Coloring Reconfiguration Under Color Swapping

Janosch Fuchs
IT Center, RWTH Aachen University, Germany

Rin Saito
Graduate School of Information Sciences, Tohoku University, Japan

Tatsuhiro Suga
Graduate School of Information Sciences, Tohoku University, Japan

Takahiro Suzuki
Graduate School of Information Sciences, Tohoku University, Japan

Yuma Tamura
Graduate School of Information Sciences, Tohoku University, Japan

Abstract

In the Coloring Reconfiguration problem, we are given two proper k-colorings of a graph and asked to decide whether one can be transformed into the other by repeatedly applying a specified recoloring rule, while maintaining a proper coloring throughout. For this problem, two recoloring rules have been widely studied: *single-vertex recoloring* and *Kempe chain recoloring*. In this paper, we introduce a new rule, called *color swapping*, where two adjacent vertices may exchange their colors, so that the resulting coloring remains proper, and study the computational complexity of the problem under this rule. We first establish a complexity dichotomy with respect to k: the problem is solvable in polynomial time for $k \leq 2$, and is PSPACE-complete for $k \geq 3$. We further show that the problem remains PSPACE-complete even on restricted graph classes, including bipartite graphs, split graphs, and planar graphs of bounded degree. In contrast, we present polynomial-time algorithms for several graph classes: for paths when $k = 3$, for split graphs when k is fixed, and for cographs when k is arbitrary.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases Combinatorial reconfiguration, graph coloring, PSPACE-complete, graph algorithm

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2025.33

Related Version *Full Version*: https://arxiv.org/abs/2511.06473

Funding *Rin Saito*: Supported by JST SPRING Grant Number JPMJSP2114.
Yuma Tamura: Supported by JSPS KAKENHI Grant Number JP25K21148.

1 Introduction

1.1 Reconfiguring of Colorings

The field of *combinatorial reconfiguration* investigates reachability and connectivity in the solution space of combinatorial problems. A combinatorial reconfiguration problem is defined with respect to a combinatorial problem Π and a *reconfiguration rule* R, which specifies how one feasible solution of Π can be transformed into another. Given an instance of Π and two feasible solutions, the reconfiguration problem asks whether it is possible to transform one into the other through a sequence of solutions, each obtained by a single application of R, such that all intermediate solutions are also feasible. Combinatorial reconfiguration problems naturally arise in applications involving dynamic systems, where solutions must be updated while maintaining feasibility at every step. In addition, studying such problems provides deeper insights into the structural properties of the solution spaces of classical combinatorial problems. We refer the reader to the surveys by van den Heuvel [20] and Nishimura [31] for comprehensive overviews of this growing area.


© Janosch Fuchs, Rin Saito, Tatsuhiro Suga, Takahiro Suzuki, and Yuma Tamura;
licensed under Creative Commons License CC-BY 4.0

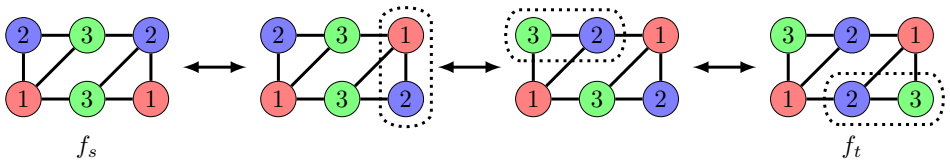


Figure 1 A reconfiguration sequence between two proper 3-colorings f_s and f_t under color swapping.

Among the various reconfiguration problems, one of the most fundamental and extensively studied is the reconfiguration of graph colorings. Let k be a positive integer. For a fixed reconfiguration rule R, the Coloring Reconfiguration problem under R is defined as follows: given a k-colorable graph G and two proper k-colorings f and f', determine whether there exists a sequence of proper k-colorings of G starting at f and ending at f', where each coloring in the sequence is obtained from the previous one by a single application of R. The k-Coloring Reconfiguration is a special case where k is fixed. The computational complexity of k-Coloring Reconfiguration has been the subject of extensive algorithmic study; see Section 3 of the survey by Mynhardt and Nasserasr [30].

Two reconfiguration rules have been widely studied for k-Coloring Reconfiguration: *single-vertex recoloring* and *Kempe chain recoloring*. In the *single-vertex recoloring* rule, a new proper k-coloring is obtained by recoloring a single vertex so that the resulting k-coloring remains proper. Under this rule, k-Coloring Reconfiguration is solvable in polynomial time when $k \leq 3$ [14], while it becomes PSPACE-complete for every fixed $k \geq 4$ [8]. Notably, this rule is closely related to Glauber dynamics in statistical physics, where a Markov chain is defined over the space of proper k-colorings of a graph G: at each step, a vertex is selected uniformly at random and recolored with a randomly chosen color such that the resulting coloring remains proper. See Sokal [33] for an introduction to the Potts model and its connections to graph coloring.

The second widely studied rule is *Kempe chain recoloring*. A new proper k-coloring is obtained by selecting a connected component C of the subgraph of G induced by two color classes (i.e., a *Kempe chain*) and swapping the two colors within C. Note that when C consists of a single vertex, this operation is equivalent to single-vertex recoloring. While k-Coloring Reconfiguration under this rule is solvable in polynomial time when $k \leq 2$ [27] (in fact, the answer is always "Yes"), it becomes PSPACE-complete for every fixed $k \geq 3$ [6]. Kempe chain recoloring was originally introduced by Kempe in 1879 in an attempt to prove the Four Color Theorem [24]. Although his proof was later found to be flawed, the technique has continued to play a central role in graph coloring theory [5, 27], statistical physics [28, 29], and the study of mixing times of Markov chains [36].

These results highlight a key aspect of reconfiguration problems: even when the definition of feasible solutions remains the same, the computational complexity of finding a reconfiguration sequence can vary greatly depending on the selected reconfiguration rule.

1.2 Our Contribution

In this paper, we introduce a new reconfiguration rule, called *color swapping* (CS) for Coloring Reconfiguration. A new proper k-coloring is obtained from a given one by swapping the colors of the endpoints of a single edge uv in G, so that the resulting coloring remains proper. For example, Figure 1 shows a reconfiguration sequence between f_s and f_t under CS, hence it is a yes-instance. We refer to Coloring Reconfiguration and k-Coloring Reconfiguration under CS as CRCS and k-CRCS, respectively. The color swapping rule can be seen as a restricted variant of Kempe chain recoloring, where each Kempe

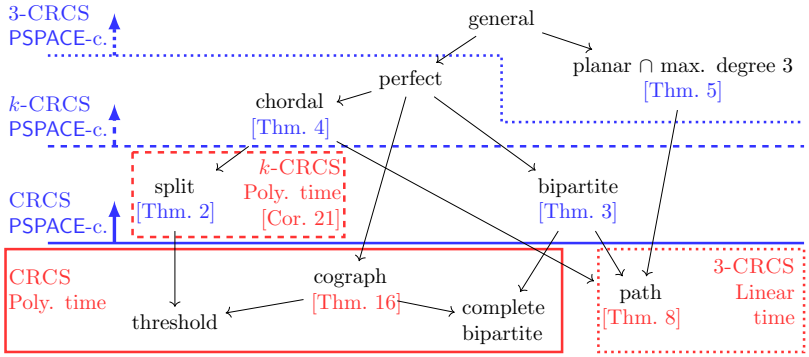


Figure 2 Our results for graph classes. Each arrow represents the inclusion relationship between classes: $A \to B$ means that the graph class B is a proper subclass of the graph class A.

chain is limited to exactly two vertices. Interestingly, this rule is also studied in statistical physics as *(local) Kawasaki dynamics*, which models dynamics in the fixed-magnetization Ising model [12, 23, 25].

The contribution of this paper is an analysis of the computational complexity of CRCS and k-CRCS; for an overview of our results, we refer to Figure 2. First, we prove that CRCS is PSPACE-complete even when the input graph is restricted to bipartite or split graphs. Furthermore, we show that there exists a positive integer k_0 such that for every fixed $k \geq k_0$, k-CRCS becomes PSPACE-complete even when the input graph is restricted to chordal graphs, which is a superclass of split graphs.

We also establish a complexity dichotomy with respect to the number k of colors: k-CRCS is PSPACE-complete for any fixed $k \geq 3$, whereas for $k \leq 2$, the problem can be solved in polynomial time. In particular, we show that 3-CRCS is PSPACE-complete even when restricted to planar graphs with maximum degree 3 and bounded bandwidth.

Complementing these hardness results, we also present several positive results. We first show that 3-CRCS can be solved in linear time on path graphs. To this end, we introduce an invariant for proper 3-colorings of paths, and design a linear-time algorithm that checks whether two input colorings have the identical invariants. While the algorithm is simple, its correctness requires a non-trivial argument.

Next, we show that CRCS can be solved in polynomial time on cographs. Our algorithm is based on a recursive procedure over the cotree of the input cograph, inspired by prior work [22] for Independent Set Reconfiguration on cographs. To adapt this approach to our problem, we introduce a new notion called *extended k-colorings* as a generalization of k-colorings.

Finally, we show that k-CRCS on split graphs is polynomial-time solvable for any fixed k. This contrasts with the PSPACE-hardness of CRCS on split graphs when k is unbounded.

Due to the space limitation, we omit the proofs of statements with the symbol $\star$ marks, which can be found in the full version of this paper.

1.3 Related Work

As mentioned in Section 1.1, the complexity of k-Coloring Reconfiguration has been extensively investigated with respect to various graph classes. Under both the single-vertex recoloring and Kempe chain recoloring rules, the problem is known to be PSPACE-complete in general. Moreover, stronger hardness results and polynomial-time algorithms have been established for specific graph classes.

Under the single-vertex recoloring rule, the problem remains PSPACE-complete even on bipartite planar graphs [8] and chordal graphs [17]. On the positive side, it is solvable in polynomial time for 2-degenerate graphs, q-trees (for fixed q), trivially perfect graphs, and split graphs [17]. Under the Kempe chain recoloring rule, the problem is PSPACE-complete even on planar graphs with maximum degree 6 [6]; however, it is polynomial-time solvable on chordal graphs, bipartite graphs, and cographs [6]. Several other algorithmic and structural aspects of k-Coloring Reconfiguration have also been studied, including finding a shortest reconfiguration sequence [9, 21] and bounding its length [1, 3, 10, 13].

The term *color swapping* is inspired by the Colored Token Swapping problem [7, 38], a reconfiguration problem involving token placements. In that problem, one is given a graph with an initial and a target coloring, which *need not be proper*, and the goal is to transform the initial coloring into the target one using the minimum number of swaps between tokens on adjacent vertices. Since feasibility is not restricted to proper colorings, this can be viewed as a variant of our reconfiguration problem in which the feasibility condition is relaxed. Yamanaka et al. [38] showed that Colored Token Swapping is NP-hard even when the number of colors is exactly three.

2 Preliminaries

For a positive integer k, we write $[k] = \{1, 2, \ldots, k\}$. For sets X and Y, the *symmetric difference* of X and Y is defined as $X \triangle Y \coloneqq (X \setminus Y) \cup (Y \setminus X)$. For a map $f\colon X \to Y$ and an element $y \in Y$, the *preimage* of y under f is defined as $f^{-1}(y) \coloneqq \{x \in X \mid f(x) = y\}$.

Let $G = (V, E)$ be an undirected graph. We use $V(G)$ and $E(G)$ to denote the vertex set and edge set of G, respectively. For a vertex v of G, $N_G(v)$ and $N_G[v]$ denote the *open neighborhood* and the *closed neighborhood* of v in G, respectively; that is, $N_G(v) = \{u \in V \mid uv \in E\}$ and $N_G[v] = N_G(v) \cup \{v\}$. For a vertex set $X \subseteq V$, we define $N_G(X) = \{v \in V \setminus X \mid u \in X, uv \in E(G)\}$ and $N_G[X] = N_G(X) \cup X$.

For a positive integer k, a *(proper) k-coloring* of a graph G is a map $f\colon V(G) \to [k]$ that assigns different colors to adjacent vertices; in other words, $f(u) \neq f(v)$ for every edge $uv \in E(G)$. An *independent set* of a graph G is a subset $I \subseteq V(G)$ such that for all $u, v \in I$, $uv \notin E(G)$ holds. A *clique* of a graph G is a subset $C \subseteq V(G)$ such that for all $u, v \in C$ with $u \neq v$, $uv \in E(G)$ holds.

2.1 Our Problems

For two proper colorings f and f' of a graph G, we say that f and f' are *adjacent* under *Color Swapping* (denoted by CS) if and only if the following two conditions hold:

1. There exists exactly one edge $uv \in E(G)$ such that $f(u) = f'(v)$ and $f(v) = f'(u)$, and
2. For all other vertices $w \in V(G) \setminus \{u, v\}$, we have $f(w) = f'(w)$.

Intuitively, f' can be obtained from f by swapping the colors of two adjacent vertices.

A sequence $f_0, f_1, \ldots, f_\ell$ of proper k-colorings of G with $f_0 = f$ and $f_\ell = f'$ is called a *reconfiguration sequence* between f and f' if f_{i-1} and f_i are adjacent under CS for all $i \in [\ell]$. We say that f and f' are *reconfigurable* if such a sequence exists. We now define the Coloring Reconfiguration under Color Swapping problem (CRCS for short). In CRCS, we are given a graph G, a positive integer k, and two proper k-colorings f_s and f_t of G. The goal is to determine whether there exists a reconfiguration sequence between f_s and f_t. For a fixed positive integer k, the k-CRCS problem is the special case of CRCS in which the two input colorings are k-colorings.

An instance (G, k, f_s, f_t) of CRCS is said to be *valid* if, for each color $i \in [k]$, the number of vertices assigned color i is the same in f_s and f_t; that is, $|f_s^{-1}(i)| = |f_t^{-1}(i)|$. Note that if f_s and f_t are reconfigurable, then (G, k, f_s, f_t) must be valid. This observation implies that if an instance (G, k, f_s, f_t) of CRCS is not valid, we can immediately return "NO." Therefore, throughout this paper, we assume that all instances of CRCS are valid.

It is easy to see that k-CRCS for $k \leq 2$ can be solved in polynomial time.

▶ **Observation 1** ($\star$). *k-CRCS can be solved in polynomial time for $k \leq 2$.*

3 PSPACE-completeness

We first observe that CRCS is solvable using polynomial space, i.e., CRCS belongs to the class PSPACE. This follows from the equivalence PSPACE = NPSPACE, which is a consequence of Savitch's theorem [32]. To see the membership of PSPACE, consider a reconfiguration sequence for CRCS as a certificate. Our polynomial-space algorithm reads each k-coloring in the sequence one by one and verifies that (i) each coloring is proper and (ii) each pair of consecutive colorings is adjacent under CS. Since each of these checks can be performed using polynomial space, this implies that CRCS can be solved in nondeterministic polynomial space. Therefore, by PSPACE = NPSPACE, we conclude that CRCS is in PSPACE.

In this section, we present polynomial-time reductions from three different problems to establish the PSPACE-completeness of our problem for several graph classes. Specifically, we reduce from the following problems: the Token Sliding problem in Section 3.1, the Coloring Reconfiguration problem in Section 3.2, and the Nondeterministic Constraint Logic problem in Section 3.3.

3.1 Reduction from Token Sliding

In this subsection, we present a polynomial-time reduction from the Token Sliding problem to CRCS. Token Sliding is also known as the Independent Set Reconfiguration under Token Sliding [18, 22].

Let G be a graph, and let $I \subseteq V(G)$ be a vertex subset of G. Recall that I is an *independent set* of G if no two vertices in I are adjacent; that is, $uv \notin E(G)$ for all $u, v \in I$. Two independent sets I and J of the same size are said to be *adjacent* under *token sliding* if and only if $|I \triangle J| = 2$ and the two vertices in $I \triangle J$ are joined by an edge in G.

In the Token Sliding problem, we are given a graph G and two independent sets $I_s, I_t \subseteq V(G)$ of the same size. The goal is to determine whether there exists a sequence $I_0, I_1, \ldots, I_\ell$ of independent sets of G such that $I_s = I_0$ and $I_t = I_\ell$, and for every $i \in [\ell]$, I_{i-1} and I_i are adjacent.

Token Sliding is known to be PSPACE-complete even when restricted to split graphs [2] or bipartite graphs [26]. A graph is *split* if its vertices can be partitioned into a clique and an independent set, and *bipartite* if its vertices can be partitioned into two independent sets.

We first show the following theorem for split graphs.

▶ **Theorem 2.** CRCS *is* PSPACE*-complete for split graphs.*

Proof. We have already observed that the problem is in PSPACE. To show the PSPACE-hardness, we give a polynomial-time reduction from Token Sliding on split graphs to CRCS.

Let (G, I_s, I_t) be an instance of Token Sliding such that $|I_s| = |I_t|$ and $G = (V, E)$ is a split graph with vertex partition (C, S), where C is a clique of G and S is an independent set of G. Assume that $|I_s| = |I_t| \geq 2$. Note that Token Sliding remains PSPACE-complete under this assumption since the problem is trivial when $|I_s| = |I_t| = 1$.

We construct an instance (G', k, f_s, f_t) of CRCS as follows. Let G' be the graph obtained from G by adding a new vertex u that is adjacent to all vertices in $V(G)$. That is, let $G' = (V', E')$ with $V' = V \cup \{u\}$ and $E' = E \cup \{uv \mid v \in V\}$. Since u is adjacent to all vertices in C, the set $C \cup \{u\}$ is a clique of G', and S is an independent set of G'. Thus, G' is also a split graph.

Let $k = |V'| - |I_s| + 1 = |V'| - |I_t| + 1$. We now define proper k-colorings f_s and f_t of G'. For each $v \in I_s$, set $f_s(v) = 1$. Then, assign a distinct color from $[k] \setminus \{1\}$ arbitrarily to each vertex in $V' \setminus I_s$ so that f_s is a proper k-coloring. Similarly, define f_t by setting $f_t(v) = 1$ for each $v \in I_t$, and assigning a distinct color from $[k] \setminus \{1\}$ arbitrarily to each vertex in $V' \setminus I_t$ so that f_t is also a proper k-coloring. Since $|V' \setminus I_s| = k - 1$ and $|V' \setminus I_t| = k - 1$, such proper k-colorings f_s and f_t exist.

This completes the construction of the instance (G', k, f_s, f_t). We claim that (G, I_s, I_t) is a yes-instance of Token Sliding if and only if (G', k, f_s, f_t) is a yes-instance of CRCS.

We first show the "only if" direction. Suppose that (G, I_s, I_t) is a yes-instance of Token Sliding. Then, there exists a sequence of independent sets $I_0, I_1, \ldots, I_\ell$ of G, with $I_0 = I_s$ and $I_\ell = I_t$, and for each $i \in [\ell]$, I_{i-1} and I_i are adjacent under token sliding. For each $i \in \{0, 1, \ldots, \ell\}$, we construct a proper k-coloring f_i of G' from I_i, following the same construction as for f_s and f_t: for each $v \in I_i$, set $f_i(v) = 1$, and assign a distinct color from $[k] \setminus \{1\}$ arbitrarily to each vertex in $V' \setminus I_i$ so that f_i is a proper k-coloring of G'.

We claim that for all $i \in [\ell]$, f_{i-1} and f_i are reconfigurable under CS. Let f' be the coloring obtained from f_{i-1} by exchanging the colors assigned to the vertices $v \in I_{i-1} \setminus I_i$ and $w \in I_i \setminus I_{i-1}$. Since each vertex in I_{i-1} is assigned the color 1 in f_{i-1}, and all other vertices are assigned distinct colors from $[k] \setminus \{1\}$, the modified coloring f' remains a proper k-coloring of G'. Moreover, since I_{i-1} and I_i are adjacent under token sliding, we have $vw \in E(G')$. Thus, f_{i-1} and f' are adjacent under CS.

We now show that f' and f_i are reconfigurable under CS. Recall that $f'(v) = f_i(v) = 1$ for all $v \in I_i$. Thus, f' and f_i may differ only on the vertices in $V(G') \setminus I_i$. Note that the restrictions of f' and f_i to $V(G') \setminus I_i$ are both bijective.

By construction, $G'[V(G') \setminus I_i]$ is connected, since it contains a universal vertex u adjacent to all others. It is known that for any connected n-vertex graph and two bijective n-colorings, there exists a reconfiguration sequence of length $O(n^2)$ between them under CS [37]. Applying this result, we can reconfigure f' into f_i using color swaps only within $V(G') \setminus I_i$. Thus, f_{i-1} and f_i are reconfigurable under CS. This implies that we can construct a reconfiguration sequence between f_s and f_t under CS. Therefore, (G', k, f_s, f_t) is a yes-instance of CRCS.

We now prove the "if" direction. Suppose that there exists a reconfiguration sequence between f_s and f_t under CS. Let $f_0, f_1, \ldots, f_\ell$ be the reconfiguration sequence, where $f_0 = f_s$ and $f_\ell = f_t$. For each $i \in \{0, 1, \ldots, \ell\}$, define $I_i = f_i^{-1}(1)$. Since the number of vertices colored 1 remains constant throughout the sequence, it follows that $|I_i| = |I_s| = |I_t| \geq 2$ for all i. By construction, the vertex u is adjacent to all other vertices in G', and thus cannot be assigned color 1 in any f_i; hence, $I_i \subseteq V(G)$. Moreover, since f_i is a proper k-coloring of G, the set I_i forms an independent set of G', and consequently also an independent set of G.

For each $i \in [\ell]$, since two consecutive proper k-colorings f_{i-1} and f_i are adjacent under CS, we can observe that I_{i-1} and I_i are either identical or adjacent under token sliding. By deleting any consecutive duplicate independent sets from the $I_0, I_1, \ldots, I_\ell$, we obtain the desired sequence between I_s and I_t. Therefore, (G, I_s, I_t) is a yes-instance of Token Sliding.

This completes the proof of Theorem 2. ◀

The similar result holds for bipartite graphs.

▶ **Theorem 3.** CRCS *is* PSPACE*-complete for bipartite graphs.*

Proof. We have already observed that the problem is in PSPACE. To show the PSPACE-hardness, we give a polynomial-time reduction from Token Sliding on bipartite graphs to CRCS.

Let (G, I_s, I_t) be an instance of Token Sliding, where G is a bipartite graph with a bipartition (S, T), and both S and T are independent sets of G. We construct an instance (G', k, f_s, f_t) of CRCS as follows (see also Figure 3).

First, we construct G' by adding three new vertices x_1, x_2, x_3 to S, and three new vertices y_1, y_2, y_3 to T. We then make x_1 adjacent to all vertices in $T \cup \{y_1, y_2, y_3\}$, and y_1 adjacent to all vertices in $S \cup \{x_1, x_2, x_3\}$. Since $S \cup \{x_1, x_2, x_3\}$ and $T \cup \{y_1, y_2, y_3\}$ are independent sets of G', the resulting graph G' is also bipartite.

We set $k = |V(G')| - |I_s| - 3$, and define proper k-colorings f_s and f_t of G' as follows. For the initial coloring f_s, assign color 1 to every vertex in $I_s \cup \{x_2, x_3, y_2, y_3\}$. Then, assign a distinct color from $[k] \setminus \{1\}$ arbitrarily to each vertex in $V(G') \setminus (I_s \cup \{x_2, x_3, y_2, y_3\})$ so that f_s is a proper k-coloring of G'. Similarly, for the target coloring f_t, assign color 1 to every vertex in $I_t \cup \{x_2, x_3, y_2, y_3\}$. Then, assign a distinct color from $[k] \setminus \{1\}$ arbitrarily to each vertex in $V(G') \setminus (I_t \cup \{x_2, x_3, y_2, y_3\})$ such that f_t is also a proper k-coloring of G'. Note that both $V(G') \setminus (I_s \cup \{x_2, x_3, y_2, y_3\})$ and $V(G') \setminus (I_t \cup \{x_2, x_3, y_2, y_3\})$ contain exactly $(k-1)$ vertices. Moreover, both $(I_s \cup \{x_2, x_3, y_2, y_3\})$ and $(I_t \cup \{x_2, x_3, y_2, y_3\})$ form independent sets of G'. Thus, it is always possible to assign the remaining $(k-1)$ colors so that both f_s and f_t are proper k-colorings of G'.

This completes the construction of the instance (G', k, f_s, f_t). We then claim that (G, I_s, I_t) is a yes-instance of Token Sliding if and only if (G', k, f_s, f_t) is a yes-instance of CRCS.

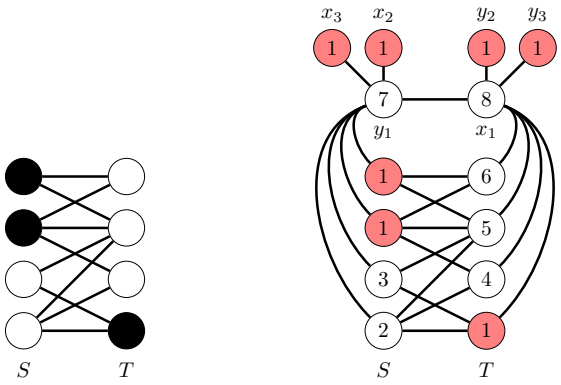
We first show the "only if" direction. Suppose that (G, I_s, I_t) is a yes-instance of Token Sliding. Then, there exists a sequence of independent sets $I_0, I_1, \ldots, I_\ell$ of G, with $I_0 = I_s$ and $I_\ell = I_t$, and for each $i \in [\ell]$, I_{i-1} and I_i are adjacent under token sliding. For each $i \in \{0, 1, \ldots, \ell\}$, we construct a proper k-coloring f_i of G' from I_i, following the same construction as for f_s and f_t. for the coloring f_i, assign color 1 to every vertex in $I_i \cup \{x_2, x_3, y_2, y_3\}$. Then, assign a distinct color from $[k] \setminus \{1\}$ arbitrarily to each vertex in $V(G') \setminus (I_i \cup \{x_2, x_3, y_2, y_3\})$ such that f_i is also a proper k-coloring.

We claim that f_{i-1} and f_i are reconfigurable under CS. Let f' be the coloring obtained from f_{i-1} by exchanging the colors assigned to the vertices $v \in I_{i-1} \setminus I_i$ and $w \in I_i \setminus I_{i-1}$. Since each vertex in $I_i \cup \{x_2, x_3, y_2, y_3\}$ is assigned the color 1, and all other vertices are assigned distinct colors from $[k] \setminus \{1\}$ by f', the modified coloring f' remains a proper k-coloring of G'. Moreover, since I_{i-1} and I_i are adjacent under token sliding, we have $vw \in E(G')$. Hence, f_i and f' are adjacent under CS.

We now prove that f' and f_i are reconfigurable under CS. Recall that $f'(v) = f_i(v) = 1$ for all $v \in I_i \cup \{x_2, x_3, y_2, y_3\}$ Thus, f' and f_i may differ only on the vertices in $[k] \setminus \{1\}$. Note that the restrictions of f' and f_i to $V(G') \setminus I_i$ are both bijective.

Recall that x_1 is adjacent to every vertex in $T \cup \{y_1, y_2, y_3\}$ and y_1 is adjacent to every vertex in $S \cup \{x_1, x_2, x_3\}$, so the subgraph induced by $V(G') \setminus (I_i \cup \{x_2, x_3, y_2, y_3\})$ is connected. As shown in the proof of Theorem 2 and using the result from [37], we see that f' and f_i are reconfigurable under CS, and thus so are f_{i-1} and f_i. By repeating this process for all $i \in [\ell]$, we obtain a reconfiguration sequence from f_s to f_t under CS. Therefore, (G', k, f_s, f_t) is a yes-instance of CRCS.

We now prove the "if" direction. Suppose that there exists a reconfiguration sequence between f_s and f_t under CS. Let $f_0, f_1, \ldots, f_\ell$ be the reconfiguration sequence, where $f_0 = f_s$ and $f_\ell = f_t$.



(a) (b)

Figure 3 (a) The bipartite graph G for an instance of the Token Sliding problem, along with its initial independent set I, where the vertices in I are marked in black. The vertex set of the graph G is partitioned into independent sets S and T. (b) The bipartite graph G', constructed from G, has a bipartition $V(G') = (S \cup \{x_1, x_2, x_3\}) \cup (T \cup \{y_1, y_2, y_3\})$, where both parts are independent sets. Each vertex is labeled with its color in the initial proper k-coloring f_s derived from I.

Since x_2 and x_3 share the same neighbor y_1 and are both assigned color 1 in f_s, no color swap involving x_2 or x_3 is allowed throughout the sequence; that is, $f_i(x_2) = f_i(x_3) = 1$ for all $i \in \{0, 1, \ldots, \ell\}$. By symmetry, we also have $f_i(y_2) = f_i(y_3) = 1$, which implies that x_1 and y_1 are never assigned color 1. For each $i \in \{0, 1, \ldots, \ell\}$, define $I_i = f_i^{-1}(1)$ and $I'_i = I_i \setminus \{x_2, x_3, y_2, y_3\}$. Then, $|I'_i| = |I_s| = |I_t|$, and since I_i is an independent set in G', it follows that I'_i is also an independent set in G.

Furthermore, since f_{i-1} and f_i are adjacent under CS for each $i \in [\ell]$, the sets I'_{i-1} and I'_i are either adjacent under token sliding or identical. By removing consecutive duplicates from the sequence $I'_0, I_1, \ldots, I'_\ell$, we obtain a reconfiguration sequence of independent sets from I_s to I_t under token sliding. Therefore, (G, I_s, I_t) is a yes-instance of Token Sliding.

This completes the proof of Theorem 2. ◀

3.2 Reduction from Coloring Reconfiguration

In this subsection, we give a polynomial-time reduction from the k-Coloring Reconfiguration problem under single-vertex recoloring to k-CRCS.

Let k be a positive integer. In k-Coloring Reconfiguration under single-vertex recoloring, we are given a graph G and two k-colorings g and g' of G. The goal is to determine whether there exists a sequence of k-colorings $g_0, g_1, \ldots, g_\ell$ with $g_0 = g$ and $g_\ell = g'$ such that for each $i \in [\ell]$, the colorings g_{i-1} and g_i differ at exactly one vertex; that is, $|\{v \in V(G) \mid g_{i-1}(v) \neq g_i(v)\}| = 1$. We simply call the problem k-Coloring Reconfiguration. It is known that there exists a positive integer k_0 such that, for any fixed $k \geq k_0$, k-Coloring Reconfiguration is PSPACE-complete on chordal graphs [17]. Recall that a graph is *chordal* if it contains no induced cycle of length at least 4.

We claim the following theorem.

▶ **Theorem 4.** *There exists a positive integer k_0 such that, for any fixed $k \geq k_0$, k-CRCS is* PSPACE*-complete on chordal graphs.*

Proof. We have already observed that the problem is in PSPACE. To show the PSPACE-hardness, we give a polynomial-time reduction from k-Coloring Reconfiguration on chordal graphs to k-CRCS.

Let (G, g_s, g_t) be an instance of k-Coloring Reconfiguration, where G is a chordal graph. We construct an instance (G', f_s, f_t) of k-CRCS as follows (see also Figure 4). For each vertex $v \in V(G)$, we add $(k-1)$ new vertices $v_1, v_2, \ldots, v_{k-1}$ and make $\{v, v_1, v_2, \ldots, v_{k-1}\}$ a clique in G'. Let $C_v = \{v, v_1, v_2, \ldots, v_{k-1}\}$. Note that the resulting graph G' is also chordal. We then define the k-coloring f_s as follows: for each $v \in V(G)$, set $f_s(v) = g_s(v)$, and for each vertex in $C_v \setminus \{v\}$, assign an arbitrary color distinct from $f_s(v)$ so that f_s is a proper k-coloring of G (which is always possible since $|C_v| = k$). Similarly, define $f_t(v) = g_t(v)$ for each $v \in V(G)$, and for each vertex in $C_v \setminus \{v\}$, assign an arbitrary color distinct from $g_t(v)$ so that f_t is a proper k-coloring of G.

This completes the construction of the instance (G', f_s, f_t). We then claim that (G, g_s, g_t) is a yes-instance of k-Coloring Reconfiguration if and only if (G', f_s, f_t) is a yes-instance of k-CRCS.

We first prove the "only if" direction. Suppose that (G, g_s, g_t) is a yes-instance of k-Coloring Reconfiguration. Then there exists a reconfiguration sequence $g_0, g_1, \ldots, g_\ell$ of proper k-colorings of G such that $g_0 = g_s$, $g_\ell = g_t$, and for each $i \in [\ell]$, g_{i-1} and g_i differ at exactly one vertex. For each $i \in \{0, 1, \ldots, \ell\}$, we construct a proper k-coloring f_i of G' from g_i, following the same construction as for f_s and f_t: for every $v \in V(G)$, set $f_i(v) = g_i(v)$, and for each $u \in C_v \setminus \{v\}$, assign an arbitrary color distinct from $f_i(v)$ so that f_i is a proper k-coloring of G'.

We claim that for all $i \in [\ell]$, f_{i-1} and f_i are reconfigurable under CS. Indeed, let $u \in V(G)$ be the unique vertex such that $g_{i-1}(u) \neq g_i(u)$. By construction, we have $f_{i-1}(u) \neq f_i(u)$, while $f_{i-1}(v) = f_i(v)$ for all $v \in V(G) \setminus \{u\}$. Let $w \in C_u$ be the unique vertex such that $f_{i-1}(w) = f_i(u)$. We construct a k-coloring f' from f_{i-1} by swapping the colors of u and w. Note that $f'(v) = f_i(v)$ for all $v \in V(G) \setminus \{u\}$.

Recall that for each $v \in V(G)$, the clique C_v in G' contains exactly k vertices, and in both colorings f_i and f', the vertices of C_v receive pairwise distinct colors. Since vertices in $C_v \setminus \{v\}$ are adjacent only to those in C_v, we can freely perform color swaps within C_v without affecting outside C_v. Thus, we can reconfigure f' into f_i by performing a sequence of color swaps only within cliques C_v for $v \in V(G)$. This implies that f_{i-1} and f_i are reconfigurable under CS. Applying this argument for each $i \in [\ell]$ yields a reconfiguration sequence from f_s to f_t in G' under CS. Therefore, (G', f_s, f_t) is a yes-instance of k-CRCS.

We now prove the "if" direction. Suppose that there exists a reconfiguration sequence $f_0, f_1, \ldots, f_\ell$ between f_s and f_t, where $f_0 = f_s$ and $f_\ell = f_t$. For each $i \in \{0, 1, \ldots, \ell\}$, define the coloring g_i of G by setting $g_i(v) = f_i(v)$ for every $v \in V(G)$. Since f_i is a proper k-coloring of G', the construction guarantees that g_i is a proper k-coloring of G.

Observe that each C_w ($w \in V(G)$) is a clique of size k. Thus, any color swap occurs only on two vertices within some clique C_w. It follows that for every $i \in [\ell]$, the colorings g_{i-1} and g_i are either identical or differ at exactly one vertex of G. By removing any consecutive duplicate colorings, we obtain a desired sequence of proper k-colorings of G from g_s to g_t. Therefore, (G, g_s, g_t) is a yes-instance of k-Coloring Reconfiguration.

This completes the proof of Theorem 4. ◀

3.3 Reduction from Nondeterministic Constraint Logic

In this subsection, we prove Theorem 5 by a polynomial-time reduction from the Nondeterministic Constraint Logic problem [19, 35].

▶ **Theorem 5.** *For every fixed integer $k \geq 3$, k-CRCS is* PSPACE*-complete for planar graphs of maximum degree 3 and bounded bandwidth.*

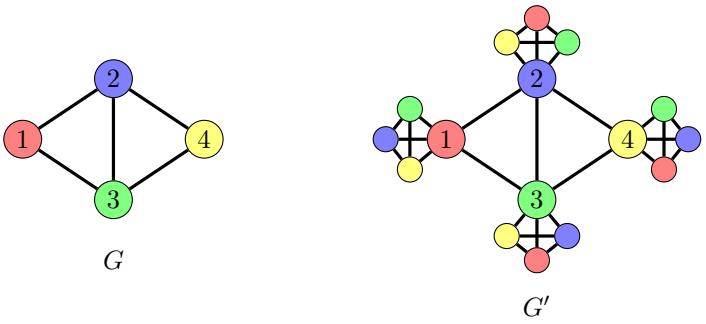


Figure 4 Construction of G from G' using four colors. Vertices of G are assigned a proper 4-coloring g_t, and those of G' the corresponding proper 4-coloring f_t.

We begin by formally defining the Nondeterministic Constraint Logic problem in Section 3.3.1. Next, we introduce an auxiliary gadget used in our reduction, which is described in Section 3.3.2. We then present the full reduction in Section 3.3.3, including the design of two types of gadgets: AND gadgets, and OR gadgets. Finally, we prove the correctness of the reduction in Section 3.3.4.

Note that our reduction is presented for $k = 3$; however, it holds analogously for cases where $k \geq 4$.

3.3.1 Definition of Nondeterministic Constraint Logic

A *Nondeterministic Constraint Logic (NCL) machine* is an undirected graph in which each edge is assigned a weight from $\{1, 2\}$ (see Figure 5 (a)). A *configuration* of an NCL machine is an orientation of its edges such that, at every vertex, the total weight of incoming edges is at least 2. Two configurations are said to be *adjacent* if they differ in the orientation of exactly one edge. In the Nondeterministic Constraint Logic problem, we are given an NCL machine M and two of its configurations, C_s and C_t. The goal is to determine whether there exists a sequence of configurations starting from C_s and ending at C_t, such that each consecutive pair of configurations in the sequence differs in the orientation of exactly one edge; that is, they are adjacent.

An NCL machine M is called an *AND/OR constraint graph* if it contains only two types of vertices: *AND vertices* and *OR vertices* (see again Figure 5 (a)). A vertex of degree 3 in M is an *AND vertex* if its incident three edges have weights 1, 1, and 2; see Figure 5 (b). Similarly, a vertex of degree 3 in M is defined as an *OR vertex* if all its incident three edges have weight 2; see Figure 5 (c). For the remainder of this paper, we will use the term *NCL machine* to refer specifically to an AND/OR constraint graph. It is known that the Nondeterministic Constraint Logic problem remains PSPACE-complete when the input NCL machine (AND/OR constraint graph) is restricted to be planar, of maximum degree 3, and of bounded bandwidth [35]. Recall that, for a graph G, the *bandwidth* of G is the minimum integer b such that G has a bijection $\pi\colon V(G) \to [|V(G)|]$ with $\max_{uv \in E(G)} |\pi(u) - \pi(v)| \leq b$.

3.3.2 Auxiliary Gadgets

Before presenting our full construction, we introduce auxiliary gadgets, called *forbidden pendants*, which prevent a vertex from being assigned a specific color.

Let $C = \{1, 2, 3\}$ be a set of colors. Consider a vertex x adjacent to a vertex y, where y is part of a 4-cycle, as illustrated in Figure 6. We consider two possible colorings of the 4-cycle in clockwise order starting from y: $(1, 2, 1, 3)$ or $(3, 1, 3, 2)$. Note that in each of these colorings, y is assigned color 1 or 3, respectively.

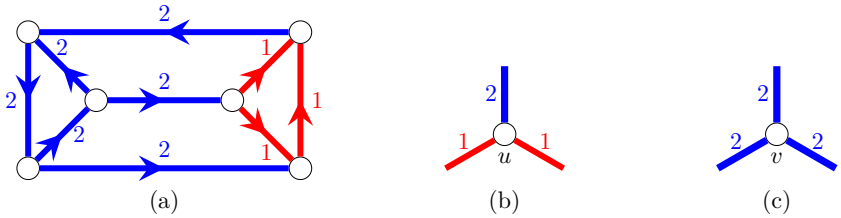


Figure 5 (a) A configuration of an NCL machine, (b) an AND vertex, and (c) an OR vertex. Edges of weight 2 are shown in blue lines, and edges of weight 1 in red lines. The NCL machine in (a) is an AND/OR constraint graph.

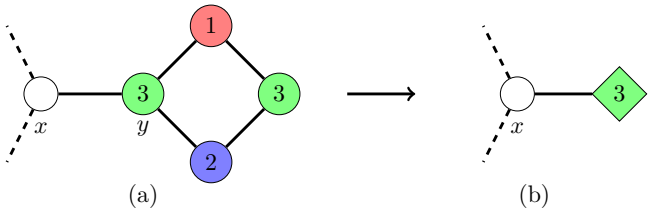


Figure 6 (a) An illustration of a 3-forbidden pendant for a vertex x, which ensures that x is never assigned color 3. (b) A simplified depiction of the gadget used to represent this pendant.

Since x is adjacent to y, it must be assigned a color different from that of y. In particular, x can be assigned only colors from the sets $\{2,3\}$ or $\{1,2\}$, respectively, depending on the color of y. Furthermore, observe that no valid color swaps can occur within the cycle or between x and y in any reconfiguration sequence. This ensures that the color of y remains fixed, and thus x is prevented from ever taking the same color as y.

If y is assigned color $c \in \{1,3\}$, we refer to this gadget as a *c-forbidden pendant* for x. For simplicity, we use the diagram shown in Figure 6 (b) to represent such a gadget.

3.3.3 AND/OR Gadgets and Our Reduction

Let $I = (M, C_s, C_t)$ be an instance of Nondeterministic Constraint Logic. Our reduction constructs a graph G by using two types of *vertex gadgets*, which simulate AND and OR vertices of M, respectively. Each vertex of M is replaced by the corresponding gadget according to its type.

For each edge $e = uv$ in M, the vertex gadgets for u and v each contain a special vertex, called a *port vertex*, which serves as an interface to the corresponding edge. These two port vertices, denoted by u_e and v_e, are connected by an edge referred to as a *port edge* (see Figure 7). Accordingly, the gadget corresponding to a vertex u of M with incident edges e_1, e_2, and e_3 includes three port vertices: u_{e_1}, u_{e_2}, and u_{e_3}.

Let u be an AND vertex in M, incident to one weight-2 edge e_1 and two weight-1 edges e_2 and e_3. We construct the corresponding AND gadget G_u, which consists of two internal vertices: u_0 and u_2^1, along with three port vertices: $u_1^1 = u_{e_1}$, $u_1^2 = u_{e_2}$, and $u_1^3 = u_{e_3}$. Each port vertex of G_u is adjacent to a 3-forbidden pendant (see Figure 8 (a)); hence, it can only be colored with color 1 or 2 in any proper coloring. Consequently, any color swap involving a port vertex in G_u must occur along the corresponding port edge.

Let u be an OR vertex in M, incident to three weight-2 edges e_1, e_2, and e_3. The corresponding OR gadget G_v consists of 12 vertices, denoted by v_j^i for $i \in [3]$ and $j \in [4]$. For each $i \in [3]$, the vertex v_4^i is a port vertex of G_v, that is, $v_4^i = v_{e_i}$. Moreover, every v_4^i is adjacent to a 3-forbidden pendant, while each of v_2^i and v_3^i is adjacent to a 1-forbidden pendant (see Figure 8(b)).

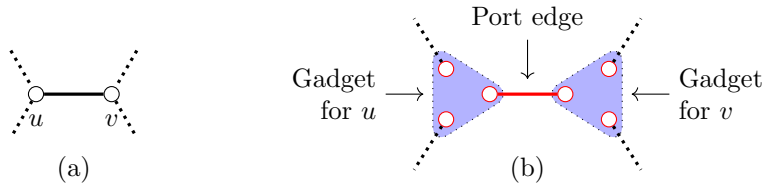


Figure 7 (a) An edge uv in G, and (b) its corresponding gadgets, where the port vertices are depicted by red circles and the shared port edge is depicted by a red line.

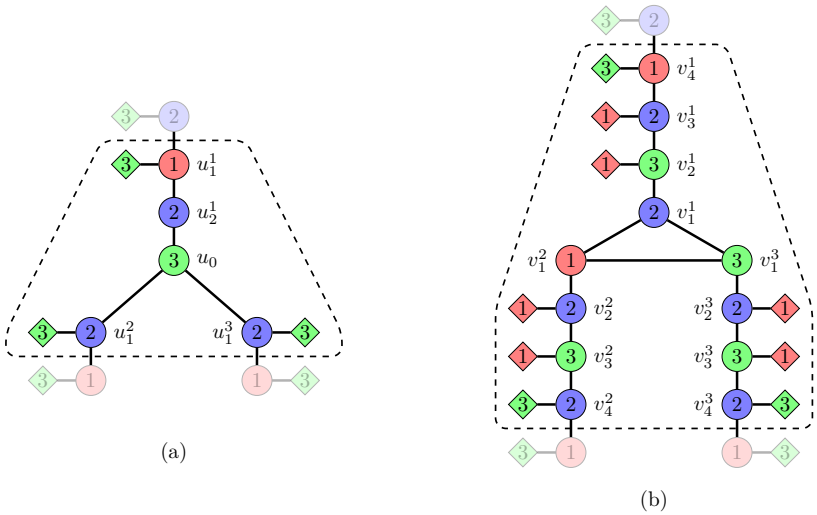


Figure 8 (a) The AND gadget and (b) the OR gadget, corresponding to (b) and (c) in Figure 5, respectively.

Similar to the AND gadget, each port vertex in G_v is restricted to colors 1 or 2 due to its adjacency to a 3-forbidden pendant. Consequently, any color swap involving a port vertex in G_v must occur along the corresponding port edge. Moreover, since both v_2^i and v_3^i for $i \in [3]$ are adjacent to 1-forbidden pendants, they must be assigned distinct colors: one must be colored 2 and the other 3. Thus, any color swap involving v_2^i or v_3^i can only occur on the edge $v_2^i v_3^i$.

Reduction

Let $I = (M, C_s, C_t)$ be an instance of Nondeterministic Constraint Logic. We construct a corresponding instance (G, f_s, f_t) of k-CRCS.

We begin by constructing a graph G from the NCL instance M as follows. For each AND or OR vertex in M, we replace it with the corresponding gadget as defined in Section 3.3.3. Then, for each edge $e = uv$ in M, we add a port edge between the corresponding port vertices u_e and v_e in the gadgets for u and v, respectively; see Figure 7.

Let G denote the resulting graph. Then, we observe the following.

▶ **Observation 6** ($\star$). *The constructed graph G is planar, of maximum degree* 3*, and has bounded bandwidth.*

We define two proper 3-colorings, f_s and f_t, of the constructed graph G. We begin by assigning colors to all c-forbidden pendants for $c \in \{1, 3\}$ according to the colorings described in Section 3.3.2.

Let e be an edge of M with an endpoint u. For each corresponding port vertex u_e, we set $f_s(u_e) = 1$ (resp. $f_t(u_e) = 1$) if the edge e is directed toward u in the configuration C_s (resp. C_t); otherwise, we assign $f_s(u_e) = 2$ (resp. $f_t(u_e) = 2$).

For each AND gadget G_u corresponding to an AND vertex u of M, we assign colors to the internal vertices u_2^1 and u_0 of G_u, which are depicted in Figure 8, so that f_s becomes a proper 3-coloring. That is, we set either $f_s(u_2^1) = 2$ and $f_s(u_0) = 3$, or $f_s(u_2^1) = 3$ and $f_s(u_0) = 2$. We define f_t similarly to f_s.

For each OR gadget G_v corresponding to an OR vertex v of M, we assign colors to the internal vertices of G_v depending on the coloring of its port vertices under f_s (resp. f_t). Specifically, for a given coloring of the port vertices, the internal vertices are colored according to one of the configurations illustrated in Figure 9, so that f_s (resp. f_t) becomes a proper 3-coloring. Since multiple valid internal colorings may exist for the same coloring of the port vertices of G_v, we may choose any such coloring arbitrarily.

This completes our polynomial-time reduction.

3.3.4 Correctness

Before proceeding to our proof, we provide an overview of the main ideas behind our reduction and outline the argument for its correctness.

Our reduction establishes a correspondence between the orientations of edges in a given instance of Nondeterministic Constraint Logic and the colorings in the constructed graph G. For each edge $e = uv$, we have associated two port vertices u_e and v_e in the gadgets for u and v, respectively. We interpret the edge as being oriented toward vertex v if u_e is assigned color 2, and as oriented toward vertex u if v_e is assigned color 2. Note that the coloring of each pair $\{u_e, v_e\}$ must be such that exactly one vertex is colored with 2 and the other with 1.

We briefly describe the behavior of the AND and OR gadgets. The AND gadget ensures that the port vertex u_1^1 can be colored with 2 only if both u_1^2 and u_1^3 are colored with 1. Once this condition is satisfied, the vertex u_2^1 can be recolored with 3 via a color swap with u_0.

Next, we explain the behavior of the OR gadgets. In an OR vertex of M, it suffices that at least one of the three incident edges is oriented inward. Accordingly, our gadget must only prohibit the configuration in which all three port vertices v_4^1, v_4^2, v_4^3 are simultaneously colored with 2. As shown in Figure 8(b), our construction enforces this constraint: if all three port vertices are colored with 2, then all intermediate vertices v_2^1, v_2^2, v_2^3 must also be colored with 2, which is impossible because the three vertices v_1^1, v_1^2, v_1^3 form a clique. Thus, at least one of the port vertices must be assigned color 1. See all proper colorings of the OR gadget shown in Figure 9.

To establish the correctness of our reduction, we present the following lemma.

▶ **Lemma 7** (⋆). *The instance (M, C_s, C_t) of* Nondeterministic Constraint Logic *is a yes-instance if and only if the constructed instance (G, f_s, f_t) of* 3-CRCS *is a yes-instance.*

4 Polynomial-time Algorithms

In this section, we present polynomial-time algorithms for CRCS and k-CRCS on paths, cographs, and split graphs.

4.1 Paths

We begin by presenting the following result for path graphs:

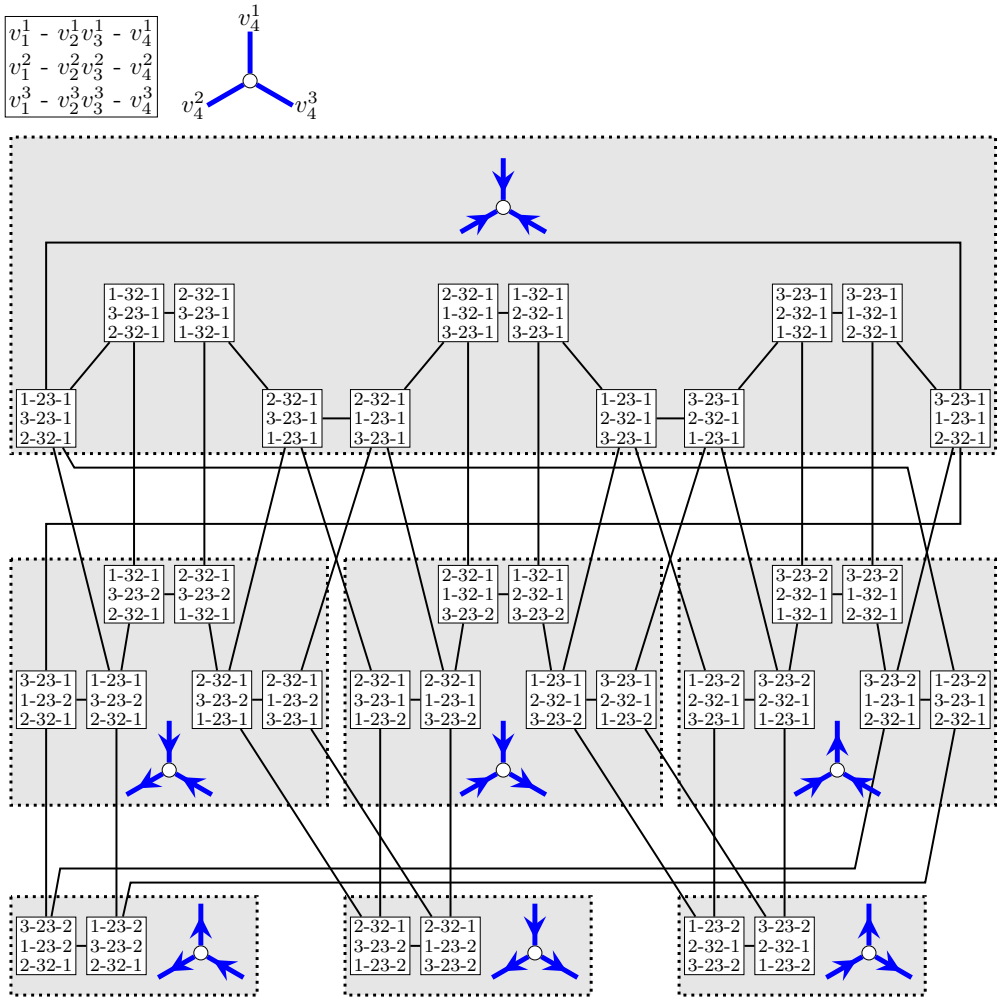


Figure 9 All valid orientations of the three edges incident to an OR vertex v, and the corresponding colorings of the OR gadget with their adjacency. Port vertices v_4^1, v_4^2, and v_4^3 are shown, and only the colors excluding those forbidden by c-forbidden pendants are indicated. In each row, no color swap is allowed between vertices separated by a hyphen due to the c-forbidden pendants.

▶ **Theorem 8.** 3-CRCS *can be solved in linear time for path graphs.*

To prove Theorem 8, we design a linear-time algorithm that solves 3-CRCS on path graphs.

In our algorithm, we compute and compare the invariants of the two input 3-colorings. If the invariants are identical, the algorithm returns `YES`; otherwise, it returns `NO`. Although the implementation of our algorithm is relatively simple, we emphasize that the core idea behind the algorithm is conceptually nontrivial, as it captures the essential structure preserved under CS.

4.1.1 Invariant for Coloring Strings

Before introducing the invariant, we first define several terms used throughout this subsection.

Given a string S, $S[i]$ denotes the i-th character of S, and $S[i, j]$ denotes the substring from the i-th to the j-th character (inclusive). If $i > j$, we define $S[i, j] := \mathsf{NIL}$, where NIL denotes the empty string (a string of length 0).

Let $P = v_1, v_2, \ldots, v_n$ be a path on n vertices. A string $S = S[1]S[2]\cdots S[n]$ is called a *coloring string* if there exists a proper 3-coloring f of P such that $S[i] = f(v_i)$ for all $i \in [n]$. We sometimes refer to S as the coloring string corresponding to f. Note that a coloring f can be encoded into its corresponding coloring string in $O(n)$ time.

We say that two consecutive characters in S are *swappable* if they can be exchanged to produce another coloring string S'. In this case, we say that S and S' are *adjacent*. If no such pair of swappable characters exists in S, then we say that S is *rigid*. Note that each swap of two characters precisely corresponds to a single color swapping operation. Finally, two coloring strings S and S' are said to be *reconfigurable* if there exists a sequence $S_0, S_1, \ldots, S_\ell$ of coloring strings such that $S_0 = S$, $S_\ell = S'$, and each consecutive pair S_{i-1}, S_i is adjacent for all $i \in [\ell]$.

We next observe a condition that determines whether two adjacent characters in a coloring string are swappable. Recall that a coloring string represents a proper 3-coloring of an n-vertex path.

▶ **Observation 9.** *Let $S = s_1 s_2 \cdots s_n$ be a coloring string. We can swap s_i and s_{i+1} in S for $i \in [n-1]$ if and only if the following conditions are satisfied:*

- *If $i = 1$, then the three characters s_1, s_2, s_3 are pairwise distinct.*
- *If $2 \le i \le n-2$, then $s_{i-1} = s_{i+2}$ holds.*
- *If $i = n-1$, then the three characters s_{n-2}, s_{n-1}, s_n are pairwise distinct.*

We now define the notion of *contraction*, which will be used to define our invariant for a coloring string. Let $S = s_1 s_2 \cdots s_n$ be a coloring string of length $n \ge 3$. We define the following three contraction operations:

(C1.) If $S = S[1, i-2]\, s_{i-1} s_i s_{i+1} s_{i+2}\, S[i+3, n]$ and $s_{i-1} = s_{i+2}$ for some $2 \le i \le n-2$, then S can be contracted to $S[1, i-2]\, s_{i+2}\, S[i+3, n]$.

(C2.) If s_1, s_2, s_3 are pairwise distinct, then $S = s_1 s_2 s_3\, S[4, n]$ can be contracted to $S[4, n]$.

(C3.) If s_{n-2}, s_{n-1}, s_n are pairwise distinct, then $S = S[1, n-3]\, s_{n-2} s_{n-1} s_n$ can be contracted to $S[1, n-3]$.

Note that in each of these cases, the contracted substring consists of three characters that are pairwise distinct.

Let $\mathsf{cont}(S)$ denote the set of all coloring strings that can be obtained from S by applying a single contraction operation. Each contraction reduces the length of the string by exactly 3, i.e., for every $S' \in \mathsf{cont}(S)$, we have $|S'| = |S| - 3$. We now define the invariant $\mathsf{inv}(S)$ of a coloring string S recursively as follows:

$$\mathsf{inv}(S) = \begin{cases} \mathsf{NIL} & \text{if } |S| \le 2, \\ \mathsf{inv}(S') & \text{if there exists } S' \in \mathsf{cont}(S), \\ S & \text{otherwise. (i.e., } S \text{ is rigid)} \end{cases}$$

The following lemma shows that $\mathsf{inv}(S)$ is uniquely determined regardless of the order in which the contractions are applied.

▶ **Lemma 10 (⋆).** *For any coloring string S, the invariant $\mathsf{inv}(S)$ is well-defined.*

A straightforward implementation of computing the invariant of a coloring string would take $O(n^2)$ time, as each contraction step may require scanning the entire string, and up to $O(n)$ such steps may be needed. However, the procedure can be implemented in linear time using a stack to efficiently simulate the recursive contractions.

▶ **Lemma 11 (⋆).** *For any coloring string S of length n, the invariant $\mathsf{inv}(S)$ can be computed in $O(n)$ time.*

4.1.2 Correctness of Algorithm

In the following, we discuss the correctness of our algorithm presented in Section 4.1.1. We begin with the following lemma, which states that taking a single swap of characters preserves the invariant.

▶ **Lemma 12** (⋆). *Let S and S' be two adjacent coloring strings. Then,* $\mathsf{inv}(S) = \mathsf{inv}(S')$.

The next two lemmas are crucial for our converse direction: if two coloring strings have the identical invariant, then they are reconfigurable.

▶ **Lemma 13** (⋆). *Let S be a coloring string of length at least 3. If S is not rigid, then there exists a coloring string S' such that $S'[1], S'[2], S'[3]$ are pairwise distinct, and S and S' are reconfigurable.*

▶ **Lemma 14** (⋆). *Let S and S' be coloring strings, and let w_1, w_2, w_3 and w'_1, w'_2, w'_3 be two triples of pairwise distinct characters such that $w_1w_2w_3S$ and $w'_1w'_2w'_3S'$ are coloring strings. If S and S' are reconfigurable, then $w_1w_2w_3S$ and $w'_1w'_2w'_3S'$ are also reconfigurable.*

The following is the main theorem in this subsection.

▶ **Theorem 15.** *Let (P, f_s, f_t) be a valid instance of* 3-CRCS *such that P is a path of n-vertices, and let S and S' be the coloring strings corresponding to f_s and f_t, respectively. Then,* $\mathsf{inv}(S) = \mathsf{inv}(S')$ *if and only if S and S' are reconfigurable.*

Proof. The "if" direction follows directly from Lemma 12. Hence, we now prove the "only-if" direction. We show that S and S' are reconfigurable if $\mathsf{inv}(S) = \mathsf{inv}(S')$, by induction on the length of S (and S').

For the base case where $|S| = |S'| \in \{0, 1, 2\}$, we always have $\mathsf{inv}(S) = \mathsf{inv}(S') = \mathsf{NIL}$, and it is clear that S and S' are reconfigurable.

For the induction step, assume that the claim holds for all coloring strings shorter than n. Now consider $n = |S| = |S'| \geq 3$. Since $\mathsf{inv}(S) = \mathsf{inv}(S')$, either both S and S' are rigid, or both are non-rigid. If both are rigid, then $S = \mathsf{inv}(S) = \mathsf{inv}(S') = S'$, so S and S' are identical and thus trivially reconfigurable.

Suppose S and S' are not rigid. By Lemma 13, there exist coloring strings S_x and S_y such that the first three characters of each string are pairwise distinct, and S is reconfigurable to S_x, while S' is reconfigurable to S_y. Note that S_x and S_y can be contracted to $S_x[4, n]$ and $S_y[4, n]$, respectively. Since $\mathsf{inv}(S_x[4, n]) = \mathsf{inv}(S_x) = \mathsf{inv}(S) = \mathsf{inv}(S') = \mathsf{inv}(S_y) = \mathsf{inv}(S_y[4, n])$, by the induction hypothesis, $S_x[4, n]$ and $S_y[4, n]$ are reconfigurable.

Applying Lemma 14, it follows that S_x and S_y are also reconfigurable. Consequently, by the transitivity of reconfigurability, S and S' are reconfigurable. ◀

4.2 Cographs

We begin by defining the class of *cographs*, also known as P_4-free graphs [15]. For two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, their *disjoint union* is defined as $G_1 \oplus G_2 = (V_1 \cup V_2, E_1 \cup E_2)$, and their *join* is defined as $G_1 \otimes G_2 = (V_1 \cup V_2, E_1 \cup E_2 \cup \{v_1v_2 \mid v_1 \in V_1, v_2 \in V_2\})$. A graph $G = (V, E)$ is a *cograph* if it can be constructed recursively according to the following rules:

1. A graph consisting of a single vertex is a cograph.
2. If G_1 and G_2 are cographs, then their disjoint union $G_1 \oplus G_2$ is also a cograph.
3. If G_1 and G_2 are cographs, then their join $G_1 \otimes G_2$ is also a cograph.

This inductive definition yields a canonical tree representation called a *cotree*, where each leaf corresponds to a vertex of the graph, and each internal node is labeled as either a disjoint union node ($\oplus$) or a join node ($\otimes$). Note that, by the definition of cographs, a cotree is a binary tree. For a cograph G, let T_G denote a cotree corresponding to G, and for a node $t \in V(T_G)$, let G_t denote the subgraph of G induced by the leaves of the subtree of T_G rooted at t. It is known that a cotree of a given cograph can be computed in linear time [34].

In this subsection, we provide a polynomial-time algorithm for CRCS on cographs.

▶ **Theorem 16.** CRCS *can be solved in polynomial time for cographs.*

4.2.1 Extended k-Colorings

To describe our algorithm, we introduce the notion of *extended colorings*. Let k be a positive integer. An *extended k-coloring* of a graph G is a map $f\colon V(G) \to [k] \cup \{*\}$ such that for every edge $uv \in E(G)$, either $f(u) \neq f(v)$ or $f(u) = f(v) = *$. That is, we allow a special flexible color $*$ in addition to the usual color set $[k]$, and adjacent vertices are permitted to share the color $*$. Given an extended k-coloring f of G, we define the set of *swappable colors* with respect to f as $S_f = \{c \in [k] \cup \{*\} \mid |f^{-1}(c)| = 1 \text{ or } (c = * \text{ and } f^{-1}(c) \neq \emptyset)\}$. Namely, a color c is swappable if it is used by exactly one vertex in G under f, or if $c = *$ and at least one vertex is assigned the color $*$ under f.

Our polynomial-time algorithm solves a generalized version of CRCS, which we call the Extended k-Coloring Reconfiguration (ECRCS) problem. In ECRCS, we are given a graph G and two extended k-colorings f_s and f_t of G, and the goal is to determine whether there exists a sequence of extended k-colorings that transforms f_s into f_t via color swaps. We use the terminology for ECRCS in the same way as for CRCS. Note that ECRCS generalizes CRCS in the following sense: for any instance (G, k, f_s, f_t) of CRCS, it holds that (G, k, f_s, f_t) is a yes-instance of CRCS if and only if it is a yes-instance of ECRCS. Furthermore, for any valid instance (G, k, f_s, f_t) of ECRCS, the initial and target colorings f_s and f_t must have the same set of swappable colors, i.e., $S_{f_s} = S_{f_t}$.

4.2.2 Polynomial-time Algorithm for Cographs

We now outline the strategy of our polynomial-time algorithm. Our approach is inspired by the polynomial-time algorithm for Token Sliding on cographs [22]. Indeed, an extended 1-coloring of a graph G naturally corresponds to an independent set of G, and thus our algorithm generalizes their result.

The algorithm proceeds recursively from the root to the leaves of the cotree T_G of the input cograph G, solving the ECRCS instance associated with each node t of T_G.

For a k-coloring f of G, let f^1 and f^2 denote the restrictions of f to $V(G_{t_1})$ and $V(G_{t_2})$, respectively. Suppose that we are given an instance (G, k, f_s, f_t) of ECRCS.

Leaf node. Let G be a cograph consisting of a single vertex v. Observe that (G, k, f_s, f_t) is a yes-instance of ECRCS if and only if $f_s(v) = f_t(v)$.

Union node. We consider the case where the root node of T_G is a union node with children t_1 and t_2. Since a union operation introduces no edges between $V(G_{t_1})$ and $V(G_{t_2})$, swaps involving vertices in $V(G_{t_1})$ and those in $V(G_{t_2})$ occur independently. Therefore, (G, k, f_s, f_t) is a yes-instance of ECRCS if and only if both $(G_{t_1}, k, f_s^1, f_t^1)$ and $(G_{t_2}, k, f_s^2, f_t^2)$ are yes-instances of ECRCS.

Join node. We now consider the case where the root node of T_G is a join node with children t_1 and t_2. Note that since t is a join node, all vertices in $V(G_{t_1})$ are adjacent to all vertices in $V(G_{t_2})$. Consequently, for any k-coloring of G, no color can appear in both f^1 and f^2.

We perform a case analysis based on whether any of the swappable color sets $S_{f_s^1}$, $S_{f_s^2}$, $S_{f_t^1}$, or $S_{f_t^2}$ is empty. We begin with the following observation.

▶ **Observation 17 (⋆).** *Let (G, k, f_s, f_t) be a yes-instance of* ECRCS*, where G is a cograph and the root node of T_G is a join node with children t_1 and t_2. Then, the following statements hold:*

1. *If at least one of $S_{f_s^1}$ or $S_{f_s^2}$ is empty, then no color swap between a vertex in $V(G_{t_1})$ and a vertex in $V(G_{t_2})$ occurs in any reconfiguration sequence from f_s to f_t.*
2. *$S_{f_s^1} = \emptyset$ if and only if $S_{f_t^1} = \emptyset$; similarly, $S_{f_s^2} = \emptyset$ if and only if $S_{f_t^2} = \emptyset$.*

We first consider the case where at least one of $S_{f_s^1}$ or $S_{f_s^2}$ is empty.

▶ **Lemma 18 (⋆).** *Let (G, k, f_s, f_t) be a valid instance of* ECRCS*, where G is a cograph and the root node of T_G is a join node with children t_1 and t_2. Suppose that at least one of $S_{f_s^1}$ or $S_{f_s^2}$ is empty. Then, (G, k, f_s, f_t) is a yes-instance of* ECRCS *if and only if both $(G_{t_1}, k, f_s^1, f_t^1)$ and $(G_{t_2}, k, f_s^2, f_t^2)$ are yes-instances of* ECRCS*.*

We then consider the remaining case. Let (G, k, f_s, f_t) be a valid instance of ECRCS, where G is a cograph and the root node of T_G is a join node with children t_1 and t_2. For each $j = 1, 2$, we define extended k-colorings $f_{s*}^j, f_{t*}^j \colon V(G_{t_j}) \to [k] \cup \{*\}$ as follows:

$$f_{s*}^j(v) = \begin{cases} * & \text{if } f_s^j(v) \in S_{f_s}, \\ f_s^j(v) & \text{otherwise} \end{cases}, \quad f_{t*}^j(v) = \begin{cases} * & \text{if } f_t^j(v) \in S_{f_t}, \\ f_t^j(v) & \text{otherwise} \end{cases}.$$

By construction, both f_{s*}^j and f_{t*}^j are extended k-colorings of G_{t_j}.

▶ **Lemma 19 (⋆).** *Let (G, k, f_s, f_t) be a valid instance of* ECRCS*, where G is a cograph and the root node of T_G is a join node with children t_1 and t_2. Suppose that both $S_{f_s^1}$ and $S_{f_s^2}$ are non-empty. Then, (G, k, f_s, f_t) is a yes-instance of* ECRCS *if and only if both $(G_{t_1}, k, f_{s*}^1, f_{t*}^1)$ and $(G_{t_2}, k, f_{s*}^2, f_{t*}^2)$ are yes-instances of* ECRCS*.*

Algorithm. The arguments for leaf, union, and join nodes naturally lead to a recursive algorithm for ECRCS on cographs. Since the set of swappable colors at each node of T_G and the corresponding extended colorings can be constructed in polynomial time, the overall algorithm runs in polynomial time. This concludes the proof of Theorem 16.

4.3 Split Graphs

Recall that CRCS is PSPACE-complete on split graphs, as shown in Theorem 2. In this subsection, we show a contrasting result: k-CRCS is solvable in polynomial time on split graphs when the number of colors k is a constant.

▶ **Theorem 20 (⋆).** CRCS *on split graphs admits a kernel with at most $k + 2k2^k$ vertices, where k is the number of colors of an input.*

Theorem 20 immediately leads to the following Corollary 21.

▶ **Corollary 21.** k-CRCS *can be solved in polynomial time for split graphs.*

5 Concluding Remarks

An intriguing direction is to determine the complexity of CRCS on graph classes that include paths. For example, does CRCS admit a polynomial-time algorithm on trees or on caterpillars? Another natural case is interval graphs, which are a subclass of chordal graphs where k-CRCS has been shown to be PSPACE-complete (Theorem 4). Since our algorithm for paths (Theorem 8) relies heavily on both the path structure and the restriction $k = 3$, it seems difficult to generalize our approach directly to broader classes. On the other hand, Token Sliding is known to be solvable in polynomial time on both trees [16] and interval graphs [4, 11]. Thus, as in the case of cographs (Theorem 16), it is conceivable that algorithms for CRCS could be derived from existing algorithms for Token Sliding.

References

1 Valentin Bartier, Nicolas Bousquet, Carl Feghali, Marc Heinrich, Benjamin Moore, and Théo Pierron. Recoloring planar graphs of girth at least five. *SIAM J. Discret. Math.*, 37(1):332–350, 2023. doi:10.1137/21M1463598.

2 Rémy Belmonte, Eun Jung Kim, Michael Lampis, Valia Mitsou, Yota Otachi, and Florian Sikora. Token sliding on split graphs. *Theory Comput. Syst.*, 65(4):662–686, 2021. doi:10.1007/S00224-020-09967-8.

3 Marthe Bonamy and Nicolas Bousquet. Recoloring bounded treewidth graphs. *Electron. Notes Discret. Math.*, 44:257–262, 2013. doi:10.1016/J.ENDM.2013.10.040.

4 Marthe Bonamy and Nicolas Bousquet. Token sliding on chordal graphs. In Hans L. Bodlaender and Gerhard J. Woeginger, editors, *Graph-Theoretic Concepts in Computer Science - 43rd International Workshop, WG 2017, Eindhoven, The Netherlands, June 21-23, 2017, Revised Selected Papers*, volume 10520 of *Lecture Notes in Computer Science*, pages 127–139. Springer, 2017. doi:10.1007/978-3-319-68705-6_10.

5 Marthe Bonamy, Nicolas Bousquet, Carl Feghali, and Matthew Johnson. On a conjecture of mohar concerning kempe equivalence of regular graphs. *J. Comb. Theory B*, 135:179–199, 2019. doi:10.1016/J.JCTB.2018.08.002.

6 Marthe Bonamy, Marc Heinrich, Takehiro Ito, Yusuke Kobayashi, Haruka Mizuta, Moritz Mühlenthaler, Akira Suzuki, and Kunihiro Wasa. Diameter of colorings under kempe changes. *Theor. Comput. Sci.*, 838:45–57, 2020. doi:10.1016/J.TCS.2020.05.033.

7 Édouard Bonnet, Tillmann Miltzow, and Pawel Rzazewski. Complexity of token swapping and its variants. *Algorithmica*, 80(9):2656–2682, 2018. doi:10.1007/S00453-017-0387-0.

8 Paul S. Bonsma and Luis Cereceda. Finding paths between graph colourings: PSPACE-completeness and superpolynomial distances. *Theor. Comput. Sci.*, 410(50):5215–5226, 2009. doi:10.1016/J.TCS.2009.08.023.

9 Paul S. Bonsma, Amer E. Mouawad, Naomi Nishimura, and Venkatesh Raman. The complexity of bounded length graph recoloring and CSP reconfiguration. In Marek Cygan and Pinar Heggernes, editors, *Parameterized and Exact Computation - 9th International Symposium, IPEC 2014, Wroclaw, Poland, September 10-12, 2014. Revised Selected Papers*, volume 8894 of *Lecture Notes in Computer Science*, pages 110–121. Springer, 2014. doi:10.1007/978-3-319-13524-3_10.

10 Nicolas Bousquet and Marc Heinrich. A polynomial version of Cereceda's conjecture. *J. Comb. Theory B*, 155:1–16, 2022. doi:10.1016/J.JCTB.2022.01.006.

11 Marcin Brianski, Stefan Felsner, Jedrzej Hodor, and Piotr Micek. Reconfiguring independent sets on interval graphs. In Filippo Bonchi and Simon J. Puglisi, editors, *46th International Symposium on Mathematical Foundations of Computer Science, MFCS 2021, August 23-27, 2021, Tallinn, Estonia*, volume 202 of *LIPIcs*, pages 23:1–23:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.MFCS.2021.23.

12 Charlie Carlson, Ewan Davies, Alexandra Kolla, and Will Perkins. Computational thresholds for the fixed-magnetization ising model. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 1459–1472. ACM, 2022. doi:10.1145/3519935.3520003.

13 Luis Cereceda, Jan van den Heuvel, and Matthew Johnson. Connectedness of the graph of vertex-colourings. *Discret. Math.*, 308(5-6):913–919, 2008. doi:10.1016/J.DISC.2007.07.028.

14 Luis Cereceda, Jan van den Heuvel, and Matthew Johnson. Finding paths between 3-colorings. *J. Graph Theory*, 67(1):69–82, 2011. doi:10.1002/JGT.20514.

15 Derek G. Corneil, H. Lerchs, and L. Stewart Burlingham. Complement reducible graphs. *Discret. Appl. Math.*, 3(3):163–174, 1981. doi:10.1016/0166-218X(81)90013-5.

16 Erik D. Demaine, Martin L. Demaine, Eli Fox-Epstein, Duc A. Hoang, Takehiro Ito, Hirotaka Ono, Yota Otachi, Ryuhei Uehara, and Takeshi Yamada. Linear-time algorithm for sliding tokens on trees. *Theor. Comput. Sci.*, 600:132–142, 2015. doi:10.1016/J.TCS.2015.07.037.

17 Tatsuhiko Hatanaka, Takehiro Ito, and Xiao Zhou. The coloring reconfiguration problem on specific graph classes. *IEICE Trans. Inf. Syst.*, 102-D(3):423–429, 2019. doi:10.1587/TRANSINF.2018FCP0005.

18 Robert A. Hearn and Erik D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theor. Comput. Sci.*, 343(1-2):72–96, 2005. doi:10.1016/j.tcs.2005.05.008.

19 Robert A. Hearn and Erik D. Demaine. *Games, puzzles and computation.* A K Peters, 2009.

20 Jan van den Heuvel. The complexity of change. In *Surveys in Combinatorics 2013*, volume 409 of *London Mathematical Society Lecture Note Series*, pages 127–160. Cambridge University Press, 2013. doi:10.1017/CBO9781139506748.005.

21 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.

22 Marcin Kaminski, Paul Medvedev, and Martin Milanic. Complexity of independent set reconfigurability problems. *Theor. Comput. Sci.*, 439:9–15, 2012. doi:10.1016/J.TCS.2012.03.004.

23 Kyozi Kawasaki. Diffusion constants near the critical point for time-dependent ising models. i. *Phys. Rev.*, 145:224–230, May 1966. doi:10.1103/PhysRev.145.224.

24 A. B. Kempe. On the geographical problem of the four colours. *American Journal of Mathematics*, 2(3):193–200, 1879.

25 Aiya Kuchukova, Marcus Pappik, Will Perkins, and Corrine Yap. Fast and slow mixing of the kawasaki dynamics on bounded-degree graphs. In Amit Kumar and Noga Ron-Zewi, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2024, August 28-30, 2024, London School of Economics, London, UK*, volume 317 of *LIPIcs*, pages 56:1–56:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.APPROX/RANDOM.2024.56.

26 Daniel Lokshtanov and Amer E. Mouawad. The complexity of independent set reconfiguration on bipartite graphs. *ACM Trans. Algorithms*, 15(1):7:1–7:19, 2019. doi:10.1145/3280825.

27 Bojan Mohar. *Kempe Equivalence of Colorings*, pages 287–297. Birkhäuser Basel, Basel, 2007. doi:10.1007/978-3-7643-7400-6_22.

28 Bojan Mohar and Jesús Salas. A new kempe invariant and the (non)-ergodicity of the Wang–Swendsen–Kotecký algorithm. *Journal of Physics A: Mathematical and Theoretical*, 42(22):225204, May 2009. doi:10.1088/1751-8113/42/22/225204.

29 Bojan Mohar and Jesús Salas. On the non-ergodicity of the Swendsen–Wang–Kotecký algorithm on the kagomé lattice. *Journal of Statistical Mechanics: Theory and Experiment*, 2010(05):P05016, May 2010. doi:10.1088/1742-5468/2010/05/P05016.

30 C. M. Mynhardt and S. Nasserasr. Reconfiguration of colourings and dominating sets in graphs: a survey. *CoRR*, abs/2003.05956, 2020. doi:10.48550/arXiv.2003.05956.

31 Naomi Nishimura. Introduction to reconfiguration. *Algorithms*, 11(4):52, 2018. doi:10.3390/A11040052.

32 Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, 1970. doi:10.1016/S0022-0000(70)80006-X.

33 Alan D. Sokal. The multivariate tutte polynomial (alias potts model) for graphs and matroids. In Bridget S. Webb, editor, *Surveys in Combinatorics, 2005: Invited lectures from the Twentieth British Combinatorial Conference, Durham, UK, July 2005*, volume 327 of *London Mathematical Society Lecture Note Series*, pages 173–226. Cambridge University Press, 2005.

34 Marc Tedder, Derek G. Corneil, Michel Habib, and Christophe Paul. Simpler linear-time modular decomposition via recursive factorizing permutations. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games*, volume 5125 of *Lecture Notes in Computer Science*, pages 634–645. Springer, 2008. doi:10.1007/978-3-540-70575-8_52.

35 Tom C. van der Zanden. Parameterized complexity of graph constraint logic. In *Proceedings of IPEC 2015*, pages 282–293, 2015. doi:10.4230/LIPICS.IPEC.2015.282.

36 Eric Vigoda. Improved bounds for sampling colorings. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 51–59. IEEE Computer Society, 1999. doi:10.1109/SFFCS.1999.814577.

37 Katsuhisa Yamanaka, Erik D. Demaine, Takehiro Ito, Jun Kawahara, Masashi Kiyomi, Yoshio Okamoto, Toshiki Saitoh, Akira Suzuki, Kei Uchizawa, and Takeaki Uno. Swapping labeled tokens on graphs. *Theor. Comput. Sci.*, 586:81–94, 2015. doi:10.1016/J.TCS.2015.01.052.

38 Katsuhisa Yamanaka, Takashi Horiyama, J. Mark Keil, David G. Kirkpatrick, Yota Otachi, Toshiki Saitoh, Ryuhei Uehara, and Yushi Uno. Swapping colored tokens on graphs. *Theor. Comput. Sci.*, 729:1–10, 2018. doi:10.1016/J.TCS.2018.03.016.