$ indicates that the parameter α is "stronger" than β, that is, β is bounded if α is bounded. (For relationships of parameters, see, e.g., [10, 16].)

More specifically, we first give a fixed-parameter algorithm solving LIST COLORING RECONFIGURATION when parameterized by k and the modular-width mw of an input graph. (The definition of modular-width will be given in Section 2.1.) Note that, according to the known results [3, 13], we cannot construct a fixed-parameter algorithm for general graphs when only one of k and mw is taken as a parameter under the assumption of P $\neq$ PSPACE. However, as later shown in Corollary 4, our algorithm implies that the problem is fixed-parameter tractable for cographs even when only k is taken as a parameter.

We then consider the shortest variant which computes the length of a shortest reconfiguration sequence (i.e., the minimum number of recoloring steps) for a yes-instance of LIST COLORING RECONFIGURATION, and show that it admits a fixed-parameter algorithm when parameterized by k and the size of a minimum vertex cover of an input graph. Moreover, as a corollary, we show that the shortest variant is fixed-parameter tractable for split graphs even when only k is taken as a parameter.

Finally, we prove that LIST COLORING RECONFIGURATION is $W[1]$-hard when parameterized only by the size of a minimum vertex cover of an input graph.

Due to the page limitation, several proofs are omitted from this extended abstract.

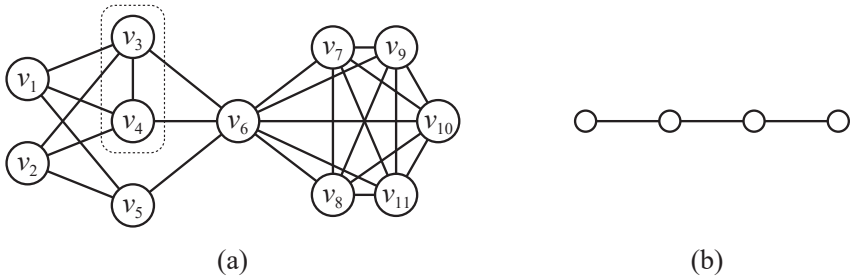


Figure 3 (a) An example of module and (b) a prime.

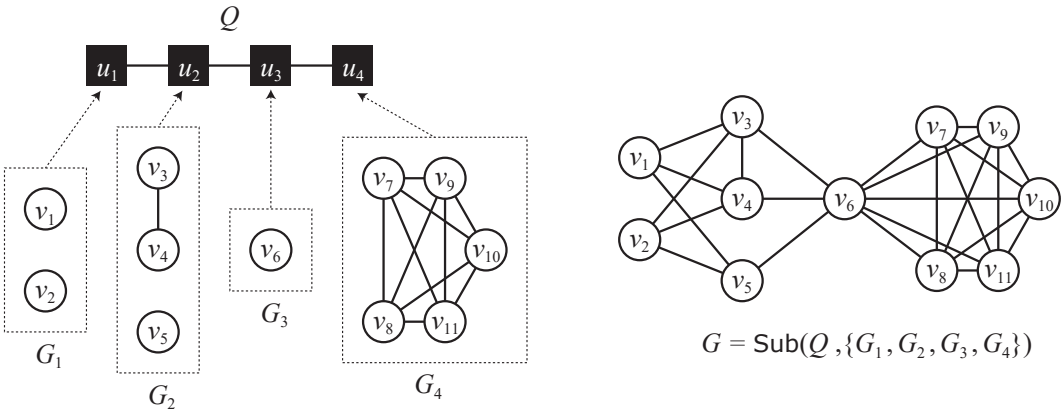


Figure 4 An example of substitution operation.

2 Preliminaries

We assume without loss of generality that graphs are simple and connected. Let $G = (V, E)$ be a graph with vertex set V and edge set E; we sometimes denote by $V(G)$ and $E(G)$ the vertex set and the edge set of G, respectively. For a vertex v in G, we denote by $N(G, v)$ the neighborhood $\{w \in V : vw \in E\}$ of v in G. For a vertex subset $V' \subseteq V$, we denote by $G[V']$ the subgraph of G induced by V', and denote $G \setminus V' = G[V(G) \setminus V']$. For a subgraph H of G, we denote $G \setminus H = G \setminus V(H)$. Let $\omega(G)$ be the size of a maximum clique of G. We have the following simple observation.

▶ **Observation 1.** *Let G be a graph with a list $L : V(G) \to 2^C$. If G has an L-coloring, then $\omega(G) \leq |C|$.*

A graph is *split* if its vertex set can be partitioned into a clique and an independent set. A graph is a *cograph* (or a P_4*-free* graph) if it contains no induced path with four vertices.

2.1 Modules and modular decomposition

A *module* of a graph $G = (V, E)$ is a vertex subset $M \subseteq V$ such that $N(G, v) \setminus M = N(G, w) \setminus M$ for every two vertices v and w in M. In other words, the module M is a set of vertices whose neighborhoods in $G \setminus M$ are the same. For example, the graph in Figure 3(a) has a module $M = \{v_3, v_4\}$ for which $N(G, v_3) \setminus M = N(G, v_4) \setminus M = \{v_1, v_2, v_6\}$ holds. Note that the vertex set V of G, the set consisting of only a single vertex, and the empty set $\emptyset$ are all modules of G; they are called *trivial*. A graph G is a *prime* if all of its modules are trivial; for an example, see Figure 3(b).

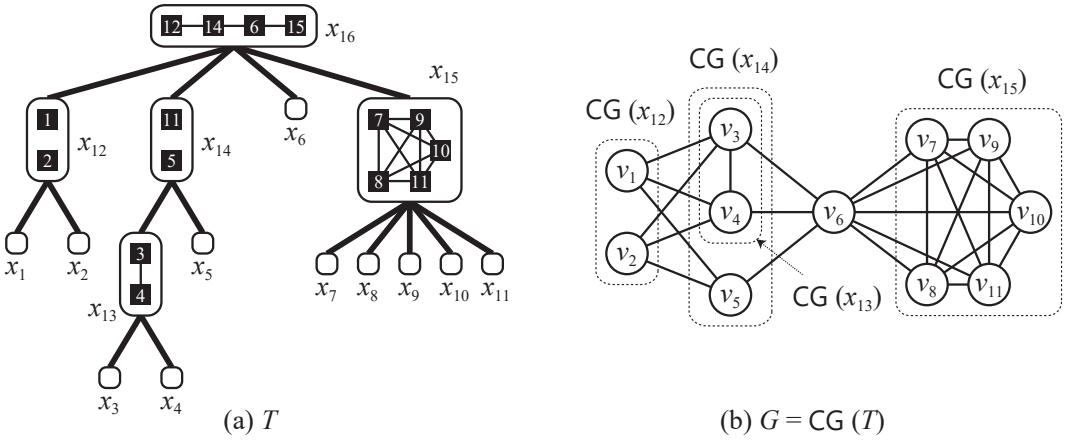


Figure 5 (a) A substitution tree T for (b) a graph G.

We now introduce the notion of modular decomposition, which was first presented by Gallai in 1967 as a graph decomposition technique [11]. For a survey, see, e.g., [12].

We first define the *substitution* operation, which constructs one graph from more than one graphs. Let Q be a graph, called a *quotient graph*, consisting of p (≥ 2) nodes $u_1, u_2, \ldots, u_p$, and let $\mathcal{F} = \{G_1, G_2, \ldots, G_p\}$ be a family of vertex-disjoint graphs such that G_i corresponds to u_i for every $i \in \{1, 2, \ldots, p\}$. The *$Q$-substitution* of $\mathcal{F}$, denoted by $\mathsf{Sub}(Q, \mathcal{F})$, is the graph which is obtained by taking a union of all graphs in $\mathcal{F}$ and then connecting every pair of vertices $v \in V(G_i)$ and $w \in V(G_j)$ by an edge if and only if u_i and u_j are adjacent in Q. That is, the vertex set of $\mathsf{Sub}(Q, \mathcal{F})$ is $\bigcup\{V(G_i) \colon G_i \in \mathcal{F}\}$, and the edge set of $\mathsf{Sub}(Q, \mathcal{F})$ is the union of $\bigcup\{E(G_i) \colon G_i \in \mathcal{F}\}$ and $\{vw \colon v \in V(G_i), w \in V(G_j), u_iu_j \in E(Q)\}$. (See Figure 4 as an example.)

A *substitution tree* is a rooted tree T such that each non-leaf node $x \in V(T)$ is associated with a quotient graph $\mathsf{Q}(x)$ and has $|V(\mathsf{Q}(x))|$ child nodes. For each node $x \in V(T)$, we can recursively define the *corresponding graph* $\mathsf{CG}(x)$ as follows: If x is a leaf, $\mathsf{CG}(x)$ consists of a single vertex. Otherwise, let $y_1, y_2, \ldots, y_p$ be $p = |V(\mathsf{Q}(x))|$ children of x, then $\mathsf{CG}(x) = \mathsf{Sub}(\mathsf{Q}(x), \{\mathsf{CG}(y_1), \mathsf{CG}(y_2), \ldots, \mathsf{CG}(y_p)\})$. For the root r of T, $\mathsf{CG}(r)$ is called the *corresponding graph of* T, and we denote $\mathsf{CG}(T) := \mathsf{CG}(r)$. We say that T is a substitution tree *for* a graph G if $\mathsf{CG}(T) = G$, and refer to a *node* in T in order to distinguish it from a vertex in G. Figure 5(a) illustrates a substitution tree for the graph G in Figure 5(b); each leaf x_i, $i \in \{1, 2, \ldots, 11\}$, corresponds to the subgraph of G consisting of a single vertex v_i. We note that the vertex set $V(\mathsf{CG}(x))$ of each corresponding graph $\mathsf{CG}(x)$, $x \in V(T)$, forms a module of $\mathsf{CG}(T)$.

A *modular decomposition tree* T (an *MD-tree* for short) for a graph G is a substitution tree for G which satisfies the following three conditions:

- Each node $x \in V(T)$ applies to one of the following three types:
  - a *series* node, whose quotient graph $\mathsf{Q}(x)$ is a complete graph;
  - a *parallel* node, whose quotient graph $\mathsf{Q}(x)$ is an edge-less graph; and
  - a *prime* node, whose quotient graph $\mathsf{Q}(x)$ is a prime with at least four vertices.
- No edge connects two series nodes.
- No edge connects two parallel nodes.

It is known that any graph G has a unique MD-tree with $O(|V(G)|)$ nodes, and it can be computed in time $O(|V(G)| + |E(G)|)$ [17]. We denote by $\mathsf{MD}(G)$ the unique MD-tree for a graph G. The *modular-width* $\mathsf{mw}(G)$ of a graph G is the maximum number of children of

a prime node in its MD-tree $\mathsf{MD}(G)$. The substitution tree T in Figure 5(a) is indeed the MD-tree for the graph G in Figure 5(b), and hence $\mathsf{mw}(G) = 4$; note that only x_{16} is a prime node in T.

We now define a variant of MD-trees, which will make our proofs and analyses simpler. A *pseudo modular decomposition tree* T (a *PMD-tree* for short) for a graph G is a substitution tree for G which satisfies the following two conditions:

- Each node $x \in V(T)$ applies to one of the following three types:
  - a 2-*join* node, whose quotient graph $\mathsf{Q}(x)$ is a complete graph with exactly two vertices;
  - a *parallel* node, whose quotient graph $\mathsf{Q}(x)$ is an edge-less graph; and
  - a *prime* node, whose quotient graph $\mathsf{Q}(x)$ is a prime with at least four vertices.
- No edge connects two parallel nodes.

▶ **Proposition 2.** *For any graph G, there exists a PMD-tree T with $O(|V(G)|)$ nodes such that each prime node $x \in V(T)$ has at most $\mathsf{mw}(G)$ children, and it can be constructed in polynomial time.*

We denote by $\mathsf{PMD}(G)$ a PMD-tree for G such that each prime node $x \in V(T)$ has at most $\mathsf{mw}(G)$ children. The *pseudo modular-width* $\mathsf{pmw}(G)$ of a graph G is the maximum number of children of a non-parallel node in its PMD-tree. Notice that $\mathsf{pmw}(G) = \max\{2, \mathsf{mw}(G)\}$ holds.

2.2 Other notation

Let $G = (V, E)$ be a graph, and let $L\colon V \to 2^C$ be a list. For two L-colorings f and f' of G, we define the *difference* $\mathsf{dif}(f, f')$ *between* f *and* f' as the set $\{v \in V\colon f(v) \neq f'(v)\}$. Notice that f and f' are adjacent if and only if $|\mathsf{dif}(f, f')| = 1$.

We express an instance $\mathcal{I}$ of LIST COLORING RECONFIGURATION by a 4-tuple (G, L, f_0, f_t) consisting of a graph G, a list L, and *initial* and *target* L-colorings f_0 and f_t of G.

Finally, we introduce a notion of "restriction" of mappings and instances. Consider an arbitrary mapping $\mu\colon V(G) \to S$, where G is a graph and S is any set. For a subgraph H of G, we denote by μ^H the *restriction* of μ on $V(H)$, that is, μ^H is a mapping from $V(H)$ to S such that $\mu^H(v) = \mu(v)$ for each vertex $v \in V(H)$. Let $\mathcal{I} = (G, L, f_0, f_t)$ be an instance of LIST COLORING RECONFIGURATION. For a subgraph H of G, we define the *restriction* $\mathcal{I}^H$ of $\mathcal{I}$ (on H) as the instance (H, L^H, f_0^H, f_t^H) of LIST COLORING RECONFIGURATION. Notice that f_0^H and f_t^H are proper L^H-colorings of H.

3 Fixed-Parameter Algorithm for Bounded Modular-Width Graphs

The following is our main theorem of this section.

▶ **Theorem 3.** LIST COLORING RECONFIGURATION *is fixed-parameter tractable when parameterized by $k + \mathsf{mw}$, where k and mw are the upper bounds on the size of the color set and the modular-width of an input graph, respectively.*

Because it is known that any cograph has modular-width zero, we have the following result as a corollary of Theorem 3.

▶ **Corollary 4.** LIST COLORING RECONFIGURATION *is fixed-parameter tractable for cographs when parameterized by the size k of the color set.*

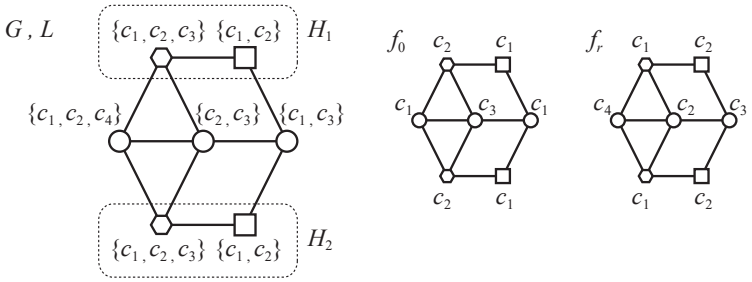


Figure 6 An instance $\mathcal{I} = (G, L, f_0, f_t)$ of LIST COLORING RECONFIGURATION, and two identical subgraphs H_1 and H_2.

Recall that $\mathsf{pmw}(G) = \max\{2, \mathsf{mw}(G)\}$, and hence $\mathsf{pmw}(G) \leq \mathsf{mw}(G) + 2$. Therefore, as a proof of Theorem 3, it suffices to give a fixed-parameter algorithm for LIST COLORING RECONFIGURATION with respect to $k + \mathsf{pmw}$, where pmw is an upper bound on $\mathsf{pmw}(G)$.

3.1 Reduction rule

In this subsection, we give a useful lemma, which compresses an input graph into a smaller graph with keeping the reconfigurability.

Let $\mathcal{I} = (G, L, f_0, f_t)$ be an instance of LIST COLORING RECONFIGURATION. For each vertex $v \in V(G)$, we define a *vertex assignment* $A(v)$ as a triple $(L(v), f_0(v), f_t(v))$ consisting of a list, and initial and target color assignments of v. Let H_1 and H_2 be two induced subgraphs of G such that $|V(H_1)| = |V(H_2)|$ and $V(H_1) \cap V(H_2) = \emptyset$. Then, H_1 and H_2 are *identical* (on $\mathcal{I}$) if there exists a bijective function $\phi\colon V(H_1) \to V(H_2)$ which satisfies the following two conditions:

1. H_1 and H_2 are isomorphic under ϕ, that is, $vw \in E(H_1)$ if and only if $\phi(v)\phi(w) \in E(H_2)$.
2. For all vertices $v \in V(H_1)$,
   a. $N(G, v) \setminus V(H_1) = N(G, \phi(v)) \setminus V(H_2)$; and
   b. $A(v) = A(\phi(v))$, that is, $L(v) = L(\phi(v))$, $f_0(v) = f_0(\phi(v))$ and $f_t(v) = f_t(\phi(v))$.

We note that the condition 2-a implies that there is no edge between H_1 and H_2. Figure 6 shows an example of identical subgraphs H_1 and H_2 on $\mathcal{I} = (G, L, f_0, f_t)$, where the bijective function maps each vertex in H_1 to a vertex in H_2 with the same shape.

We now prove the following key lemma, which holds for any graph.

▶ **Lemma 5.** (Reduction rule) *Let $\mathcal{I} = (G, L, f_0, f_t)$ be an instance of* LIST COLORING RECONFIGURATION*, and let H_1 and H_2 be two identical subgraphs of G. Then, $\mathcal{I}^{G \setminus H_2}$ is a yes-instance if and only if $\mathcal{I}$ is.*

3.2 Kernelization

Let $\mathcal{I} = (G, L, f_0, f_t)$ be an instance of LIST COLORING RECONFIGURATION. Suppose that the color set C has at most k colors, G is a connected graph with $\mathsf{pmw}(G) \leq \mathsf{pmw}$, and all vertices of G are totally ordered according to an arbitrary binary relation $\prec$.

3.2.1 Sufficient condition for identical subgraphs

We first give a sufficient condition for which two nodes in a PMD-tree $\mathsf{PMD}(G)$ for G correspond to identical subgraphs. Let $x \in V(\mathsf{PMD}(G))$ be a node, let $p := |V(\mathsf{CG}(x))|$, and assume that all vertices in $V(\mathsf{CG}(x))$ are labeled as $v_1, v_2, \ldots, v_p$ according to $\prec$; that is,

$v_i \prec v_j$ holds for each i, j with $1 \leq i < j \leq p$. Let $m \geq p$ be some integer which will be defined later. We now define an $(m+1) \times m$ matrix $\mathcal{M}_m(x)$ as follows:

$$(\mathcal{M}_m(x))_{i,j} = \begin{cases} 1 & \text{if } i,j \leq p \text{ and } v_iv_j \in E(\mathsf{CG}(x)); \\ 0 & \text{if } i,j \leq p \text{ and } v_iv_j \notin E(\mathsf{CG}(x)); \\ 0 & \text{if } p < i \leq m \text{ or } p < j \leq m; \\ A(v_j) & \text{if } i = m+1 \text{ and } j \leq p; \\ \emptyset & \text{otherwise,} \end{cases}$$

where $(\mathcal{M}_m(x))_{i,j}$ denotes an (i,j)-element of $\mathcal{M}_m(x)$. Notice that $\mathcal{M}_m(x)$ contains the adjacency matrix of $\mathsf{CG}(x)$ at its upper left $p \times p$ submatrix, and the bottommost row represents the vertex assignment of each vertex in $V(\mathsf{CG}(x))$. We call $\mathcal{M}_m(x)$ an *m-ID-matrix* of x. For example, consider the node x_{13} in Figure 5(a). Then, $p = 2$, and a 4-ID-matrix of x_{13} is as follows:

$$\mathcal{M}_4(x_{13}) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ A(x_3) & A(x_4) & \emptyset & \emptyset \end{bmatrix}$$

▶ **Lemma 6.** *Let y_1 and y_2 be two children of a parallel node x in $\mathsf{PMD}(G)$, and let m be an integer with $m \geq \max\{|V(\mathsf{CG}(y_1))|, |V(\mathsf{CG}(y_2))|\}$. If $\mathcal{M}_m(y_1) = \mathcal{M}_m(y_2)$ holds, then $\mathsf{CG}(y_1)$ and $\mathsf{CG}(y_2)$ are identical.*

3.2.2 Kernelization algorithm

We now describe how to kernelize an input instance. Our algorithm traverses a PMD-tree $\mathsf{PMD}(G)$ of G by a depth-first search in post-order starting from the root of $\mathsf{PMD}(G)$, that is, the algorithm processes a node of $\mathsf{PMD}(G)$ after its all children are processed.

Let $x \in V(\mathsf{PMD}(G))$ be a node which is currently visited. If x is a non-parallel node, we do nothing. Otherwise (i.e., if x is a parallel node,) let Y be the set of all children of x, and let $m := \max_{y \in Y} |V(\mathsf{CG}(y))|$. We first construct m-ID-matrices of all children of x. If there exist two nodes y_1 and y_2 such that $\mathcal{M}_m(y_1) = \mathcal{M}_m(y_2)$, then $\mathsf{CG}(y_1)$ and $\mathsf{CG}(y_2)$ are identical; and hence we remove $\mathsf{CG}(y_2)$ from G by Lemma 5. Then, we modify $\mathsf{PMD}(G)$ in order to keep it still being a PMD-tree for the resulting graph as follows. We remove a subtree rooted at y_2 from $\mathsf{PMD}(G)$, and delete a node corresponding to y_2 from a quotient graph $\mathsf{Q}(x)$ of x. If this removal makes x having only one child y in the PMD-tree, we contract the edge xy into a new node x' such that $\mathsf{Q}(x') = \mathsf{Q}(x)$.

The running time of this kernelization can be estimated as follows. For each node $x \in V(\mathsf{PMD}(G))$, the construction of m-ID-matrices can be done in time $O(|Y| \cdot m^2) = O(|V(G)|^3)$. We can check if $\mathcal{M}_m(y_1) = \mathcal{M}_m(y_2)$ for each pair of children y_1 and y_2 of x in time $O(m^2) = O(|V(G)|^2)$. Moreover, a modification of $\mathsf{PMD}(G)$, which follows an application of Lemma 5, can be done in polynomial time. Recall that the number of children of x and the size of a PMD-tree $\mathsf{PMD}(G)$ are both bounded linearly in $|V(G)|$, and hence our kernelization can be done in polynomial time.

3.2.3 Size of the kernelized instance

We finally prove that the size of the obtained instance $\mathcal{I}' = (G', L', f'_0, f'_t)$ depends only on $k + \mathsf{pmw}$; recall that pmw is the upper bound on $\mathsf{pmw}(G)$. By Observation 1, we can assume

that the maximum clique size $\omega(G')$ is at most k. In addition, G' is connected since G is connected and an application of Lemma 5 does not affect the connectivity of the graph. Therefore, it suffices to prove the following lemma.

▶ **Lemma 7.** *The graph G' has at most $h_{k,\mathsf{pmw}}(\omega(G'))$ vertices, where $h_{k,\mathsf{pmw}}(i)$ is recursively defined for an integer $i \geq 1$ as follows:*

$$h_{k,\mathsf{pmw}}(i) = \begin{cases} 1 & \text{if } i = 1; \\ \mathsf{pmw} \cdot h_{k,\mathsf{pmw}}(i-1) \cdot \sqrt{2}^{(h_{k,\mathsf{pmw}}(i-1))^2} \cdot (2^k \cdot k^2)^{h_{k,\mathsf{pmw}}(i-1)} & \text{otherwise.} \end{cases}$$

In particular, $h_{k,\mathsf{pmw}}(\omega(G'))$ depends only on $k + \mathsf{pmw}$.

Finally, we prove Theorem 3. By the above discussions, we can compute the kernelized instance $\mathcal{I}' = \mathcal{I}^{G'}$ of LIST COLORING RECONFIGURATION in polynomial time. Because the size of $\mathcal{I}'$ depends only on $k + \mathsf{pmw}$, we can solve $\mathcal{I}'$ by enumerating all $L^{G'}$-colorings. The running time for this enumeration depends only on $k + \mathsf{pmw}$, and hence we obtain a fixed-parameter algorithm for LIST COLORING RECONFIGURATION.

This completes the proof of Theorem 3.

4 Shortest Variant

In this section, we study the shortest variant, LIST COLORING SHORTEST RECONFIGURATION. We note that the shortest length can be expressed by a polynomial number of bits, because there are at most k^n colorings for a graph with n vertices and k colors. Therefore, the answer can be output in polynomial time. The following is our result.

▶ **Theorem 8.** LIST COLORING SHORTEST RECONFIGURATION *is fixed-parameter tractable when parameterized by $k + \mathsf{vc}$, where k and vc are the upper bounds on the sizes of the color set and a minimum vertex cover of an input graph, respectively.*

As a corollary, we have the following result.

▶ **Corollary 9.** LIST COLORING SHORTEST RECONFIGURATION *is fixed-parameter tractable for split graphs when parameterized by the size k of the color set.*

As a proof of Theorem 8, we give such a fixed-parameter algorithm. Our basic idea is the same as the fixed-parameter algorithm in Section 3. However, in order to compute the shortest length, we consider a more general "weighted" version of LIST COLORING SHORTEST RECONFIGURATION, which is defined as follows. Let $\mathcal{I} = (G, L, f_0, f_t)$ be an instance of LIST COLORING RECONFIGURATION, and assume that each vertex $v \in V(G)$ has a *weight* $w(v) \in \mathbb{N}$, where $\mathbb{N}$ is the set of all positive integers. For two adjacent L-colorings f and f' of a graph G, we define the *gap* $\mathsf{gap}_w(f, f')$ *between* f *and* f' as the weight $w(v)$ of v, where v is a unique vertex in $\mathsf{dif}(f, f')$. The *length* $\mathsf{len}_w(\mathcal{S})$ *of* a reconfiguration sequence $\mathcal{S} = \langle f_0, f_1, \ldots, f_\ell \rangle$ is defined as $\mathsf{len}_w(\mathcal{S}) = \sum_{i=1}^{\ell} \mathsf{gap}_w(f_{i-1}, f_i)$. We denote by $\mathsf{OPT}(\mathcal{I}, w)$ the minimum length of a reconfiguration sequence between f_0 and f_t; we define $\mathsf{OPT}(\mathcal{I}, w) = +\infty$ if $\mathcal{I}$ is a no-instance of LIST COLORING RECONFIGURATION. Then, LIST COLORING SHORTEST RECONFIGURATION can be seen as computing $\mathsf{OPT}(\mathcal{I}, w)$ for the case where every vertex has weight one. Thus, to prove Theorem 8, it suffices to construct a fixed-parameter algorithm for the weighted version when parameterized by $k + \mathsf{vc}$.

As with Section 3, we again use the concept of kernelization to prove Theorem 8. More precisely, for a given instance $(\mathcal{I}, w)$, we first construct an instance $(\mathcal{I}' = (G', L', f'_0, f'_t), w')$ in

polynomial time such that the size of $\mathcal{I}'$ depends only on $k+\mathsf{vc}$, and $\mathsf{OPT}(\mathcal{I}', w') = \mathsf{OPT}(\mathcal{I}, w)$ holds. Then, we can compute $\mathsf{OPT}(\mathcal{I}', w')$ by computing a (weighted) shortest path between f'_0 and f'_t in an edge-weighted graph defined as follows: the vertex set consists of all L'-colorings of G', and each pair of adjacent L'-colorings are connected by an edge with a weight corresponding to the gap between them.

4.1 Reduction rule for the weighted version

In this subsection, we give the counterpart of Lemma 5 for the weighted version.

Let $(\mathcal{I} = (G, L, f_0, f_t), w)$ be an instance of the weighted version, and assume that there exist two identical subgraphs H_1 and H_2 of G, both of which consist of single vertices, say, $V(H_1) = \{v_1\}$ and $V(H_2) = \{v_2\}$. We now define a new instance $(\mathcal{I}', w')$ as follows:

- $\mathcal{I}' = \mathcal{I}^{G \setminus H_2}$; and
- $w'(v_1) = w(v_1) + w(v_2)$ and $w'(v) = w(v)$ for any $v \in V(G) \setminus \{v_1, v_2\}$.

Intuitively, v_2 is merged into v_1 together with its weight. Then, we have the following lemma.

▶ **Lemma 10.** $\mathsf{OPT}(\mathcal{I}, w) = \mathsf{OPT}(\mathcal{I}', w')$.

4.2 Kernelization

Finally, we give a kernelization algorithm as follows.

Let $(\mathcal{I} = (G, L, f_0, f_t), w)$ be an instance of the weighted version such that G has a vertex cover of size at most vc. Because such a vertex cover can be computed in time $O(2^{\mathsf{vc}} \cdot |V(G)|)$ [8], we now assume that we are given a vertex cover V_C of size at most vc. Notice that $V_I := V \setminus V_C$ forms an independent set of G. Suppose that there exist two vertices $v_1, v_2 \in V_I$ such that $N(G, v_1) = N(G, v_2)$ and $A(v_1) = A(v_2)$ hold. Then, induced subgraphs $G[\{v_1\}]$ and $G[\{v_2\}]$ are identical. Therefore, we can apply Lemma 10 to remove v_2 from G, and modify a weight function without changing the optimality. As a kernelization, we repeatedly apply Lemma 10 for all such pairs of vertices in V_I, which can be done in polynomial time. Let G' be the resulting subgraph of G, and let $V'_I := V(G') \setminus V_C$. Since V_C is of size at most vc, it suffices to prove the following lemma.

▶ **Lemma 11.** $|V'_I| \leq 2^{\mathsf{vc}} \cdot 2^k \cdot k^2$.

This completes the proof of Theorem 8.

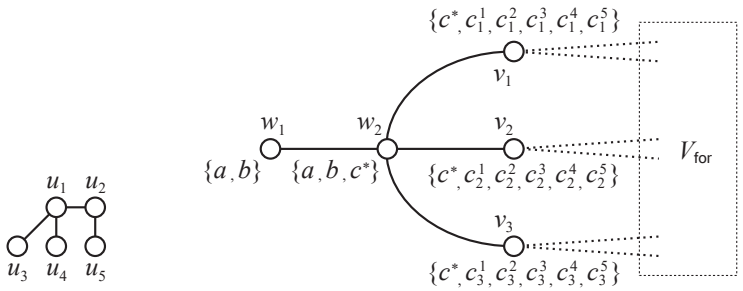
5 W[1]-Hardness

Because even the shortest variant is fixed-parameter tractable when parameterized by $k + \mathsf{vc}$, one may expect that vc is a strong parameter and the problem is fixed-parameter tractable with only vc. However, we prove the following theorem in this section.

▶ **Theorem 12.** List coloring reconfiguration *is W[1]-hard when parameterized by* vc*, where* vc *is the upper bound on the size of a minimum vertex cover of an input graph.*

Recall that list coloring reconfiguration is PSPACE-complete even for a fixed constant $k \geq 4$. Therefore, the problem is intractable if we take only one parameter, either k or vc.

In order to prove Theorem 12, we give an FPT-reduction from the independent set problem when parameterized by the solution size s, in which we are given a graph H and an integer $s \geq 0$, and asked whether H has an independent set of size at least s. This problem is known to be W[1]-hard [8].



(a) H , $s = 3$ (b) G , L

Figure 7 (a) An instance H of INDEPENDENT SET, and (b) the graph G and the list L. The set V_{for} contains vertices of $(i, j; p, q)$-forbidding gadgets for all $(i, j) \in \{(1, 2), (1, 3), (2, 3)\}$ and all $(p, q) \in \{(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (1, 2), (2, 1), (1, 3), (3, 1), (1, 4), (4, 1), (2, 5), (5, 2)\}$; thus $|V_{\mathsf{for}}| = 39$.

5.1 Construction

Let H be a graph with n vertices $u_1, u_2, \ldots, u_n$, and s be an integer as an input for INDEPENDENT SET. Then, we construct the corresponding instance (G, L, f_0, f_t) of LIST COLORING RECONFIGURATION as follows. (See also Figure 7.)

We first create s vertices $v_1, v_2, \ldots, v_s$, which are called *selection vertices*; let V_{sel} be the set of all selection vertices. For each $i \in \{1, 2, \ldots, s\}$, we set $L(v_i) = \{c^*, c_i^1, c_i^2, \ldots, c_i^n\}$. In our reduction, we will construct G and L so that assigning the color c_i^p, $p \in \{1, 2, \ldots, n\}$, to $v_i \in V_{\mathsf{sel}}$ corresponds to choosing the vertex $u_p \in V(H)$ as a vertex in an independent set of H. Then, in order to make a correspondence between a color assignment to V_{sel} and an independent set of size s in H, we need to construct the following properties:

- For each $p \in \{1, 2, \ldots, n\}$, we use at most one color from $\{c_1^p, c_2^p, \ldots, c_s^p\}$; this ensures that each vertex $u_p \in V(H)$ can be chosen at most once in an independent set.
- For each $p, q \in \{1, 2, \ldots, n\}$ with $u_p u_q \in E(H)$, we use at most one color from $\{c_1^p, c_2^p, \ldots, c_s^p, c_1^q, c_2^q, \ldots, c_s^q\}$; then, no two adjacent vertices in H are chosen in an independent set.

To do this, we define an $(i, j; p, q)$-forbidding gadget for $i, j \in \{1, 2, \ldots s\}$ and $p, q \in \{1, 2, \ldots, n\}$. The $(i, j; p, q)$-*forbidding gadget* is a vertex w which is adjacent to v_i and v_j and has a list $L(w) = \{c_i^q, c_j^p\}$. Observe that the vertex w forbids that v_i and v_j are simultaneously colored with c_i^p and c_j^q, respectively. In order to satisfy the desired properties above, we now add our gadgets as follows: for all $i, j \in \{1, 2, \ldots s\}$ with $i < j$,

- add an $(i, j; p, p)$-forbidding gadget for every vertex $u_p \in V(H)$; and
- add $(i, j; p, q)$- and $(i, j; q, p)$-forbidding gadgets for every edge $u_p u_q \in E(H)$.

We denote by V_{for} the set of all vertices in the forbidding gadgets. We finally create an edge consisting of two vertices w_1 and w_2 such that $L(w_1) = \{a, b\}$ and $L(w_2) = \{a, b, c^*\}$, and connect w_2 with all selection vertices in V_{sel}.

Finally, we construct two L-colorings f_0 and f_t of G as follows:

- for each $v_i \in V_{\mathsf{sel}}$, $f_0(v_i) = f_t(v_i) = c^*$;
- for each $w \in V_{\mathsf{for}}$, $f_0(w)$ and $f_t(w)$ are arbitrary chosen colors from $L(w)$; and
- $f_0(w_1) = f_t(w_2) = a$, and $f_t(w_1) = f_0(w_2) = b$.

Note that both f_0 and f_t are proper L-colorings of G. This completes the construction of (G, L, f_0, f_t).

5.2 Correctness of the reduction

In this subsection, we prove the following three statements:

- (G, L, f_0, f_t) can be constructed in time polynomial in the size of H.
- The upper bound vc on the size of a minimum vertex cover of G depends only on s.
- H is a yes-instance of INDEPENDENT SET if and only if (G, L, f_0, f_t) is a yes-instance of LIST COLORING RECONFIGURATION.

In order to prove the first statement, it suffices to show that the size of (G, L, f_0, f_t) is bounded polynomially in $n = |V(H)|$. From the construction, we have $|V(G)| = |V_{\mathsf{sel}}| + |V_{\mathsf{for}}| + |\{w_1, w_2\}| \leq s + s^2 \times (|V(H)| + 2|E(H)|) + 2 = O(n^4)$. In addition, each list contains $O(n)$ colors. Therefore, the construction can be done in time $O(n^{O(1)})$.

The second statement immediately follows from the fact that $\{w_2\} \cup V_{\mathsf{sel}}$ is a vertex cover in G of size $s + 1$; observe that $G \setminus (\{w_2\} \cup V_{\mathsf{sel}}) = G[\{w_1\} \cup V_{\mathsf{for}}]$ contains no edge.

Finally, we prove the third statement as follows.

▶ **Lemma 13.** *H is a yes-instance of* INDEPENDENT SET *if and only if (G, L, f_0, f_t) is a yes-instance of* LIST COLORING RECONFIGURATION.

This completes the proof of Theorem 12.

6 Conclusion

In this paper, we have studied LIST COLORING RECONFIGURATION from the viewpoint of parameterized complexity, in particular, with several graph parameters. We painted an interesting map of graph parameters in Figure 2 which shows the boundary between fixed-parameter tractability and intractability.

References

1. Marthe Bonamy and Nicolas Bousquet. Recoloring bounded treewidth graphs. *Electronic Notes in Discrete Mathematics*, 44:257–262, 2013. doi:10.1016/j.endm.2013.10.040.
2. Marthe Bonamy, Matthew Johnson, Ioannis Lignos, Viresh Patel, and Daniël Paulusma. Reconfiguration graphs for vertex colourings of chordal and chordal bipartite graphs. *Journal of Combinatorial Optimization*, 27(1):132–143, 2014. doi:10.1007/s10878-012-9490-y.
3. Paul Bonsma and Luis Cereceda. Finding paths between graph colourings: PSPACE-completeness and superpolynomial distances. *Theoretical Computer Science*, 410(50):5215–5226, 2009. doi:10.1016/j.tcs.2009.08.023.
4. Paul Bonsma, Amer E. Mouawad, Naomi Nishimura, and Venkatesh Raman. The complexity of bounded length graph recoloring and CSP reconfiguration. In *Parameterized and Exact Computation - 9th International Symposium, IPEC 2014, Wroclaw, Poland, September 10-12, 2014. Revised Selected Papers*, pages 110–121, 2014. doi:10.1007/978-3-319-13524-3_10.
5. Richard C. Brewster, Sean McGuinness, Benjamin Moore, and Jonathan A. Noel. A dichotomy theorem for circular colouring reconfiguration. *Theoretical Computer Science*, 639:1–13, 2016. doi:10.1016/j.tcs.2016.05.015.
6. Luis Cereceda. *Mixing Graph Colourings*. PhD thesis, The London School of Economics and Political Science, 2007.
7. Luis Cereceda, Jan van den Heuvel, and Matthew Johnson. Finding paths between 3-colorings. *Journal of Graph Theory*, 67(1):69–82, 2011. doi:10.1002/jgt.20514.
8. Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.

9 Martin Dyer, Abraham D. Flaxman, Alan M. Frieze, and Eric Vigoda. Randomly coloring sparse random graphs with fewer colors than the maximum degree. *Random Structures & Algorithms*, 29(4):450–465, 2006. doi:10.1002/rsa.20129.

10 Jakub Gajarský, Michael Lampis, and Sebastian Ordyniak. Parameterized algorithms for modular-width. In *Parameterized and Exact Computation - 8th International Symposium, IPEC 2013, Sophia Antipolis, France, September 4-6, 2013, Revised Selected Papers*, pages 163–176, 2013. doi:10.1007/978-3-319-03898-8_15.

11 Tibor Gallai. Transitiv orientierbare graphen. *Acta Mathematica Academiae Scientiarum Hungarica*, 18(1):25–66, 1967. doi:10.1007/BF02020961.

12 Michel Habib and Christophe Paul. A survey of the algorithmic aspects of modular decomposition. *Computer Science Review*, 4(1):41–59, 2010. doi:10.1016/j.cosrev.2010.01.001.

13 Tatsuhiko Hatanaka, Takehiro Ito, and Xiao Zhou. The list coloring reconfiguration problem for bounded pathwidth graphs. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E98.A(6):1168–1178, 2015. doi:10.1587/transfun.E98.A.1168.

14 Takehiro Ito, Erik D. Demaine, Nicholas J.A. Harvey, Christos H. Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno. On the complexity of reconfiguration problems. *Theoretical Computer Science*, 412(12):1054–1065, 2011. doi:10.1016/j.tcs.2010.12.005.

15 Matthew Johnson, Dieter Kratsch, Stefan Kratsch, Viresh Patel, and Daniël Paulusma. Finding shortest paths between graph colourings. *Algorithmica*, 75(2):295–321, 2016. doi:10.1007/s00453-015-0009-7.

16 Sampath Kannan, Moni Naor, and Steven Rudich. Implicat representation of graphs. *SIAM Journal on Discrete Mathematics*, 5(4):596–603, 1992. doi:10.1137/0405049.

17 Ross M. McConnell and Fabien de Montgolfier. Linear-time modular decomposition of directed graphs. *Discrete Applied Mathematics*, 145(2):198–209, 2005. doi:10.1016/j.dam.2004.02.017.

18 Jan van den Heuvel. The complexity of change. In *Surveys in Combinatorics 2013*, pages 127–160. 2013. doi:10.1017/CBO9781139506748.005.

19 Marcin Wrochna. Reconfiguration in bounded bandwidth and treedepth. *CoRR*, abs/1405.0847, 2014.

20 Marcin Wrochna. Homomorphism reconfiguration via homotopy. In *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, pages 730–742, 2015. doi:10.4230/LIPIcs.STACS.2015.730.