Complexity of Coloring Reconfiguration under Recolorability Constraints

Hiroki Osasa[1,a)] Akira Suzuki[1,b)] Takehiro Ito[1,c)] Xiao Zhou[1,d)]

Abstract: For an integer $k \geq 1$, k-coloring reconfiguration is one of the most well-studied reconfiguration problems, defined as follows: In the problem, we are given two (vertex-)colorings of a graph using k colors, and asked to transform one into the other by recoloring only one vertex at a time, while at all times maintaining a proper coloring. The problem is known to be PSPACE-complete if $k \geq 4$, and solvable for any graph in polynomial time if $k \leq 3$. In this paper, we introduce a recolorability constraint, which forbids some pairs of colors to be recolored directly. The recolorability constraint is given in terms of an undirected graph R such that each node in R corresponds to a color and each edge in R represents a pair of colors that can be recolored directly. Then, we show the hardness of the problem based on the structure of recolorability constraints R. The problem is PSPACE-complete if R has more than one cycle or a vertex of degree at least four.

1. Introduction

Recently, *reconfiguration problems* [11] have been intensively studied in the field of theoretical computer science. The problem arises when we wish to find a step-by-step transformation between two feasible solutions of a search problem such that all intermediate results are also feasible and each step conforms to a fixed reconfiguration rule, that is, an adjacency relation defined on feasible solutions of the original search problem. (See, e.g., a survey [16] and references in [7], [12].)

One of the most well-studied reconfiguration problems is based on the (vertex-)coloring search problem [1], [2], [3], [4], [6], [8], [9], [13], [17], defined as follows. In the k-coloring reconfiguration problem, we are given two proper k-colorings f_0 and f_r of the same graph G, and asked to determine whether there is a sequence $\langle f_0, f_1, \ldots, f_\ell \rangle$ of proper k-colorings of G such that $f_\ell = f_r$ and f_i can be obtained from f_{i-1} by recoloring only a single vertex in G for all $i \in \{1, 2, \ldots, \ell\}$. The complexity status of this reconfiguration problem has been clarified based on several "standard" measures (e.g., the number of colors [3], [6] and graph classes [2], [9], [17]) which are used well also for analyzing the original search problem.

In this paper, we propose a new measure to capture the hardness of coloring reconfiguration according to recoloring steps.

1.1 Our problem

For an integer $k \geq 1$, let C be the *color set* consisting of k colors $1, 2, \ldots, k$. Let G be a graph with vertex set $V(G)$ and edge set $E(G)$. Recall that a *k-coloring* f of G is a mapping $f : V(G) \to C$ such that $f(v) \neq f(w)$ holds for each edge $vw \in E(G)$.

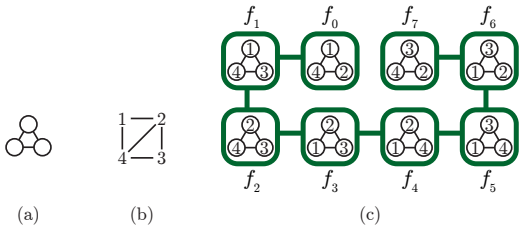

Fig. 1 (a) Input graph G, (b) a recolorability graph R with four colors 1, 2, 3 and 4, and (c) an $(f_0 \to f_7)$-reconfiguration sequence.

In this paper, we introduce the concept of "recolorability" and generalize the adjacency relation on k-colorings. The *recolorability* on the color set C is given in terms of an undirected graph R, called the *recolorability graph* on C, such that $V(R) = C$; each edge $ij \in E(R)$ represents a "recolorable" pair of colors $i, j \in V(R) = C$. Then, two k-colorings f and f' of G are *adjacent* (*under* R) if the following two conditions (a) and (b) hold:

(a) $\left|\{v \in V(G) : f(v) \neq f'(v)\}\right| = 1$, that is, f' can be obtained from f by *recoloring* a single vertex $v \in V(G)$; and

(b) if $f(v) \neq f'(v)$ for a vertex $v \in V(G)$, then $f(v)f'(v) \in E(R)$, that is, the colors $f(v)$ and $f'(v)$ form a recolorable pair.

Figure 1(c) shows eight different 4-colorings of the graph in Fig. 1(a). Then, for each $i \in \{1, 2, \ldots, 7\}$, two 4-colorings f_{i-1} and f_i are adjacent under the recolorability graph R in Fig. 1(b). As defined above, the known adjacency relation in [1], [2], [3], [4], [6], [8], [9], [13], [17] only requires the condition (a) above, that is, we can recolor a vertex from any color to any color directly. Observe that this corresponds to the case where R is a complete graph of size k, and hence our adjacency relation generalizes the known one.

Given a graph G, and two k-colorings f_0 and f_r of G, the color-

[1] Graduate School of Information Sciences, Tohoku University, Japan.
[a)] osawa@ecei.tohoku.ac.jp
[b)] a.suzuki@ecei.tohoku.ac.jp
[c)] takehiro@ecei.tohoku.ac.jp
[d)] zhou@ecei.tohoku.ac.jp

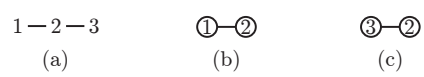

Fig. 2 (a) Recolorability graph R with three colors 1, 2 and 3, and (b) and (c) 3-colorings f_0 and f_r of a graph consisting of a single edge, respectively.

ING RECONFIGURATION problem UNDER R-RECOLORABILITY is the decision problem of determining whether there exists a sequence $\langle f_0, f_1, \ldots, f_\ell \rangle$ of k-colorings of G such that $f_\ell = f_r$ and f_{i-1} and f_i are adjacent under R for all $i \in \{1, 2, \ldots, \ell\}$; such a desired sequence is called an $(f_0 \to f_r)$-*reconfiguration sequence*. For example, the sequence $\langle f_0, f_1, \ldots, f_7 \rangle$ in Fig. 1(c) is an $(f_0 \to f_7)$-reconfiguration sequence.

We emphasize that the concept of recolorability graphs changes the situation drastically from k-COLORING RECONFIGURATION. For example, the $(f_0 \to f_7)$-reconfiguration sequence in Fig. 1(c) is a shortest one between f_0 and f_7 under the recolorability graph R in Fig. 1(b). However, in 4-COLORING RECONFIGURATION (in other words, if R would be K_4 and would have the edge joining colors 1 and 3), we can recolor the vertex from 1 to 3 directly. As another example, the instance illustrated in Fig. 2 is a no-instance for our problem even if the number of colors is larger than the number of vertices in an input graph (a single edge), but is clearly a yes-instance for 3-COLORING RECONFIGURATION.

1.2 Related results

As we have mentioned above, k-COLORING RECONFIGURATION has been studied intensively from various viewpoints.

From the viewpoint of the number k of colors in the color set C, a sharp analysis has been obtained: Bonsma and Cereceda [3] proved that k-COLORING RECONFIGURATION is PSPACE-complete if $k \geq 4$. On the other hand, Cereceda et al. [6] proved that k-COLORING RECONFIGURATION is solvable for any graph in polynomial time if $k \in \{1, 2, 3\}$, despite the fact that the original search problem (i.e., asking for the existence of one 3-coloring in a given graph) is NP-complete. In addition, for any yes-instance of 3-COLORING RECONFIGURATION, an $(f_0 \to f_r)$-reconfiguration sequence with the shortest length can be found in polynomial time [6], [13].

From the viewpoint of graph classes, Wrochna [17] proved that k-COLORING RECONFIGURATION remains PSPACE-complete even for graphs with bounded bandwidth (and hence bounded pathwidth). Bonamy et al. [2] gave some sufficient condition with respect to graph structures so that any pair of k-colorings of a graph has a reconfiguration sequence: for example, chordal graphs and chordal bipartite graphs satisfy their sufficient condition.

From the viewpoint of parameterized complexity, the length ℓ of a desired sequence is taken as a parameter for various reconfiguration problems [14]. Bonsma et al. [4] and Johnson et al. [13] independently developed a fixed-parameter algorithm to solve k-COLORING RECONFIGURATION when parameterized by $k + \ell$. In contrast, if the problem 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 [13].

As generalizations of k-COLORING RECONFIGURATION, reconfiguration problems for H-colorings [18] and circular colorings [5] have been studied. Note that both colorings are generalizations of k-colorings, and always form k-colorings of the same graph; but k-colorings do not always form these colorings. The two reconfiguration problems take the same adjacency relation as the original k-COLORING RECONFIGURATION (i.e., satisfying only the condition (a) in Section 1.1), but the set of feasible solutions does not always contain all k-colorings. On the other hand, our problem takes the same set of feasible solutions as the original k-COLORING RECONFIGURATION (i.e., *all* k-colorings), but takes different adjacency relation which obeys a recolorability graph R.

1.3 Our contribution

In this paper, we show the hardness of the COLORING RECONFIGURATION problem UNDER R-RECOLORABILITY based on the graph structure of recolorability graphs. We show the PSPACE-completeness of the problem for two cases. We first prove in Section 3.2 that the problem is PSPACE-complete for any recolorability graph with maximum degree at least four. We then show in Section 3.3 that the problem is PSPACE-complete for any recolorability graph R which contains a connected component R_C such that $|E(R_C)| > |V(R_C)|$, equivalently, for the case where the recolorability graph contains a connected component R_C having at least two cycles. This result implies that there exists a graph R of maximum degree three for which the problem remains PSPACE-complete. We note that our first result is strong in the sense that it shows PSPACE-completeness for *all* recolorability graphs of maximum degree at least four.

We omit some proofs from this extended abstract.

2. Preliminaries

Since we deal with (vertex-)coloring, we may assume without loss of generality that an input graph G is simple, connected and undirected. For a vertex $v \in V(G)$, let $N(G, v) = \{w \in V(G) : vw \in E(G)\}$. We say that a graph H is a *supergraph* of a graph G if both $V(G) \subseteq V(H)$ and $E(G) \subseteq E(H)$ hold; and hence H can be G itself.

For a graph G and a recolorability graph R on C, we define the *R-reconfiguration graph* on G, denoted by $C_R(G)$, as follows: $C_R(G)$ is an undirected graph such that each node of $C_R(G)$ corresponds to a k-coloring of G, and two nodes in $C_R(G)$ are joined by an edge if their corresponding k-colorings are adjacent under R. We sometimes call a node in $C_R(G)$ simply a k-coloring if it is clear from the context. A path in $C_R(G)$ from a k-coloring f to another one f' is called an $(f \to f')$-*reconfiguration sequence*. Note that any $(f \to f')$-reconfiguration sequence is *reversible*, that is, the path in $C_R(G)$ forms an $(f' \to f)$-reconfiguration sequence, too. Then, the COLORING RECONFIGURATION problem UNDER R-RECOLORABILITY is the decision problem of determining whether $C_R(G)$ contains an $(f_0 \to f_r)$-reconfiguration sequence. Note that the problem does not ask for an actual $(f_0 \to f_r)$-reconfiguration sequence as the output.

We introduce the concept of "frozen" vertices from the viewpoint of recoloring, which plays an important role in the paper. For a k-coloring f of a graph G and a recolorability graph R

on C, a vertex $v \in V(G)$ is said to be *frozen on* f (*under* R) if $f'(v) = f(v)$ holds for any k-coloring f' of G such that $C_R(G)$ has an $(f \to f')$-reconfiguration sequence.

3. PSPACE-completeness

In this section, we clarify the computational hardness of the problem from the viewpoint of recolorability graphs R. In Section 3.1, we introduce the list variant of the problem. Interestingly, the list variant is equivalent to the non-list one in our reconfiguration problem, and hence it suffices to construct reductions to the list variant. In Sections 3.2 and 3.3, we give our hardness results.

3.1 List recolorability

In the *list* variant, each vertex v of a graph G is associated with a subgraph $L_R(v)$ of the common recolorability graph R; we call $L_R(v)$ the *list recolorability of* v, and sometimes call the list assignment (mapping) L_R the *list* R*-recolorability*. Note that $L_R(v)$ is not necessarily a spanning subgraph of R. Let $k = |V(R)|$. Then, a k-coloring f of G is called a *list coloring* of G if $f(v) \in V(L_R(v))$ for all vertices v in G. Observe that for any supergraph R' of R, any list R-recolorability is also list R'-recolorability. We say that two list colorings f and f' are *adjacent under* L_R if they differ in exactly one vertex v such that $f(v)f'(v) \in E(L_R(v))$. Analogous to the R-reconfiguration graph, we define the L_R*-reconfiguration graph* on G, denoted by $C_{L_R}(G)$, as the undirected graph whose nodes correspond to list colorings of G, and two nodes in $C_{L_R}(G)$ are joined by an edge if their corresponding list colorings are adjacent under L_R.

Let G be an input graph with a list R-recolorability L_R. Then, for two list colorings f_0 and f_r of G, the COLORING RECONFIGURATION problem UNDER LIST R-RECOLORABILITY (the *list variant*, for short) is the decision problem of determining whether $C_{L_R}(G)$ contains an $(f_0 \to f_r)$-reconfiguration sequence. Observe that COLORING RECONFIGURATION UNDER R-RECOLORABILITY can be seen as the list variant such that $L_R(v) = R$ holds for every vertex $v \in V(G)$. Furthermore, note that $C_{L_R}(G)$ forms a subgraph of $C_R(G)$.

Interestingly, the list variant for our reconfiguration problem is equivalent to the non-list one, as in the following theorem.

Theorem 1. COLORING RECONFIGURATION UNDER LIST R-RECOLORABILITY *can be reduced to* COLORING RECONFIGURATION UNDER R-RECOLORABILITY *in time polynomial in* $|V(G)|$ *and* $|V(R)|$*, where* G *is an input graph of the list variant.*

Proof. Let G be an input graph for the list variant with a list R-recolorability L_R, and suppose that we are given two list colorings f_0 and f_r of G. Then, we construct a corresponding instance of COLORING RECONFIGURATION UNDER R-RECOLORABILITY; we denote by G' the corresponding graph, and by f'_0 and f'_r the corresponding initial and target k-colorings of G', respectively, where $k = |V(R)|$.

Indeed, we will give a gadget which forbids recoloring a vertex $v \in V(G)$ directly from a color i to another color j for each pair $ij \in E(R) \setminus E(L_R(v))$. Note that, for each color i in $V(R) \setminus V(L_R(v))$, we can add the vertex i to $L_R(v)$ as an isolated vertex (by adding the forbidding gadgets between i and all colors j such that $ij \in E(R)$). Then, since f_0 and f_r are list colorings of G, both $f_0(v) \neq i$ and $f_r(v) \neq i$ hold and hence v is never recolored to the isolated color i.

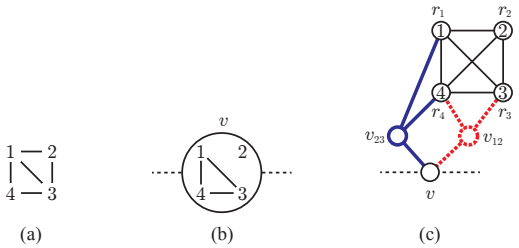

Fig. 3 (a) Recolorability graph R such that $L_R(v)$ is a subgraph of R for each vertex $v \in V(G)$, (b) a vertex $v \in V(G)$ whose list recolorability $L_R(v)$ is written inside, and (c) the vertex v in G', where the (red) thick dotted part corresponds to forbidding the pair of colors 1 and 2 in $E(R) \setminus E(L_R(v))$ and the (blue) thick part corresponds to forbidding the pair of colors 2 and 3 in $E(R) \setminus E(L_R(v))$.

To construct such a forbidding gadget, we will use a (newly added) clique of size $k = |V(R)|$ such that all vertices are colored with distinct colors. Notice that no vertex in the clique can be recolored to any color, that is, they are frozen vertices on the k-coloring. We use this property, and construct the corresponding instance, as follows.

We first add to G a new clique K_k of k vertices $r_1, r_2, \ldots, r_k$. Then, for each vertex $v \in V(G)$, consider any pair of colors i and j such that $ij \in E(R) \setminus E(L_R(v))$. We add a new vertex v_{ij} to G, and join it with v. In addition, we join v_{ij} with all vertices in $V(K_k) \setminus \{r_i, r_j\}$. (See Fig. 3(a)–(c) as an example of the application of this procedure.) Let G' be the resulting graph after applying the procedure above to all vertices $v \in V(G)$ and all pairs $ij \in E(R) \setminus E(L_R(v))$. For notational convenience, we denote by V_F the set of all vertices v_{ij} in G' that are newly added for each vertex $v \in V(G)$ and $ij \in E(R) \setminus E(L_R(v))$. We note that $V(G')$ is partitioned into $V(G)$, $V(K_k)$, and V_F. Furthermore, each vertex $v_{ij} \in V_F$ satisfies $N(G', v_{ij}) \cap V(G) = \{v\}$. We denote by v this unique vertex in $N(G', v_{ij}) \cap V(G)$ for each vertex $v_{ij} \in V_F$. Then, the corresponding k-colorings f'_0 and f'_r of G' are defined as follows: for each $l \in \{0, r\}$ and a vertex $w \in V(G')$,

$$f'_l(w) = \begin{cases} f_l(w) & \text{if } w \in V(G); \\ i & \text{if } w = r_i \in V(K_k); \\ j & \text{if } w = v_{ij} \in V_F \text{ and } f_l(v) = i; \text{ and} \\ i & \text{otherwise, that is, } w = v_{ij} \in V_F \text{ and } f_l(v) \neq i. \end{cases}$$

Then, all vertices $r_1, r_2, \ldots, r_k$ are frozen on both f'_0 and f'_r (indeed, under any recolorability graph). This completes the construction of the corresponding instance. This construction can be done in time polynomial in $|V(G)|$ and $k = |V(R)|$.

The correctness proof of our reduction is omitted from this extended abstract. □

Recall that for any supergraph R' of R, any list R-recolorability is also a list R'-recolorability, therefore we obtain the following corollary:

Corollary 2. *Let* R' *be an arbitrary supergraph of a recolorability graph* R*. Then,* COLORING RECONFIGURATION UNDER LIST R-

RECOLORABILITY *can be reduced to* COLORING RECONFIGURATION UNDER R'-RECOLORABILITY *in time polynomial in* $|V(G)|$ *and* $|V(R')|$*, where* G *is an input graph of the list variant.*

3.2 Recolorability graphs of maximum degree at least four

In this subsection, we consider the case where a recolorability graph is of maximum degree at least four. We emphasize again that the following theorem holds for an arbitrary recolorability graph as long as its maximum degree is at least four.

Theorem 3. *Let* R' *be any recolorability graph whose maximum degree is at least four. Then,* COLORING RECONFIGURATION UNDER R'-RECOLORABILITY *is PSPACE-complete.*

Proof. Observe that the problem can be solved in (most conveniently, nondeterministic [15]) polynomial space, and hence it is in PSPACE. Therefore, we show that the problem is PSPACE-hard for such a recolorability graph R'. Notice that, since R' is of maximum degree at least four, R' is a supergraph of a star $K_{1,4}$. Therefore, by Corollary 2 it suffices to prove that the list variant remains PSPACE-hard even for a list R-recolorability such that $R = K_{1,4}$. (See Fig. 4(a).) To show this, we give a polynomial-time reduction from 4-COLORING RECONFIGURATION, which is known to be PSPACE-complete [3].

Let G be an input graph for 4-COLORING RECONFIGURATION, and let f_0 and f_r be two given 4-colorings of G; let $C = \{1, 2, 3, 4\}$ be the color set. As a corresponding instance of the list variant, we take the same graph G in which the list recolorability $L_R(v)$ of each vertex $v \in V(G)$ is a star $K_{1,4}$ such that its center is a new color 5 and its leaves are the four colors $1, 2, 3$ and 4. Then, both f_0 and f_r are list colorings of the corresponding graph G, and we take the 4-colorings f_0 and f_r as the corresponding list colorings. This completes the construction of the corresponding instance, and hence it can be done in polynomial time.

The correctness proof of our reduction is omitted from this extended abstract. □

3.3 Recolorability graphs with more than one cycle

In this subsection, we consider the case where a recolorability graph R' contains a connected component having more than one cycle. Our result is expressed as follows:

Theorem 4. *Let* R' *be a recolorability graph which contains a connected component* R *such that* $|E(R)| > |V(R)|$*. Then,* COLORING RECONFIGURATION UNDER R'-RECOLORABILITY *is PSPACE-complete.*

To prove Theorem 4, by Corollary 2 it suffices to prove that the list variant remains PSPACE-hard for a list R-recolorability,

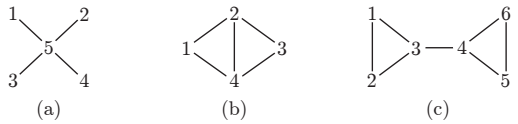

Fig. 4 (a) Recolorability graph $K_{1,4}$. In our reduction, the star $K_{1,4}$ with center color 5 is the list recolorability of all vertices $v \in V(G)$. (b) Recolorability graph which is a diamond graph. (c) Recolorability graph which is a $2K_3 + e$ graph.

where R is a connected component in R' such that $|E(R)| > |V(R)|$. We first characterize the structure of R by two small graphs: A graph is called a *diamond* graph if it can be obtained by deleting exactly one edge from a complete graph K_4 of size four (see Fig. 4(b)); a $2K_3 + e$ graph is a graph obtained by adding exactly one edge to disjoint union of two triangles K_3 (see Fig. 4(c).) We have the following lemma.

Lemma 5. *Let* R *be a connected graph such that* $|E(R)| > |V(R)|$*. Then,* R *satisfies at least one of the following statements*:

(a) R *has a vertex whose degree is at least four*;
(b) R *is a supergraph of some subdivision of a diamond graph*; *and*
(c) R *is a supergraph of some subdivision of a* $2K_3 + e$ *graph.*

If Lemma 5(a) holds for the recolorability graph R, then COLORING RECONFIGURATION UNDER R-RECOLORABILITY is PSPACE-complete by Theorem 3. Therefore, it suffices to prove the PSPACE-hardness for the other cases, that is, the recolorability graph R is either (b) any subdivision of a diamond graph, or (c) any subdivision of a $2K_3 + e$ graph.

We now give a sketch of our proof. We first prove that the list variant remains PSPACE-hard for a list R-recolorability when R is either a diamond graph or a $2K_3 + e$ graph (without subdivisions). We use these claims as the bases of inductive proofs for the cases (b) and (c). In this extended abstract, we only prove the base for the case (b), as follows.

Lemma 6. *Let* D *be a diamond graph. Then,* COLORING RECONFIGURATION UNDER LIST D-RECOLORABILITY *is PSPACE-complete.*

Proof. We give a polynomial-time reduction from NONDETERMINISTIC CONSTRAINT LOGIC (NCL, for short) [10], defined as follows. An NCL "machine" is specified by an undirected graph together with an assignment of weights from $\{1, 2\}$ to each edge of the graph. An (*NCL*) *configuration* of this machine is an orientation (direction) of the edges such that the sum of weights of in-coming arcs at each vertex is at least two. Figure 5(a) illustrates a configuration of an NCL machine, where each weight-2 edge is depicted by a thick (blue) line and each weight-1 edge by a thin (orange) line. Then, two NCL configurations are *adjacent* if they differ in a single edge direction. Given an NCL machine and its two configurations, it is known to be PSPACE-complete to determine whether there exists a sequence of adjacent NCL configurations which transforms one into the other [10].

In fact, the problem remains PSPACE-complete even for AND/OR *constraint graphs*, which consist only of two types of vertices, called "NCL AND vertices" and "NCL OR vertices." A vertex of degree three is called an *NCL* AND *vertex* if its three incident edges have weights 1, 1 and 2. (See Fig. 5(b).) An NCL AND vertex u behaves as a logical AND, in the following sense: the weight-2 edge can be directed outward for u if and only if both two weight-1 edges are directed inward for u. Note that, however, the weight-2 edge is not necessarily directed outward even when both weight-1 edges are directed inward. A vertex of degree three is called an *NCL* OR *vertex* if its three incident edges

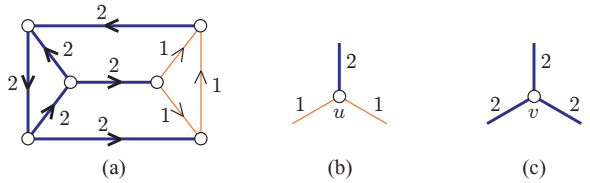

Fig. 5 (a) A configuration of an NCL machine, (b) NCL AND vertex u, and (c) NCL OR vertex v.

have weights 2, 2 and 2. (See Fig. 5(c).) An NCL OR vertex v behaves as a logical OR: one of the three edges can be directed outward for v if and only if at least one of the other two edges is directed inward for v. It should be noted that, although it is natural to think of NCL AND/OR vertices as having inputs and outputs, there is nothing enforcing this interpretation; especially for NCL OR vertices, the choice of input and output is entirely arbitrary because an NCL OR vertex is symmetric. For example, the NCL machine in Fig. 5(a) is an AND/OR constraint graph. From now on, we call an AND/OR constraint graph simply an *NCL machine*, and call an edge in an NCL machine an *NCL edge*.

Gadgets.

We first subdivide every NCL edge vw into a path $vv'w'w$ of length three by adding two new vertices v' and w'; the newly added vertices v' and w' are called *connectors* for v and w, respectively. (See Fig. 6(a) and (b).) We call the edge $v'w'$ a *link edge* between two NCL vertices v and w, and call the edges vv' and ww' *NCL one-third edges* for v and w, respectively. Notice that every vertex in the resulting graph belongs to exactly one of stars $K_{1,3}$ such that the center v of each $K_{1,3}$ corresponds to an NCL AND/OR vertex and the three leaves are connectors for v. Furthermore, these stars are all mutually disjoint, and joined together by link edges. (See Fig. 6(c) as an example.)

Therefore, our reduction involves constructing three types of gadgets which correspond to link edges and stars of NCL AND/OR vertices. In our gadgets, all connectors v' for NCL AND/OR vertices v have the same list recolorability $L_{\mathsf{D}}(v')$ such that $V(L_{\mathsf{D}}(v')) = \{2, 4\}$ and $E(L_{\mathsf{D}}(v')) = \{24\}$. Then, in our reduction, assigning the color 4 to v' always corresponds to directing the NCL one-third edge vv' from v' to v (i.e., the inward direction for v), while assigning the color 2 to v' always corresponds to directing vv' from v to v' (i.e., the outward direction for v).

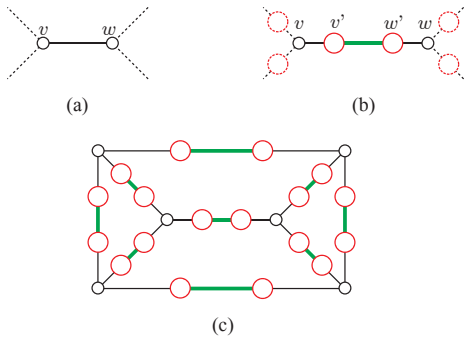

Fig. 6 (a) An NCL edge vw, (b) its subdivision into a path $vv'w'w$, and (c) the resulting graph which corresponds to the NCL machine in Fig. 5(a), where each connector is depicted by a (red) large circle and each link edge by a thin (green) line.

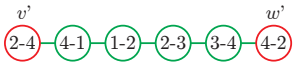

Fig. 7 The link edge gadget $G_{v'w'}$ between two connectors v' and w'.

(i) Link edge gadget.

Figure 7 illustrates our link edge gadget $G_{v'w'}$ for each link edge $v'w'$, where v' and w' are connectors for NCL AND/OR vertices v and w, respectively. The graph in each vertex (circle) indicates the list recolorability of the vertex. Recall that, in a given NCL machine, v and w are joined by a single NCL edge. Therefore, the link edge gadget should be consistent with the orientations of the NCL edge, as follows (see also Fig. 8(a) and (b)): If we assign 4 to v' (the inward direction for v), then w' must be colored with 2 (the outward direction for w); conversely, v' must be colored with 2 if we assign 4 to w'. In particular, the gadget must forbid a list coloring which assigns 4 to both v' and w' (the inward directions for both v and w), because such a list coloring corresponds to the direction which contributes to both v and w illegally. On the other hand, assigning 2 to both v' and w' (the outward directions for both v and w) corresponds to the *neutral* orientation of the NCL edge vw which contributes to neither v nor w, and hence we simply do not care such an orientation.

Figure 8(c) illustrates the L_{D}-reconfiguration graph $C_{L_{\mathsf{D}}}(G_{v'w'})$ on the link edge gadget $G_{v'w'}$. Each rectangle corresponds to a node of $C_{L_{\mathsf{D}}}(G_{v'w'})$, that is, a list coloring of $G_{v'w'}$, where the underlined bold number represents the color assigned to the vertex. Then, $C_{L_{\mathsf{D}}}(G_{v'w'})$ is connected, and there is no list coloring which assigns 4 to both v' and w', as claimed above. Furthermore, the reversal of the NCL edge vw can be simulated by the path on $C_{L_{\mathsf{D}}}(G_{v'w'})$ via the neutral orientation of vw, as illustrated in Fig. 8(c). Thus, this gadget works correctly as a link edge.

(ii) AND gadget.

Figure 9 illustrates our AND gadget G_{and} for each NCL AND vertex v, where v_a, v_b and v_c correspond to the three connectors for v. In the figure, the connectors v_a and v_b come from the two weight-1 NCL edges, while the connector v_c comes from the weight-2 NCL edge. We now explain this gadget works as an NCL AND vertex. Similarly as for the link edge gadget, the AND gadget must forbid the case where all the connectors v_a, v_b and v_c are colored with 2 at the same time (i.e., all NCL one-third edges vv_a, vv_b and vv_c take the outward direction for v). In addition, the gadget must simulate the following situation: v_c can be colored with 2 (i.e., the weight-2 edge vv_c can take the outward direction for v) only when both v_a and v_b are colored with 4 at the same time (i.e., both the weight-1 edges vv_a and vv_b take the inward direction for v).

Figure 10(a) illustrates all feasible orientations of the three NCL one-third edges vv_a, vv_b and vv_c, whose corresponding assignments of colors to the connectors are depicted in Fig. 10(b). Due to the space limitation, in Fig. 10(b), we only indicate the colors assigned to v_a, v_c and v_b, but Fig. 10(c) shows all list (proper) colorings of G_{and} that assign the colors 2, 4 and 4 to v_a, v_c and v_b, respectively. Then, as illustrated in Fig. 10(c), these list colorings are "internally connected," that is, any two list colorings are reconfigurable with each other without recoloring any connector of G_{and}. Furthermore, this gadget preserves the "ex-

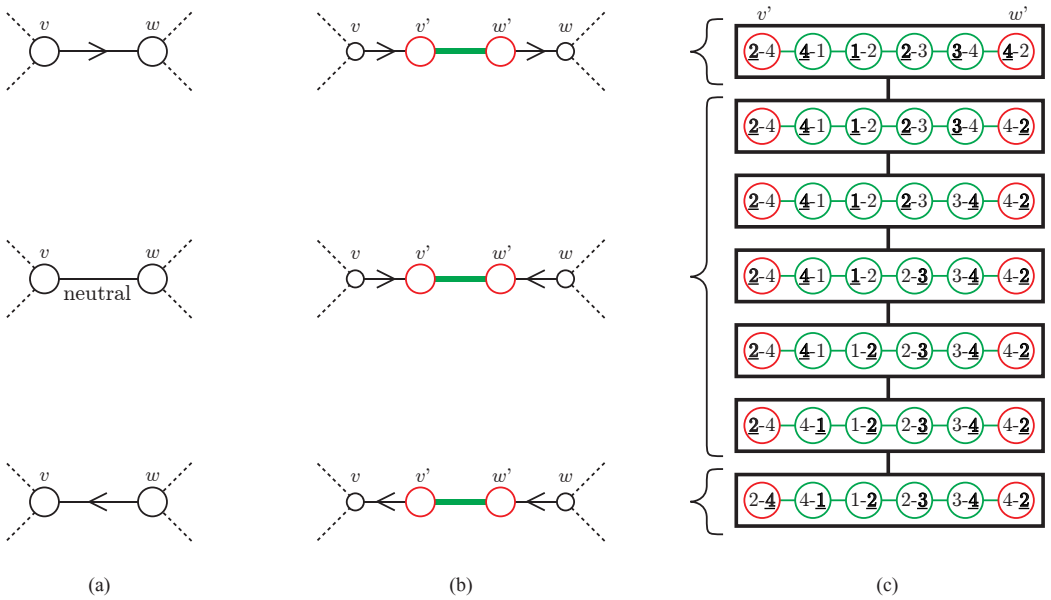

Fig. 8 (a) Three orientations of an NCL edge vw, (b) their corresponding orientations of the NCL one-third edges vv' and ww', and (c) all list colorings of the link edge gadget $G_{v'w'}$ in the L_{D}-reconfiguration graph $C_{L_{\mathsf{D}}}(G_{v'w'})$.

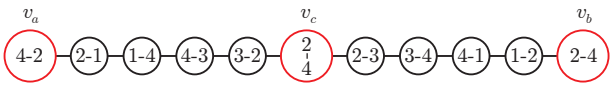

Fig. 9 AND gadget G_{and} with three connectors v_a, v_b and v_c.

ternal adjacency" in the following sense: if we contract the list colorings in $C_{L_{\mathsf{D}}}(G_{\mathsf{and}})$ having the same color assignments to the connectors into a single vertex, then the resulting graph is exactly the graph depicted in Fig. 10(a). We have checked by a computer search that these two properties hold for our AND gadget. Therefore, we can conclude that our AND gadget correctly works as an NCL AND vertex.

(iii) OR **gadget.**

Figure 11 illustrates our OR gadget G_{or} for each NCL OR vertex v, where v_x, v_y and v_z correspond to the three connectors for v. We now explain this gadget works as an NCL OR vertex. For each NCL OR vertex v, it suffices that at least one of the three NCL edges take the inward direction for v. Thus, the OR gadget must forbid only the case where all the connectors v_x, v_y and v_z are colored with 2 at the same time. Indeed, our gadget in Fig. 11 forbids such the case, because otherwise all three vertices v_{b1}, v_{b2} and v_{b3} in Part B must be colored with 4 and this yields that there is no available color for vertices in Part A.

Similarly as for the AND gadget, we have checked by a computer search that our OR gadget is internally connected and preserves the external adjacency. Therefore, we can conclude that our OR gadget correctly works as an NCL OR vertex.

Reduction.

As we have mentioned above, we first subdivide every NCL edge vw into a path $vv'w'w$ of length three by adding two connectors v' and w'. (See Fig. 6.) Then, we replace each of link edges and NCL AND/OR vertices with its corresponding gadget; let G be the resulting graph. In addition, we construct two list colorings of G which correspond to two given configurations C_0 and C_r of the NCL machine. Note that there are (in general, exponentially) many list colorings which correspond to the same NCL configuration. However, by the construction of the three gadgets, no two distinct NCL configurations correspond to the same list coloring of G. We thus choose any two list colorings f_0 and f_r of G which correspond to C_0 and C_r, respectively. This completes the construction of the corresponding instance for the list variant under list D-recolorability. This construction can be done in polynomial time.

We omit the correctness proof of our reduction from this extended abstract. □

Acknowledgments This work is partially supported by JST CREST Grant Number JPMJCR1402, and JSPS KAKENHI Grant Numbers JP16K00003, JP16K00004 and JP17K12636, Japan.

References

[1] M. Bonamy and N. Bousquet. Recoloring bounded treewidth graphs. In *the 7th Latin-American Algorithms, Graphs and Optimization Symposium* (*LAGOS 2013*), volume 44 of *Electronic Notes in Discrete Mathematics*, pages 257–262, 2013.

[2] M. Bonamy, M. Johnson, I. Lignos, V. Patel, and D. Paulusma. Reconfiguration graphs for vertex colourings of chordal and chordal bipartite graphs. *Journal of Combinatorial Optimization*, 27:132–143, 2014.

[3] P. S. Bonsma and L. Cereceda. Finding paths between graph colourings: PSPACE-completeness and superpolynomial distances. *Theoretical Computer Science*, 410:5215–5226, 2009.

[4] P. S. Bonsma, A. E. Mouawad, N. Nishimura, and V. Raman. The complexity of bounded length graph recoloring and CSP reconfiguration. In *the 9th International Symposium on Parameterized and Exact Computation* (*IPEC 2014*), volume 8894 of *Lecture Notes in Computer Science*, pages 110–121, 2014.

[5] R. C. Brewster, S. McGuinness, B. Moore, and J. A. Noel. A dichotomy theorem for circular colouring reconfiguration. *Theoretical Computer Science*, 639:1–13, 2016.

[6] L. Cereceda, J. van den Heuvel, and M. Johnson. Finding paths between 3-colorings. *Journal of Graph Theory*, 67:69–82, 2011.

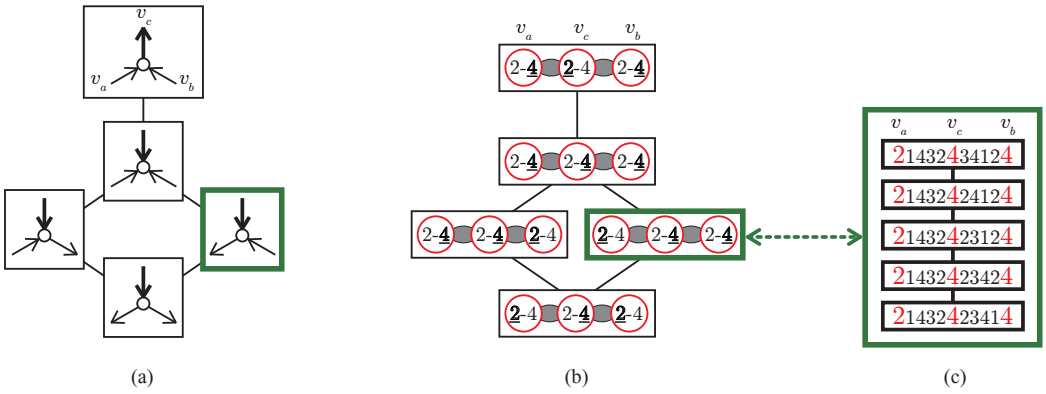

Fig. 10 (a) All feasible orientations of the three NCL one-third edges incident to an NCL AND vertex together with their adjacency, (b) image of L_{D}-reconfiguration graph $C_{L_{\mathsf{D}}}(G_{\mathsf{and}})$ on the AND gadget G_{and}, and (c) the inside of the rightmost (green) thick box in the image which corresponds to assigning the colors 2, 4 and 4 to v_a, v_c and v_b, respectively, where we simply write the colors assigned to G_{and} by a sequence of colors.

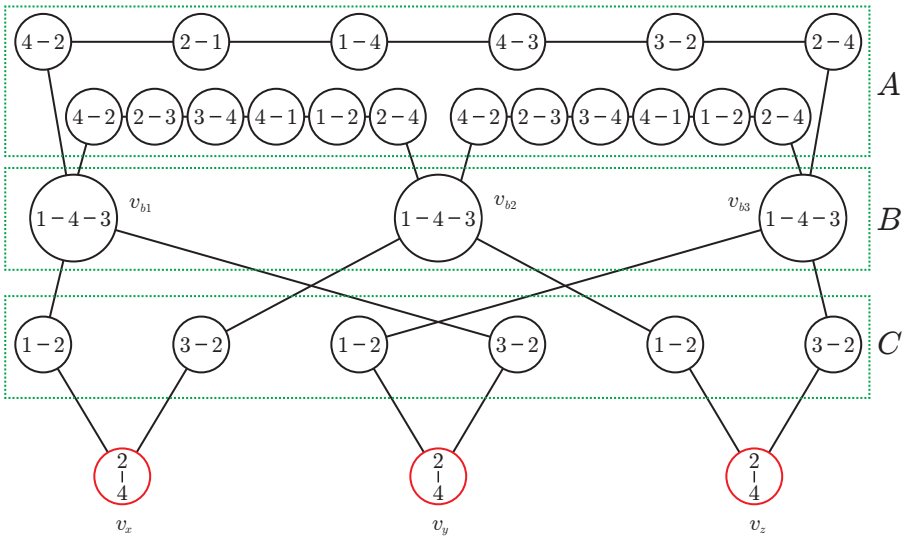

Fig. 11 OR gadget G_{or} with three connectors v_x, v_y and v_z.

[7] E. D. Demaine, M. L. Demaine, E. Fox-Epstein, D. A. Hoang, T. Ito, H. Ono, Y. Otachi, R. Uehara, and T. Yamada. Linear-time algorithm for sliding tokens on trees. *Theoretical Computer Science*, 600:132–142, 2015.

[8] C. Feghali, M. Johnson, and D. Paulusma. A reconfigurations analogue of brooks' theorem and its consequences. *Journal of Graph Theory*, 83:340–358, 2016.

[9] T. Hatanaka, T. Ito, and X. Zhou. The list coloring reconfiguration problem for bounded pathwidth graphs. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 98-A:1168–1178, 2015.

[10] R. A. Hearn and E. D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343:72–96, 2005.

[11] T. Ito, E. D. Demaine, N. J. Harvey, C. H. Papadimitriou, M. Sideri, R. Uehara, and Y. Uno. On the complexity of reconfiguration problems. *Theoretical Computer Science*, 412:1054–1065, 2011.

[12] T. Ito, H. Ono, and Y. Otachi. Reconfiguration of cliques in a graph. In *the 12th Annual Conference on Theory and Applications of Models of Computation* (*TAMC 2015*), volume 9076 of *Lecture Notes in Computer Science*, pages 212–223, 2015.

[13] M. Johnson, D. Kratsch, S. Kratsch, V. Patel, and D. Paulusma. Finding shortest paths between graph colourings. *Algorithmica*, 75:295–321, 2016.

[14] A. E. Mouawad, N. Nishimura, V. Raman, N. Simjour, and A. Suzuki. On the parameterized complexity of reconfiguration problems. *Algorithmica*, 78:274–297, 2017.

[15] W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4:177–192, 1970.

[16] J. van den Heuvel. The complexity of change. In *Surveys in Combinatorics 2013*, volume 409 of *London Mathematical Society Lecture Note Series*, pages 127–160, 2013.

[17] M. Wrochna. Reconfiguration in bounded bandwidth and treedepth. *CoRR*, abs/1405.0847, 2014.

[18] M. Wrochna. Homomorphism reconfiguration via homotopy. In *the 32nd International Symposium on Theoretical Aspects of Computer Science* (*STACS 2015*), volume 30 of *Leibniz International Proceedings in Informatics*, pages 730–742, 2015.