The Coloring Reconfiguration Problem on Specific Graph Classes

Tatsuhiko Hatanaka[1,a)] Takehiro Ito[1,b)] Xiao Zhou[1,c)]

Abstract: We study the problem of transforming one (vertex) k-coloring of a graph into another one by changing only one vertex color assignment at a time, while at all times maintaining a k-coloring, where k denotes the number of colors. This decision problem is known to be PSPACE-complete even for bipartite graphs and any fixed constant $k \geq 4$. In this paper, we study the problem from the viewpoint of graph classes. We first show that the problem remains PSPACE-complete for chordal graphs even if the number of colors is a fixed constant. We then demonstrate that, even when the number of colors is a part of input, the problem is solvable in polynomial time for several graph classes, such as split graphs and trivially perfect graphs.

Keywords: Combinatorial reconfiguration, graph algorithm, vertex coloring

1. Introduction

Recently, *reconfiguration problems* [13] have been intensively studied in the field of theoretical computer science. These problems model several "dynamic" situations where we wish to find a step-by-step transformation between two feasible solutions of a combinatorial (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. This framework has been applied to several well-studied combinatorial problems, including SATISFIABILITY, INDEPENDENT SET, VERTEX COVER, DOMINATING SET, and so on. (See, e.g., a survey [12] and references in [9].)

1.1 Our problem

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

Let $C = \{c_1, c_2, \ldots, c_k\}$ be the set of k colors. Throughout the paper, k denotes the number of colors in C. A (proper) *k-coloring* of a graph $G = (V, E)$ is a mapping $f : V \to C$ such that $f(v) \neq f(w)$ for every edge $vw \in E$. Figure 1 illustrates four 4-colorings of the same graph G; the color assigned to each vertex is attached to the vertex.

Suppose that we are given two k-colorings f_0 and f_r of a graph G (e.g., the leftmost and rightmost ones in Fig. 1), and we are asked whether we can transform one into the other via k-colorings of G such that each differs from the previous one in only one vertex color assignment. This decision problem is called the COLORING RECONFIGURATION problem. For the particular instance of Fig. 1, the answer is "yes" as illustrated in the figure, where the vertex whose color assignment was changed from the previous one is depicted by a black circle. We emphatically write k-COLORING RECONFIGURATION when the number k of colors is fixed, that is, k is not a part of input.

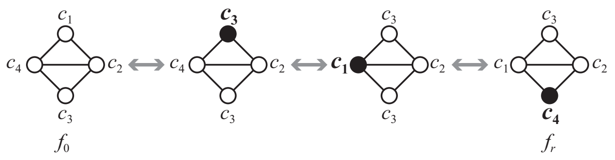


Fig. 1 A reconfiguration sequence between two 4-colorings f_0 and f_r of G.

1.2 Known and related results

COLORING RECONFIGURATION is one of the most well-studied reconfiguration problems from various viewpoints [1], [2], [3], [4], [5], [6], [7], [8], [11], [14], [16], [17], including the parameterized complexity [4], [14], (in)tractability with respect to graph classes [3], [5], [16], generalized variants such as the list coloring variant [3], [11], [16], the H-coloring variant [17] and the circular coloring variant [6].

Bonsma and Cereceda [3] proved that k-COLORING RECONFIGURATION is PSPACE-complete even for (i) bipartite graphs and any fixed $k \geq 4$, (ii) planar graphs and any k, $4 \leq k \leq 6$, and (iii) bipartite planar graphs and $k = 4$. On the other hand, Cereceda et al. [8] gave a polynomial-time algorithm to solve COLORING RECONFIGURATION for any graph and $k \leq 3$. Thus, the complexity status of COLORING RECONFIGURATION is analyzed sharply with respect to k.

Because the problem remains PSPACE-complete even for very restricted instances, some sufficient conditions have been proposed so that any pair of k-colorings of a graph has a desired transformation [1], [3], [7]; in other words, if a given instance

[1] Graduate School of Information Sciences, Tohoku University, Aobayama 6-6-05, Sendai 980-8579, Japan.
a) hatanaka@ecei.tohoku.ac.jp
b) takehiro@ecei.tohoku.ac.jp
c) zhou@ecei.tohoku.ac.jp

satisfies one of sufficient conditions, then it is a yes-instance (but, the opposite direction does not necessarily hold.) Bonsma and Cereceda [3] proved that if k is at least the degeneracy of a graph G plus two, then there is a desired transformation between any pair of k-colorings of G. Bonamy et al. [1] gave some sufficient condition with respect to graph structures: for example, chordal graphs and chordal bipartite graphs satisfy their sufficient condition.

Recently, Bonsma and Paulusma [5] gave a polynomial-time algorithm to solve COLORING RECONFIGURATION for $(k-2)$-connected chordal graphs; note that k is not necessarily a constant in their algorithm. They posed an open question which asks whether the problem is solvable in polynomial time for all chordal graphs.

1.3 Our contribution

In this paper, we study COLORING RECONFIGURATION from the viewpoint of graph classes. More specifically, we first show that k-COLORING RECONFIGURATION remains PSPACE-complete for chordal graphs; note that k is some fixed constant. Therefore, we answer the open question posed by Bonsma and Paulusma [5]. We then demonstrate that COLORING RECONFIGURATION is solvable in polynomial time for several graph classes, even when k is a part of input; such graph classes include split graphs and trivially perfect graphs.

2. Preliminaries

In this section, we define some basic terms and notation.

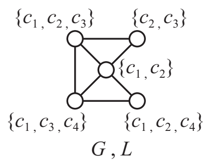
Let $G = (V, E)$ be a graph with vertex set V and edge set E; we sometimes denote by $V(G)$ and $E(G)$ the vertex set and the edge set of G, respectively. For a vertex v in G, we denote by $N(G, v)$ and $\mathsf{deg}(G, v)$ the neighborhood $\{w \in V \mid vw \in E\}$ and the degree $|N(G, v)|$ of v in G, respectively. We denote by $\omega(G)$ the size of a maximum clique in G.

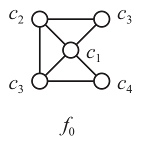
2.1 LIST COLORING RECONFIGURATION

In this subsection, we formally define COLORING RECONFIGURATION. Because we sometimes use the notion of list colorings, we define it as a special case of LIST COLORING RECONFIGURATION as follows.

In list coloring, each vertex $v \in V(G)$ of a graph G has a set $L(v) \subseteq C = \{c_1, c_2, \ldots, c_k\}$ of colors, called the *list of* v; we sometimes call the list assignment $L : V \to 2^C$ itself a *list*. Then, a k-coloring f of G is called an *L-coloring* of G if $f(v) \in L(v)$ holds for every vertex $v \in V(G)$. Thus, a k-coloring of G is an L-coloring of G when $L(v) = C$ holds for every vertex v in G, and hence L-coloring is a generalization of k-coloring.

For two L-colorings f and f' of a graph G, a *reconfiguration sequence* between f and f' is a sequence $\langle f_p, f_{p+1}, \ldots, f_q\rangle$ of L-colorings of G such that $f_p = f$, $f_q = f'$, and $|\{v \in V(G)\colon f_{i-1}(v) \neq f_i(v)\}| = 1$ holds for each $i \in \{p+1, p+2, \ldots, q\}$. Note that any reconfiguration sequence is *reversible*, that is, $\langle f_q, f_{q-1}, \ldots, f_p\rangle$ is a reconfiguration sequence between f' and f. We say that two L-colorings f and f' are *reconfigurable* if there is a reconfiguration sequence between them. Then, the LIST COLORING RECONFIGURATION problem is defined as follows:





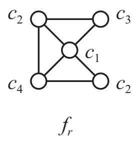


Fig. 2 Example for frozen vertices: The upper three vertices are frozen on f_0 and f_r because they form a clique of size three, and their lists contain only three colors in total.

Input: A graph G, a list L, two L-colorings f_0 and f_r of G
Question: Determine whether f_0 and f_r are reconfigurable or not.

Note that LIST COLORING RECONFIGURATION is a decision problem, and hence does not require the specification of an actual reconfiguration sequence.

We denote by a 4-tuple (G, L, f_0, f_r) an instance of LIST COLORING RECONFIGURATION. COLORING RECONFIGURATION is indeed LIST COLORING RECONFIGURATION when restricted to the case where $L(v) = C$ holds for every vertex v in an input graph G. We thus simply denote by a 4-tuple (G, k, f_0, f_r) an instance of COLORING RECONFIGURATION, and by a triple (G, f_0, f_r) an instance of k-COLORING RECONFIGURATION; recall that k is fixed in the latter case.

2.2 Frozen vertices

In this subsection, we introduce the notion of "frozen vertices."

Let f be an L-coloring of a graph G with a list L. Then, a vertex $v \in V(G)$ is said to be *frozen on* f if $f'(v) = f(v)$ holds for every L-coloring f' of G which is reconfigurable from f. Therefore, v cannot be recolored in any reconfiguration sequence. Thus, (G, L, f_0, f_r) is a no-instance if $f_0(v) \neq f_r(v)$ holds for at least one frozen vertex v on f_0 or f_r. By the definition, a frozen vertex v on an L-coloring f stays frozen on any L-coloring which is reconfigurable from f.

Generally speaking, it is not easy to characterize such frozen vertices for a given L-coloring. However, there is a simple sufficient condition for which a vertex is frozen, as follows. (See Fig. 2 as an example of Observation 1.)

Observation 1. *Let G be a graph with a list L, and assume that G contains a clique V_Q of size q. If $|\bigcup_{v \in V_Q} L(v)| = q$, then all vertices $v \in V_Q$ are frozen on any L-coloring of G.*

3. PSPACE-completeness

A graph is *chordal* if it contains no induced cycle of length at least four. In this section, we prove the following theorem.

Theorem 1. *There exists a fixed constant k' such that k-*COLORING RECONFIGURATION *is PSPACE-complete for chordal graphs and every $k \geq k'$.*

It is known that k-COLORING RECONFIGURATION belongs to PSPACE [3]. Therefore, as a proof of Theorem 1, we show that there exists a fixed constant k' such that k-COLORING RECONFIGURATION is PSPACE-hard for chordal graphs and any $k \geq k'$, by giving a polynomial-time reduction from LIST COLORING RECONFIGURATION [16].

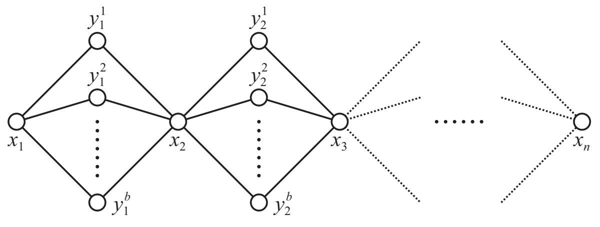


Fig. 3 Graph H.

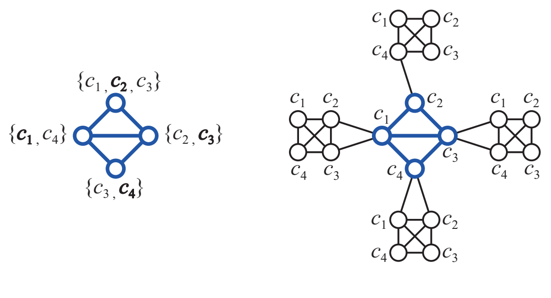


Fig. 4 (a) A graph H', a list L and an L-coloring g_0, and (b) a constructed graph G and k-coloring f_0.

3.1 List coloring reconfiguration

Wrochna [16] proved that there exist two constants b and m such that list coloring reconfiguration remains PSPACE-complete even when an input instance (H, L, g_0, g_r) satisfies the following conditions (see also Fig. 3):

(a) $H = (X \cup Y, E)$ is a bipartite graph with bipartition X and Y such that $X = \{x_1, x_2, \dots, x_n\}$, $Y = \{y_i^j \mid 1 \le i \le n-1,\ 1 \le j \le b\}$, and $E = \{x_i y_i^j,\ y_i^j x_{i+1} \mid 1 \le i \le n-1,\ 1 \le j \le b\}$;

(b) the list $L(v)$ of each vertex $v \in V(H)$ is a subset of the color set $C_1 \cup C_2$ such that $C_1 \cap C_2 = \emptyset$ and $|C_1| = |C_2| = m$;

(c) $L(x_i) = C_1$ if i is odd, $L(x_i) = C_2$ otherwise; and

(d) $L(y) \subseteq C_1 \cup C_2$ for all $y \in Y$.

The graph H above can be modified to an interval graph (and hence a chordal graph) H' by adding an edge $x_i x_{i+1}$ for each $i \in \{1, 2, \dots, n-1\}$. This modification does not affect the existence and the reconfigurability of L-colorings, because any two vertices x_i and x_{i+1} joined by the new edge have distinct lists C_1 and C_2. We note in passing that this modification gives the following theorem. For an integer $d \ge 0$, a graph G is *d-degenerate* if every subgraph H of G has at least one vertex v such that $\mathsf{deg}(H, v) \le d$.

Theorem 2. List coloring reconfiguration *is PSPACE-complete for 2-degenerate interval graphs.*

3.2 Reduction

We then construct an instance (G, f_0, f_r) of k-coloring reconfiguration from the instance (H', L, g_0, g_r) above of list coloring reconfiguration, as follows.

Let $k \ge k' = |C_1 \cup C_2| = |\bigcup_{u \in V(H')} L(u)| = 2m$. For each vertex $u \in V(H')$, we introduce a complete graph W_u with k vertices, which is called a *frozen clique gadget*. (See Fig. 4 as an example, where $k = 4$.) The vertices in W_u are labeled as $w_1^u, w_2^u, \dots, w_k^u$, and each vertex w_i^u corresponds to the color c_i for each $i \in \{1, 2, \dots, k\}$. We denote by W the set of all vertices in frozen clique gadgets, that is, $W = \bigcup_{u \in V(H')} V(W_u)$.

We next add an edge between $u \in V(H')$ and $w_i^u \in V(W_u)$ if and only if $L(u)$ does *not* contain color c_i. The constructed graph G is chordal, because the addition of frozen clique gadgets does not produce any induced cycle with length at least four.

Finally, we define f_0 and f_r, as follows:

$$f_0(v) = \begin{cases} c_i & \text{if } v = w_i^u \in V(W_u) \text{ for some } u \in V(H'); \\ g_0(v) & \text{otherwise,} \end{cases}$$

and

$$f_r(v) = \begin{cases} c_i & \text{if } v = w_i^u \in V(W_u) \text{ for some } u \in V(H'); \\ g_r(v) & \text{otherwise.} \end{cases}$$

Therefore, we have $f_0(v) = f_r(v)$ for all vertices $v \in W$. From the construction, we note that both f_0 and f_r are proper k-colorings of G.

This completes our construction of the corresponding instance (G, f_0, f_r) of k-coloring reconfiguration. This construction can be done in polynomial time.

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

4. Polynomial-Time Solvable Cases

In this section, we demonstrate that coloring reconfiguration can be solved in polynomial time for some graph classes, even when the number k of colors is a part of input.

We indeed consider split graphs and trivially perfect graphs, both of which are subclasses of chordal graphs. The following sufficient condition for yes-instances on chordal graphs will play an important role.

Lemma 1 ([1]). *Let (G, k, f_0, f_r) be an instance of* coloring reconfiguration *such that G is a chordal graph. If $\omega(G) \le k - 1$, then it is a* yes*-instance.*

4.1 Split graphs

In this subsection, we consider split graphs. A graph is *split* if its vertex set can be partitioned into a clique and an independent set.

Theorem 3. Coloring reconfiguration *can be solved in linear time for split graphs.*

Proof. We give such a linear-time algorithm for split graphs. Let $\mathcal{I} = (G, k, f_0, f_r)$ be a given instance of coloring reconfiguration such that G is split. We first obtain a partition of $V(G)$ into a clique V_Q and an independent set V_I such that V_Q has the maximum size $\omega(G)$. Such a partition can be obtained in linear time [10]. Because f_0 and f_r are proper k-colorings of G, we have $|V_Q| = \omega(G) \le k$. Therefore, there are two cases to consider.

Case 1: $|V_Q| < k$.

In this case, $|V_Q| = \omega(G) \le k - 1$ holds. Since G is split and hence is a chordal graph, Lemma 1 implies that $\mathcal{I}$ is a yes-instance.

Case 2: $|V_Q| = k$.

In this case, every vertex in V_Q is frozen on f_0 and f_r. Thus, $\mathcal{I}$ is a no-instance if there exists a vertex $u \in V_Q$ such that $f_0(u) \neq f_r(u)$. Otherwise, because V_I is an independent set and both f_0 and f_r are proper k-colorings of G, we can directly recolor each vertex $w \in V_I$ from $f_0(w)$ to $f_r(w)$; $\mathcal{I}$ is a yes-instance.

We finally estimate the running time of our algorithm. We can obtain desired subsets V_Q and V_I in linear time [10]. Then, the algorithm simply checks if $|V_Q| < k$, and if $f_0(u) = f_r(u)$ holds for every vertex $u \in V_Q$. Therefore, our algorithm runs in linear time. □

4.2 Trivially perfect graphs

In this subsection, we consider trivially perfect graphs. The class of trivially perfect graphs has many characterizations. We here give its recursive definition. For two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, their *union* $G_1 \cup G_2$ is the graph such that $V(G_1 \cup G_2) = V_1 \cup V_2$ and $E(G_1 \cup G_2) = E_1 \cup E_2$, while their *join* $G_1 \vee G_2$ is the graph such that $V(G_1 \vee G_2) = V_1 \cup V_2$ and $E(G_1 \vee G_2) = E_1 \cup E_2 \cup \{vw : v \in V_1, w \in V_2\}$. Then, a *trivially perfect graph* can be recursively defined, as follows:

(1) a graph consisting of a single vertex is a trivially perfect graph;
(2) if G_1 and G_2 are trivially perfect graphs, then their union $G_1 \cup G_2$ is a trivially perfect graph; and
(3) if G_1 and G_2 are trivially perfect graphs such that G_2 consists of a single vertex u, then their join $G_1 \vee G_2$ is a trivially perfect graph.

Notice that, by the join operation (3) above, the single vertex u in G_2 becomes a universal vertex in $G_1 \vee G_2$.

Theorem 4. Coloring reconfiguration *can be solved in linear time for trivially perfect graphs.*

Proof. We give such a linear-time algorithm for trivially perfect graphs. Since any trivially perfect graph G is a cograph, we can represent G by a binary tree, called a cotree, which can be naturally obtained from the recursive definition of trivially perfect graphs: a *cotree* $T = (V_T, E_T)$ of a trivially perfect graph G is a binary tree such that each leaf of T corresponds to a single vertex in G, and each internal node of T has exactly two children and is labeled with either union $\cup$ or join $\vee$; notice that, for each join node in T, one of the two children must be a leaf of T. Such a cotree of G can be constructed in linear time [15]. Each node $i \in V_T$ corresponds to a subgraph G_i of G which is induced by all vertices corresponding to the leaves of T that are the descendants of i in T. Clearly, $G = G_0$ for the root 0 of T.

We note that the maximum clique sizes $\omega(G_i)$ for all $i \in V_T$ can be computed in linear time, by a bottom-up computation according to the cotree T, as follows:

$$\omega(G_i) = \begin{cases} 1 & \text{if } i \text{ is a leaf of } T; \\ \max\{\omega(G_x), \omega(G_y)\} & \text{if } i \text{ is a union node} \\ & \quad \text{with children } x \text{ and } y; \\ \omega(G_x) + 1 & \text{if } i \text{ is a join node} \\ & \quad \text{with children } x \text{ and } y \\ & \quad \text{such that } y \text{ is a leaf of } T. \end{cases}$$

Therefore, we assume without loss of generality that we are given a trivially perfect graph G together with its cotree $T = (V_T, E_T)$ such that the maximum clique size $\omega(G_i)$ is associated to each node $i \in V_T$.

Let $\mathcal{I} = (G, k, f_0, f_r)$ be a given instance of coloring reconfiguration such that G is a trivially perfect graph. For each node $i \in V_T$ and a k-coloring f of G, we denote by f^i the k-coloring of the subgraph G_i such that $f^i(v) = f(v)$ holds for every $v \in V(G_i)$. We propose the following algorithm to solve the problem, and will prove its correctness.

Input: An instance $\mathcal{I} = (G, k, f_0, f_r)$ of coloring reconfiguration such that G is a trivially perfect graph
Output: yes/no as the answer to $\mathcal{I}$

Step 1. If $|V(G)| = 1$ or $\omega(G) < k$, then return yes.
Step 2. In this step, G has more than one vertex, and hence the root of the cotree T is either a union node or a join node. Let x and y be two children of the root of T. Then, we execute either (a) or (b):
- **Case (a):** The root is a union node. Return yes if both (G_x, k, f_0^x, f_r^x) and (G_y, k, f_0^y, f_r^y) are yes-instances; otherwise return no.
- **Case (b):** The root is a join node. Assume that G_y consists of a single vertex u. Return no if $f_0(u) \neq f_r(u)$; otherwise return the answer to $(G_x, k-1, f_0^x, f_r^x)$.

We first verify the correctness of Step 1. If $|V(G)| = 1$, then we can directly recolor the vertex w in G from $f_0(w)$ to $f_r(w)$; thus, $\mathcal{I}$ is a yes-instance. If $\omega(G) < k$, then Lemma 1 yields that $\mathcal{I}$ is a yes-instance because G is a trivially perfect graph and hence is a chordal graph. Thus, Step 1 correctly returns yes.

We then verify the correctness of Step 2(a). This step is executed when the root of T is a union node. Then, there is no edge between G_x and G_y. Therefore, it suffices to solve each of (G_x, k, f_0^x, f_r^x) and (G_y, k, f_0^y, f_r^y), and combine their answers. Thus, Step 2(a) works correctly.

We finally verify the correctness of Step 2(b). This step is executed when the root of T is a join node. In addition, $\omega(G) = k$ holds because it is executed after Step 1. Since $u \in V(G_y)$ becomes a universal vertex in $G = G_x \vee G_y$, it is contained in any maximum clique in G. Since $\omega(G) = k$ holds, u is frozen on f_0 and f_r. Thus, if $f_0(u) \neq f_r(u)$, then $\mathcal{I}$ is a no-instance. Otherwise no vertex in $V(G) \setminus \{u\}$ can use the color $f_0(u) = f_r(u)$ in any reconfiguration sequence, because u is a universal vertex in G and is frozen on f_0 and f_r. Therefore, (G, k, f_0, f_r) is a yes-instance if and only if $(G_x, k-1, f_0^x, f_r^x)$ is a yes-instance. Thus, Step 2(b) works correctly.

Although the algorithm above is written as a recursive function, it can be implemented so as to run in linear time, as follows: we first traverse the cotree T of a given (whole) trivially perfect graph G from the root to leaves, and assign the sub-instance (G_i, k, f_0^i, f_r^i) to each node $i \in V_T$; we then solve the sub-instances from leaves to the root of T by combining their children's answers.

This completes our proof of Theorem 4. □

5. Conclusions

In this paper, we have studied coloring reconfiguration from the viewpoint of graph classes. We first proved that k-coloring reconfiguration is PSPACE-complete for chordal graphs; this answers the open question posed by Bonsma and Paulusma [5]. We then demonstrated that coloring reconfiguration is solvable in polynomial time for several graph classes, even when k is a part of input; such graph classes include split graphs and trivially perfect graphs.

Acknowledgments This work is partially supported by JST CREST Grant Number JPMJCR1402, and by JSPS KAKENHI Grant Numbers JP16J02175, JP16K00003, and JP16K00004, Japan.

References

[1] Bonamy, M., Johnson, M., Lignos, I., Patel, V., Paulusma, D.: Reconfiguration graphs for vertex colourings of chordal and chordal bipartite graphs. *J. Combinatorial Optimization*, Vol. 27, pp. 132–143 (2014).
[2] Bonamy, M., Bousquet, N.: Recoloring bounded treewidth graphs. *Electronic Notes in Discrete Mathematics*, Vol. 44, pp. 257–262 (2013).
[3] Bonsma, P., Cereceda, L.: Finding paths between graph colourings: PSPACE-completeness and superpolynomial distances. *Theoretical Computer Science*, Vol. 410, pp. 5215–5226 (2009).
[4] Bonsma, P., Mouawad, A.E., Nishimura, N., Raman, V.: The complexity of bounded length graph recoloring and CSP reconfiguration. *Proc. of IPEC 2014*, LNCS 8894, pp. 110–121 (2014).
[5] Bonsma, P., Paulusma, D.: Using contracted solution graphs for solving reconfiguration problems. *Proc. of MFCS 2016*, LIPIcs 58, pp. 20:1–20:15 (2016).
[6] Brewster, R.C., McGuinness, S., Moore, B., Noel, J.A.: A dichotomy theorem for circular colouring reconfiguration. *Theoretical Computer Science*, Vol. 639, pp. 1–13 (2016).
[7] Cereceda, L.: Mixing Graph Colourings. Ph.D. Thesis, London School of Economics and Political Science (2007).
[8] Cereceda, L., van den Heuvel, J., Johnson, M.: Finding paths between 3-colorings. *J. Graph Theory*, Vol. 67, pp. 69–82 (2011).
[9] Demaine, E.D., Demaine, M.L., Fox-Epstein, E., Hoang, D.A., Ito, T., Ono, H., Otachi, Y., Uehara, R., Yamada, T.: Linear-time algorithm for sliding tokens on trees. *Theoretical Computer Science*, Vol. 600, pp. 132–142 (2015).
[10] Hammer, P.L., Simeone, B.: The splittance of a graph. *Combinatorica*, Vol. 1, pp. 275–284 (1981).
[11] Hatanaka, T., Ito, T., Zhou, X.: The list coloring reconfiguration problem for bounded pathwidth graphs, *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences* E98-A, pp. 1168–1178 (2015).
[12] van den Heuvel, J.: The complexity of change. *Surveys in Combinatorics 2013*, London Mathematical Society Lecture Notes Series 409 (2013).
[13] Ito, T., Demaine, E.D., Harvey, N.J.A., Papadimitriou, C.H., Sideri, M., Uehara, R., Uno, Y.: On the complexity of reconfiguration problems. *Theoretical Computer Science*, Vol. 412, pp. 1054–1065 (2011).
[14] Johnson, M., Kratsch, D., Kratsch, S., Patel, V., Paulusma, D.: Finding shortest paths between graph colourings. *Algorithmica*, Vol. 75, pp. 295–321 (2016).
[15] McConnell, R.M., Spinrad, J.P.: Linear-time modular decomposition of directed graphs. *Discrete Applied Mathematics*, Vol. 145, pp. 198–209 (2005).
[16] Wrochna, M.: Reconfiguration in bounded bandwidth and treedepth. `arXiv:1405.0847` (2014).
[17] Wrochna, M.: Homomorphism reconfiguration via homotopy. *Proc. of STACS 2015*, LIPIcs 30, pp. 730–742 (2015).