Reconfiguration of List Edge-Colorings in a Graph

Takehiro Ito[1]*, Marcin Kamiński[2]**, and Erik D. Demaine[3]

[1] Graduate School of Information Sciences, Tohoku University,
Aoba-yama 6-6-05, Sendai, 980-8579, Japan.
takehiro@ecei.tohoku.ac.jp

[2] Department of Computer Science, Université Libre de Bruxelles,
CP 212, Bvd. du Triomphe, 1050 Bruxelles, Belgium.
marcin.kaminski@ulb.ac.be

[3] MIT Computer Science and Artificial Intelligence Laboratory,
32 Vassar St., Cambridge, MA 02139, USA.
edemaine@mit.edu

Abstract. We study the problem of reconfiguring one list edge-coloring of a graph into another list edge-coloring by changing only one edge color assignment at a time, while at all times maintaining a list edge-coloring, given a list of allowed colors for each edge. First we show that this problem is PSPACE-complete, even for planar graphs of maximum degree 3 and just six colors. We then consider the problem restricted to trees. We show that any list edge-coloring can be transformed into any other under the sufficient condition that the number of allowed colors for each edge is strictly larger than the degrees of both its endpoints. This sufficient condition is best possible in some sense. Our proof yields a polynomial-time algorithm that finds a transformation between two given list edge-colorings of a tree with n vertices using $O(n^2)$ recolor steps. This worst-case bound is tight: we give an infinite family of instances on paths that satisfy our sufficient condition and whose reconfiguration requires $\Omega(n^2)$ recolor steps.

1 Introduction

Reconfiguration problems arise when we wish to find a step-by-step transformation between two feasible solutions of a problem such that all intermediate results are also feasible. Ito *et al.* [8] proposed a framework of reconfiguration problems, and gave complexity and approximability results for reconfiguration problems derived from several well-known problems, such as INDEPENDENT SET, CLIQUE, MATCHING, etc. In this paper, we study a reconfiguration problem for list edge-colorings of a graph.

An (ordinary) *edge-coloring* of a graph G is an assignment of colors from a color set C to each edge of G so that every two adjacent edges receive different colors. In *list edge-coloring*, each edge e of G has a set $L(e)$ of colors, called the *list* of e. Then, an edge-coloring f of G is called an *L-edge-coloring* of G if $f(e) \in L(e)$ for each edge e, where $f(e)$ denotes the color assigned to e by f. Figure 1 illustrates three L-edge-colorings of the same graph with the same list L; the color assigned to each edge is surrounded by a box in the list. Clearly, an edge-coloring is merely an L-edge-coloring for which $L(e) = C$ for every edge e of G, and hence list edge-coloring is a generalization of edge-coloring.

Suppose now that we are given *two* L-edge-colorings 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

* This work is partially supported by Grant-in-Aid for Scientific Research: 22700001 (T. Ito).

** Chargé de Recherches du F.R.S. – FNRS.

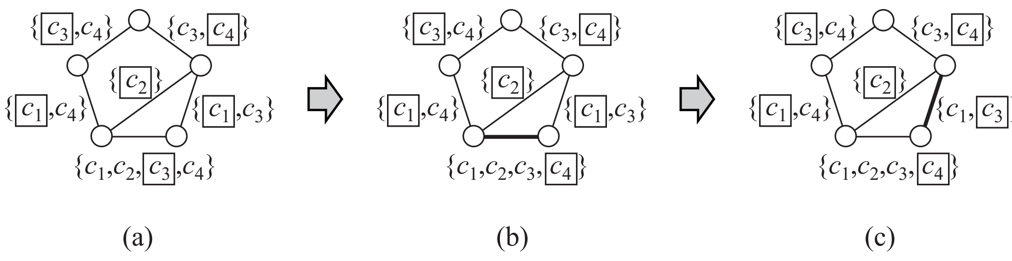


Fig. 1. A sequence of L-edge-colorings of a graph.

other via L-edge-colorings of G such that each differs from the previous one in only one edge color assignment. We call this problem the LIST EDGE-COLORING RECONFIGURATION problem. For the particular instance of Fig. 1, the answer is "yes," as illustrated in Fig. 1, where the edge whose color assignment was changed from the previous one is depicted by a thick line. One can imagine a variety of practical scenarios where an edge-coloring (e.g., representing a feasible schedule) needs to be changed (to use a newly found better solution or to satisfy new side constraints) by individual color changes (preventing the need for any coordination) while maintaining feasibility (so that nothing goes wrong during the transformation). Reconfiguration problems are also interesting in general because they provide a new perspective and deeper understanding of the solution space and of heuristics that navigate that space.

Reconfiguration problems have been studied extensively in recent literature [1, 3, 4, 6–8, 11], in particular for (ordinary) vertex-colorings. For a positive integer k, a *k-vertex-coloring* of a graph is an assignment of colors from $\{c_1, c_2, \ldots, c_k\}$ to each vertex so that every two adjacent vertices receive different colors. Then, the k-VERTEX-COLORING RECONFIGURATION problem is defined analogously. Bonsma and Cereceda [1] proved that k-VERTEX-COLORING RECONFIGURATION is PSPACE-complete for $k \geq 4$; they also proved that the reconfiguration problem for *list vertex-colorings* is PSPACE-complete even for planar graphs of maximum degree 4 and four colors. On the other hand, Cereceda *et al.* [4] proved that k-VERTEX-COLORING RECONFIGURATION is solvable in polynomial time for $1 \leq k \leq 3$. Edge-coloring in a graph G can be reduced to vertex-coloring in the "line graph" of G. However, by this reduction, we can solve only a few instances of LIST EDGE-COLORING RECONFIGURATION; all edges e of G must have the same list $L(e) = C$ of size $|C| \leq 3$ although any edge-coloring of G requires at least $\Delta(G)$ colors, where $\Delta(G)$ is the maximum degree of G. Furthermore, the reduction does not work the other way, so we do not obtain any complexity results.

In this paper, we give three results for LIST EDGE-COLORING RECONFIGURATION. The first is to show that the problem is PSPACE-complete, even for planar graphs of maximum degree 3 and six colors. The second is to give a sufficient condition for which there exists a transformation between any two L-edge-colorings of a tree. Specifically, for a tree T, we prove that any two L-edge-colorings of T can be transformed into each other if $|L(e)| \geq \max\{d(v), d(w)\} + 1$ for each edge $e = vw$ of T, where $d(v)$ and $d(w)$ are the degrees of the endpoints v and w of e, respectively. Our proof for the sufficient condition yields a polynomial-time algorithm that finds a transformation between two given L-edge-colorings of T via $O(n^2)$ intermediate L-edge-colorings, where n is the number of vertices in T. On the other hand, as the third result, we show that our worst-case bound on the number of intermediate L-edge-colorings is tight: we give an infinite family of instances on paths that satisfy our sufficient condition and whose transformation requires $\Omega(n^2)$ intermediate L-edge-colorings. An early version of the paper has been presented in [9].

Our sufficient condition for trees was motivated by several results on the well-known "list coloring conjecture" [10]: it is conjectured that any graph G has an L-edge-coloring if $|L(e)| \geq \chi'(G)$ for each edge e, where $\chi'(G)$ is the chromatic index of G, that is, the minimum number of colors required for an ordinary edge-coloring of G. This conjecture has not been proved yet, but some results are known for restricted classes of graphs [2, 5, 10]. In particular, Borodin *et al.* [2] proved that any bipartite graph G has an L-edge-coloring if $|L(e)| \geq \max\{d(v), d(w)\}$ for each edge $e = vw$. Because any tree is a bipartite graph, one might think that it would be straightforward to extend their result [2] to our sufficient condition. However, this is not the case, because the focus of reconfiguration problems is not the *existence* (as in the previous work) but the *reachability* between two feasible solutions; there must exist a transformation between any two L-edge-colorings if our sufficient condition holds.

Finally, we remark that our sufficient condition is best possible in some sense. Consider a star $K_{1,n-1}$ of $n-1$ edges in which each edge e has the same list $L(e) = C$ of size $|C| = n-1$. Then, $|L(e)| = \max\{d(v), d(w)\}$ for all edges $e = vw$, and it is easy to see that there is no transformation between any two L-edge-colorings of the star.

2 PSPACE-completeness

Before proving PSPACE-completeness, we introduce some terms and define the problem more formally. In Section 1, we have defined an L-edge-coloring of a graph $G = (V, E)$ with a list L. We say that two L-edge-colorings f and f' of G are *adjacent* if

$$\big|\{e \in E : f(e) \neq f'(e)\}\big| = 1,$$

that is, f' can be obtained from f by changing the color assignment of a single edge e; the edge e is said to be *recolored* between f and f'. A *reconfiguration sequence* between two L-edge-colorings f_0 and f_t of G is a sequence of L-edge-colorings $f_0, f_1, \ldots, f_t$ of G such that f_{i-1} and f_i are adjacent for $i = 1, 2, \ldots, t$. We also say that two L-edge-colorings f and f' are *connected* if there exists a reconfiguration sequence between f and f'. Clearly, any two adjacent L-edge-colorings are connected. Then, the LIST EDGE-COLORING RECONFIGURATION problem is to determine whether two given L-edge-colorings of a graph G are connected. Note that this problem is a decision problem, and hence does not ask an actual reconfiguration sequence. For a reconfiguration sequence between two L-edge-colorings, its *length* is defined as the number of L-edge-colorings contained in the reconfiguration sequence, and hence the length of the reconfiguration sequence in Fig. 1 is 3.

The main result of this section is the following theorem.

Theorem 1. LIST EDGE-COLORING RECONFIGURATION *is* PSPACE-*complete for planar graphs of maximum degree* 3 *whose lists are chosen from six colors.*

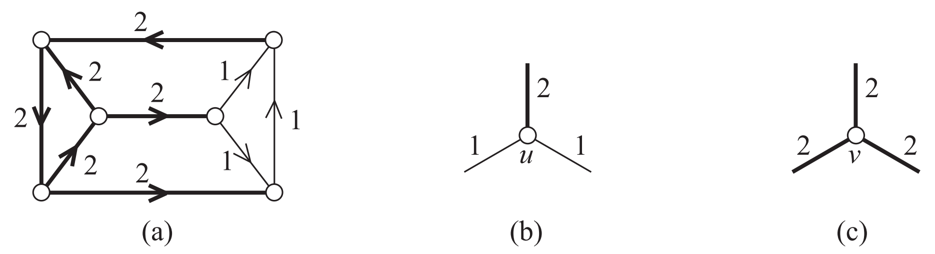


Fig. 2. (a) A configuration of an NCL machine, (b) NCL And vertex u, and (c) NCL Or vertex v.

In order to prove Theorem 1, we give a polynomial-time reduction from Nondeterministic Constraint Logic (NCL) [7] to our problem. An NCL "machine" is specified by a *constraint graph*: an undirected graph together with an assignment of weights from $\{1, 2\}$ to each edge of the graph. A *configuration* of this machine is an orientation (direction) of the edges such that the sum of weights of incoming edges at each vertex is at least 2. Figure 2(a) illustrates a configuration of an NCL machine, where each weight-2 edge is depicted by a thick line and each weight-1 edge by a thin line. A *move* from one configuration is simply the reversal of a single edge direction which results in another (feasible) configuration. Given an NCL machine and its two configurations, it is PSPACE-complete to determine whether there exists a sequence of moves which transforms one configuration into the other [7].

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 3 is called an *NCL* And *vertex* if its three incident edges have weights 1, 1 and 2. (See Fig. 2(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 3 is called an *NCL* Or *vertex* if its three incident edges have weights 2, 2 and 2. (See Fig. 2(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 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. The NCL machine in Fig. 2(a) is an And/Or constraint graph. From now on, we call an And/Or constraint graph simply an *NCL machine*.

Proof of Theorem 1.

It is easy to see that list edge-coloring reconfiguration can be solved in (most conveniently, nondeterministic [12]) polynomial space. Therefore, in the remainder of this section, we show that the problem is PSPACE-hard by giving a polynomial-time reduction from NCL. This reduction involves constructing two types of gadgets which correspond to NCL And and Or vertices. We call an edge of an NCL machine an *NCL edge*, and say simply an *edge* of a graph for list edge-coloring reconfiguration.

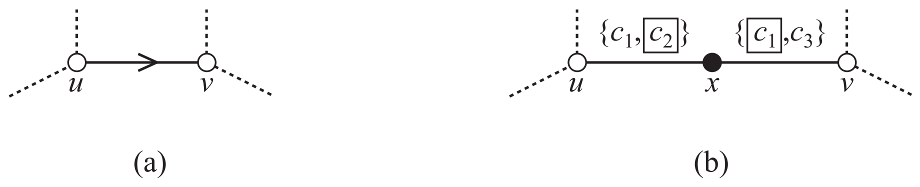


Fig. 3. (a) An NCL edge uv and (b) its corresponding edges ux and xv of a graph with lists $L(ux) = \{c_1, c_2\}$ and $L(xv) = \{c_1, c_3\}$.

Assume in our reduction that the color c_1 corresponds to "directed inward," and that both colors c_2 and c_3 correspond to "directed outward." Consider an NCL edge uv directed from u to v. (See Fig. 3(a).) Then, the NCL edge is directed outward for u, but is directed inward for v. Clearly, in list edge-coloring, each edge can receive only one color. Therefore, we need to split one NCL edge uv into two edges ux and xv of a graph with lists $L(ux) = \{c_1, c_2\}$ and $L(xv) = \{c_1, c_3\}$, as illustrated in Fig. 3(b). The new vertex x is sometimes called *midpoint* of an NCL edge uv. Note that every NCL half-edge joins a non-midpoint and a midpoint; our viewpoint is always on the non-midpoint when we say "directed inward" or "directed outward." It is easy to see that one of ux and xv can be colored with c_1 if and only if the other edge is colored with either c_2 or c_3. This property represents that an NCL half-edge can be directed inward if and only if the other half is directed outward. Note that, if neither ux nor xv is colored with c_1, then the corresponding NCL edge uv can be directed arbitrarily.

Figure 4 illustrates three kinds of "AND gadgets," each of which corresponds to an NCL AND vertex u; two edges $u_x x$ and $u_y y$ correspond to the two weight-1 NCL half-edges, and the edge $u_z z$ corresponds to the weight-2 NCL half-edge; thus, the three vertices x, y and z correspond to midpoints adjacent with u in an NCL machine. Since NCL AND and OR vertices are connected together into an arbitrary NCL machine, there should be eight kinds of AND gadgets according to the choice of lists $\{c_1, c_2\}$ and $\{c_1, c_3\}$ for three edges $u_x x$, $u_y y$ and $u_z z$. However, since the two weight-1 NCL edges are symmetric, it suffices to consider these three kinds: all the three edges have the same list, as in Fig. 4(a); $u_z z$ has a different list from the other two edges, as in Fig. 4(b); and one of $u_x x$ and $u_y y$ has a different list from the other two edges, as in Fig. 4(c).

We denote by $\mathcal{F}(A; c_x, c_y, c_z)$ the set of all L-edge-colorings f of an AND gadget A such that $f(u_x x) = c_x$, $f(u_y y) = c_y$ and $f(u_z z) = c_z$. Then, all the L-edge-colorings in $\mathcal{F}(A; c_x, c_y, c_z)$ correspond to the same direction of the three NCL half-edges in the NCL AND vertex. We now check that the three kinds of AND gadgets satisfy the same constraints as an NCL AND vertex; we check this property by enumerating all possible L-edge-colorings of the AND gadgets. For example, in the AND gadget A of Fig. 4(a), $u_z z$ can be colored with c_2 (directed outward) if and only if both $u_x x$ and $u_y y$ are colored with c_1 (directed inward); in other words, $|\mathcal{F}(A; c_x, c_y, c_2)| \geq 1$ if and only if $c_x = c_y = c_1$. In addition, every AND gadget A satisfies the following two properties:

(i) For each triple (c_x, c_y, c_z) such that $|\mathcal{F}(A; c_x, c_y, c_z)| \geq 2$, any two L-edge-colorings f and f' in $\mathcal{F}(A; c_x, c_y, c_z)$ are "internally connected," that is, there exists a reconfiguration sequence between f and f' via L-edge-colorings only in $\mathcal{F}(A; c_x, c_y, c_z)$; and

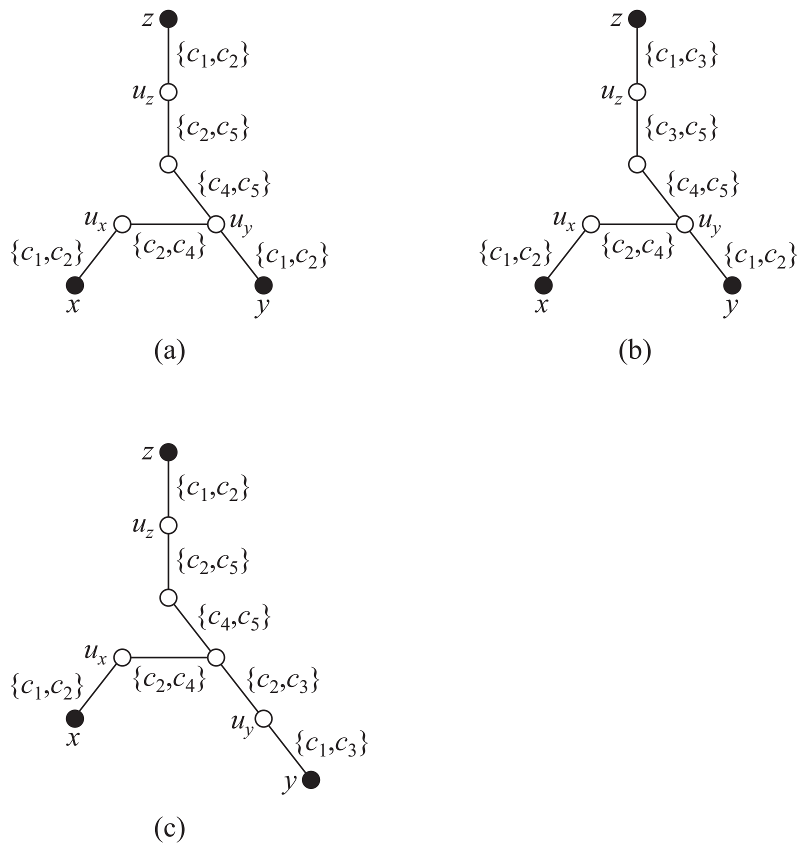


Fig. 4. Three kinds of And gadgets corresponding to an NCL And vertex u. The vertices x, y and z correspond to midpoints adjacent with u in an NCL machine.

(ii) For every two triples (c_x, c_y, c_z) and (c'_x, c'_y, c'_z) which differ in a single coordinate, if $|\mathcal{F}(A; c_x, c_y, c_z)| \geq 1$ and $|\mathcal{F}(A; c'_x, c'_y, c'_z)| \geq 1$, then there exist two L-edge-colorings $f \in \mathcal{F}(A; c_x, c_y, c_z)$ and $f' \in \mathcal{F}(A; c'_x, c'_y, c'_z)$ such that f and f' are adjacent.

Then, it is easy to see that the reversal of a single NCL half-edge direction in an NCL And vertex can be simulated by a reconfiguration sequence between two L-edge-colorings each of which is chosen arbitrarily from the set $\mathcal{F}(A; c_x, c_y, c_z)$, where the triple (c_x, c_y, c_z) corresponds to the direction of the three NCL half-edges.

Figure 5 illustrates two kinds of "Or gadgets," each of which corresponds to an NCL Or vertex v; three edges $v_x x$, $v_y y$ and $v_z z$ correspond to the three weight-2 NCL half-edges; thus, the three vertices x, y and z correspond to midpoints adjacent with v in an NCL machine. Since an NCL Or vertex is entirely symmetric, it suffices to consider these two kinds: all the three edges $v_x x$, $v_y y$ and $v_z z$ have the same list, as in Fig. 5(a); and one edge has a different list from the other two edges, as in Fig. 5(b). Then, similarly as And gadgets, it is easy to see that both kinds of Or gadgets satisfy the same constraints as an NCL Or vertex, and that the reversal of a single NCL half-edge direction in an NCL Or vertex can be simulated by a reconfiguration sequence between two corresponding L-edge-colorings.

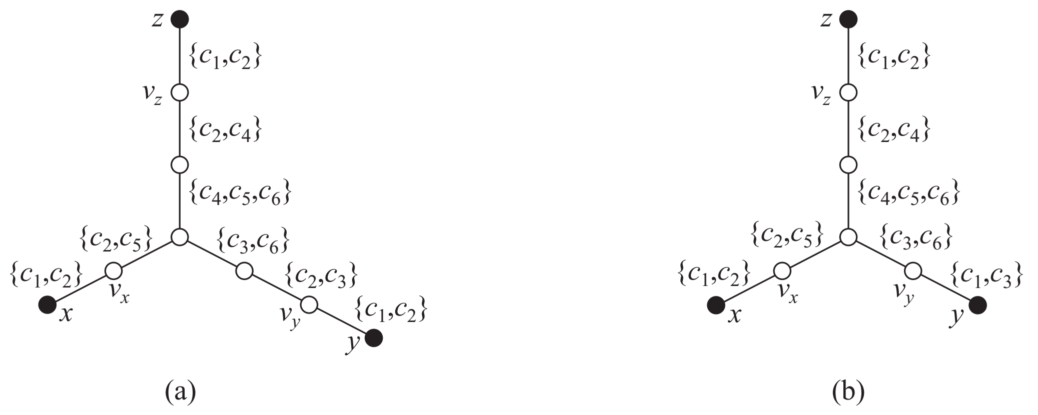


Fig. 5. Two kinds of Or gadgets corresponding to an NCL Or vertex v. The vertices x, y and z correspond to midpoints adjacent with v in an NCL machine.

We now construct the corresponding instance of LIST EDGE-COLORING RECONFIGURATION. Given NCL machine, we construct a graph G with a list L by replacing NCL And and Or vertices (together with their NCL half-edges) with And and Or gadgets, respectively. Then, every configuration of the NCL machine can be mapped to at least one (in general, to exponentially many) L-edge-colorings of G. We can choose an arbitrary one for each of two given configurations of the NCL machine, because each And gadget satisfies Property (i) above and each Or gadget does the counterpart. Since NCL remains PSPACE-complete even if an NCL machine is planar [7], G is a planar graph of maximum degree 3. Furthermore, each list $L(e)$ is a subset of $\{c_1, c_2, \ldots, c_6\}$.

It is now easy to see that there is a sequence of moves which transforms one configuration into the other if and only if there is a reconfiguration sequence between the two L-edge-colorings of G.

This completes the proof of Theorem 1. □

3 Trees

Since LIST EDGE-COLORING RECONFIGURATION is PSPACE-complete, it is rather unlikely that the problem can be solved in polynomial time for general graphs. However, in Section 3.1, we give a sufficient condition for which any two L-edge-colorings of a tree T are connected; our sufficient condition can be checked in polynomial time. Moreover, our proof yields a polynomial-time algorithm that finds a reconfiguration sequence of length $O(n^2)$ between two given L-edge-colorings, where n is the number of vertices in T. In Section 3.2, we give an infinite family of instances on paths that satisfy our sufficient condition and whose reconfiguration sequence requires length $\Omega(n^2)$.

3.1 Sufficient condition

The main result of this subsection is the following theorem, whose sufficient condition is in some sense best possible as we mentioned in Section 1.

Theorem 2. *For a tree T with n vertices, any two L-edge-colorings f and f' of T are connected if $|L(e)| \geq \max\{d(v), d(w)\} + 1$ for each edge $e = vw$ of T. Moreover, there is a reconfiguration sequence of length $O(n^2)$ between f and f'.*

Since $\Delta(T) \geq \max\{d(v), d(w)\}$ for all edges vw of a tree T, Theorem 2 immediately implies the following sufficient condition for which any two (ordinary) edge-colorings of T are connected. Note that, for a positive integer k, a *k-edge-coloring* of a tree T is an L-edge-coloring of T for which all edges e have the same list $L(e) = \{c_1, c_2, \dots, c_k\}$.

Corollary 1. *For a tree T with n vertices, any two k-edge-colorings f and f' of T are connected if $k \geq \Delta(T) + 1$. Moreover, there is a reconfiguration sequence of length $O(n^2)$ between f and f'.*

It is obvious that the sufficient condition of Corollary 1 is also best possible in some sense; consider the star $K_{1,n-1}$ in Section 1.

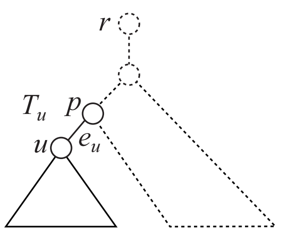
In the remainder of this subsection, as a proof of Theorem 2, we give a polynomial-time algorithm that finds a reconfiguration sequence of length $O(n^2)$ between two given L-edge-colorings f_0 and f_t of a tree T if our sufficient condition holds.

We first give an outline of our algorithm. By the breadth-first search starting from an arbitrary vertex r of degree 1, we order all edges $e_1, e_2, \dots, e_{n-1}$ of a tree T. At the ith step, $1 \leq i \leq n-1$, the algorithm recolors e_i from the current color to its target color $f_t(e_i)$ without recoloring any of the edges $e_1, e_2, \dots, e_{i-1}$. Therefore, e_i is never recolored after the ith step, while e_j with $j > i$ may be recolored even if e_j is colored with $f_t(e_j)$. We will show later that every edge of T can be recolored in such a way, and hence we eventually obtain the target L-edge-coloring f_t after $(n-1)$ steps of the algorithm. Our algorithm recolors each edge e_j with $j \geq i$ at most once in the ith step, and hence e_i receives its target color $f_t(e_i)$ by recoloring at most $(n-i)$ edges. We thus obtain a reconfiguration sequence of total length $\sum_{i=1}^{n-1}(n-i) = O(n^2)$.

Definitions.

For a tree T, we denote by $V(T)$ and $E(T)$ the vertex set and edge set of T, respectively. Suppose that we are given a tree T with a list L such that

$$|L(e)| \geq \max\{d(v), d(w)\} + 1 \tag{1}$$



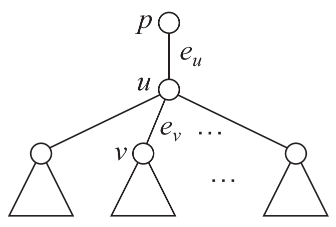


Fig. 6. (a) Subtree T_u in the whole tree T and (b) inside of T_u.

for each edge $e = vw$ in $E(T)$. We choose an arbitrary vertex r of degree 1 as the root of T, and regard T as a rooted tree. For a vertex u in $V(T) \setminus \{r\}$, let p be the parent of u in T. We denote by T_u the subtree of T which is rooted at p and is induced by p, u and all descendants of u in T. (See Fig. 6(a).) It should be noted that T_u includes the edge $e_u = pu$, but does not include the other edges incident to p. Therefore, T_u consists of a single edge if u is a leaf of T. We always denote by e_u the edge which joins u and its parent p.

Let f be an L-edge-coloring of a tree T. For a vertex v of T, we say that a color c is *available on v in f* if $c \notin \{f(vx) : vx \in E(T)\}$, that is, c is not assigned to any of the edges incident to v. For an edge $e = vw$ of T and its endpoint v, we define a subset $C_{\text{av}}(f, e, v)$ of $L(e)$, as follows:

$$C_{\text{av}}(f, e, v) = L(e) \setminus \{f(vx) : vx \in E(T)\}. \tag{2}$$

That is, $C_{\text{av}}(f, e, v)$ is the set of all colors in $L(e)$ that are available on v for e. Therefore, $C_{\text{av}}(f, e, v) \cap C_{\text{av}}(f, e, w)$ is the set of all colors in $L(e)$ that are available for $e = vw$ when we wish to recolor e from $f(e)$.

We now have the following lemma.

Lemma 1. *Let $e_u = pu$ be an arbitrary edge in T such that p is the parent of u. Let c be any color in $C_{\text{av}}(f, e_u, p)$. Then, there exists an L-edge-coloring f' of T such that $f'(e_u) = c$ and f' can be obtained by recoloring each edge in T_u at most once.*

Proof. We prove the lemma by induction on the number of edges in T_u. By Eq. (2) c is not assigned to any of the edges incident to p in the whole tree T. Therefore, if T_u contains exactly one edge $e_u = pu$ and hence u is a leaf of T, then e_u can be recolored to any color in $C_{\text{av}}(f, e_u, p)$. Thus, the lemma clearly holds for this case.

We may assume that the color $c \in C_{\text{av}}(f, e_u, p)$ is assigned to some edge $e_v = uv$, as illustrated in Fig. 6(b); because, otherwise, the lemma clearly holds. By Eqs. (1) and (2) we have

$$\begin{aligned} |C_{\text{av}}(f, e_v, u)| &\geq \big|L(e_v)\big| - \big|\{f(ux) : ux \in E(T)\}\big| \\ &\geq \max\{d(u), d(v)\} + 1 - d(u) \\ &\geq 1. \end{aligned}$$

Therefore, $C_{\text{av}}(f, e_v, u)$ contains at least one color c', and hence we can apply the induction hypothesis for the edge $e_v = uv$ and the color c'. Then, we have an L-edge-coloring f'' of T such that $f''(e_v) = c'$ without recoloring any edge in $T \setminus T_v$. Since c was assigned to $e_v = uv$ in f, we have $c \in C_{\text{av}}(f'', e_u, u)$. Clearly, $C_{\text{av}}(f'', e_u, p) = C_{\text{av}}(f, e_u, p)$, and hence $c \in C_{\text{av}}(f'', e_u, p) \cap C_{\text{av}}(f'', e_u, u)$. Therefore, we can now recolor $e_u = pu$ from $f(e_u)$ to c. Note that each edge in T_u is recolored at most once, as required. □

Algorithm.

We are now ready to describe our algorithm. Assume that all edges $e_1, e_2, \ldots, e_{n-1}$ of a tree T are ordered by the breadth-first search starting from the root r of T. At the ith step, $1 \leq i \leq n-1$, the algorithm recolors e_i to its target color $f_t(e_i)$. Consider the ith step of the algorithm, and let f be the current L-edge-coloring of T obtained after

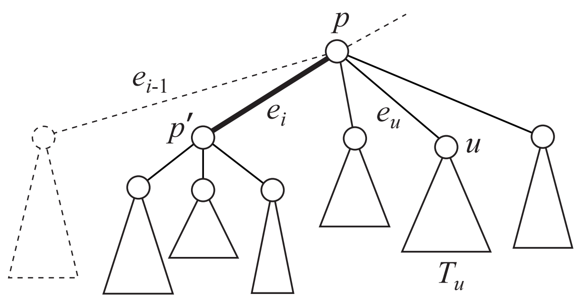


Fig. 7. The ith step of the algorithm.

$(i-1)$ steps of the algorithm; let $f = f_0$ for the first step $i = 1$. Then, we wish to recolor $e_i = pp'$ from the current color $f(e_i)$ to the target color $f_t(e_i)$. (See also Fig. 7.) There are the following two cases to consider.

Case (a): $f_t(e_i) \in C_{\rm av}(f, e_i, p) \cap C_{\rm av}(f, e_i, p')$

In this case, $f_t(e_i)$ is available for e_i, that is, there is no edge which is adjacent with e_i and is colored with $f_t(e_i)$. Therefore, we can simply recolor e_i from $f(e_i)$ to its target color $f_t(e_i)$.

Case (b): $f_t(e_i) \notin C_{\rm av}(f, e_i, p) \cap C_{\rm av}(f, e_i, p')$

In this case, there are at most two edges pu and $p'u'$ which are colored with $f_t(e_i)$ and are sharing the endpoints p and p' with e_i, respectively. Let p be the parent of p', as illustrated in Fig. 7.

We first consider the case $f_t(e_i) \notin C_{\rm av}(f, e_i, p)$. Then, the color $f_t(e_i)$ is assigned to some edge $e_u = pu$. But, in the target L-edge-coloring f_t, the color $f_t(e_i)$ is *not* assigned to any edge incident to p other than $e_i = pp'$. Since the edges $e_1, e_2, \ldots, e_{i-1}$ have already received their target colors, e_u must appear after e_i in the breadth-first search order. (See also Fig. 7.) By Eqs. (1) and (2) we have

$$|C_{\rm av}(f, e_u, p)| \geq \max\{d(p), d(u)\} + 1 - d(p) \geq 1.$$

Therefore, $C_{\rm av}(f, e_u, p)$ contains a color c, and hence we can apply Lemma 1 to the edge $e_u = pu$ and the color c. We can thus obtain an L-edge-coloring f' of T such that $f'(e_u) = c$ without recoloring any of the edges $e_1, e_2, \ldots, e_{i-1}$. Note that $f_t(e_i) \in C_{\rm av}(f', e_i, p)$ since $f_t(e_i)$ was assigned to $e_j = pu$ in f.

We then consider the case $f_t(e_i) \notin C_{\rm av}(f, e_i, p')$. Let f' be the L-edge-coloring of T obtained above; let $f' = f$ if $f_t(e_i) \in C_{\rm av}(f, e_i, p)$. Since $f_t(e_i) \in C_{\rm av}(f', e_i, p)$, we apply Lemma 1 to the edge e_i and the color $f_t(e_i)$. Then, we can obtain an L-edge-coloring f'' of T such that $f''(e_i) = f_t(e_i)$, as required.

In this way, we can always recolor e_i to $f_t(e_i)$ at the ith step, $1 \leq i \leq n-1$, without recoloring any of the edges $e_1, e_2, \ldots, e_{i-1}$. Therefore, our algorithm terminates with the target L-edge-coloring f_t. Since the algorithm recolors an edge at most once in each step, at most $(n-i)$ edges are recolored in the ith step. Therefore, the total length of the reconfiguration sequence is $\sum_{i=1}^{n-1}(n-i) = O(n^2)$.

This completes the proof of Theorem 2. □

3.2 Length of reconfiguration sequence

We showed in Section 3.1 that any two L-edge-colorings of a tree T are connected via a reconfiguration sequence of length $O(n^2)$ if our sufficient condition holds. In this subsection, we show that this worst-case bound on the length is tight: we give an infinite family of instances on paths that satisfy our sufficient condition and whose reconfiguration sequence requires length $\Omega(n^2)$.

Consider a path $P = v_0 v_1 \dots v_{3m+1}$ of $3m+1$ edges in which every edge e has the same list $L(e) = \{c_1, c_2, c_3\}$. Clearly, the list L satisfies Eq. (1), and hence any two L-edge-colorings of P are connected. We construct two L-edge-colorings f_0 and f_t of P, as follows: (see also Fig. 8):

$$f_0(v_i v_{i+1}) = \begin{cases} c_3 & \text{if } i \equiv 0 \mod 3; \\ c_2 & \text{if } i \equiv 1 \mod 3; \\ c_1 & \text{if } i \equiv 2 \mod 3 \end{cases} \tag{3}$$

for each edge $v_i v_{i+1}$, $0 \le i \le 3m$, and

$$f_t(v_i v_{i+1}) = \begin{cases} c_3 & \text{if } i \equiv 0 \mod 3; \\ c_1 & \text{if } i \equiv 1 \mod 3; \\ c_2 & \text{if } i \equiv 2 \mod 3 \end{cases} \tag{4}$$

for each edge $v_i v_{i+1}$, $0 \le i \le 3m$. Then, we have the following theorem.

Theorem 3. *For the path P and its two L-edge-colorings f_0 and f_t defined above, every reconfiguration sequence between f_0 and f_t requires length $\Omega(n^2)$, where n is the number of vertices in P.*

Proof. For an L-edge-coloring f of P and an internal vertex v_i, $1 \le i \le 3m$, we define the *sign $s(f, v_i)$ of v_i on f*, as follows:

$$s(f, v_i) = \begin{cases} +1 & \text{if } \big(f(v_{i-1}v_i), f(v_i v_{i+1})\big) \in \{(c_1, c_2), (c_2, c_3), (c_3, c_1)\}; \\ -1 & \text{if } \big(f(v_{i-1}v_i), f(v_i v_{i+1})\big) \in \{(c_3, c_2), (c_2, c_1), (c_1, c_3)\}. \end{cases}$$

Therefore, for all internal vertices v_i, $1 \le i \le 3m$, we have $s(f_0, v_i) = -1$ and $s(f_t, v_i) = +1$. We represent an L-edge-coloring f of P by a *sign sequence* $S(f) = \big(s(f, v_1), s(f, v_2),$

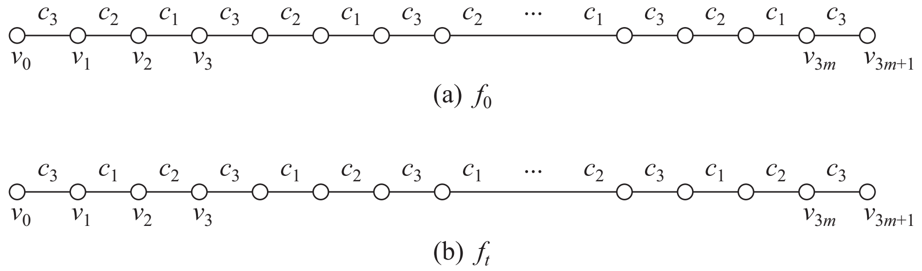


Fig. 8. Two L-edge-colorings f_0 and f_t of the path P.

$\ldots, s(f, v_{3m})\big)$ which consists of signs of the vertices v_i, $1 \le i \le 3m$. Then, f_0 is represented by

$$S(f_0) = (-1, -1, \ldots, -1), \tag{5}$$

and f_t is represented by

$$S(f_t) = (+1, +1, \ldots, +1). \tag{6}$$

Note that there are more than one L-edge-colorings of P which correspond to the same sign sequence. However, as a necessary condition, any reconfiguration sequence between f_0 and f_t is required to transform $S(f_0)$ into $S(f_t)$. For an L-edge-coloring f of P, we denote by $n^+(f)$ and $n^-(f)$ the numbers of "+1"s and "−1"s in the sign sequence $S(f)$, respectively. Clearly $n^+(f) + n^-(f) = 3m$, and hence it suffices to consider $n^+(f)$ and the placement of "+1"s in $S(f)$.

We now analyze a "recolor step" from the viewpoint of sign sequences. Consider any two adjacent L-edge-colorings f and f' of P, and let $v_i v_{i+1}$ be the edge which is recolored between f and f'. Note that $s(f, v) = s(f', v)$ if v is neither v_i nor v_{i+1}. This recolor step can be classified into the following two types (a) and (b).

(a) $v_i v_{i+1}$ is either $v_0 v_1$ or $v_{3m} v_{3m+1}$

Consider the case $v_i v_{i+1} = v_0 v_1$. (The other case is similar.) Since $f(v_1 v_2) = f'(v_1 v_2)$ and all the edges of P have the same list $\{c_1, c_2, c_3\}$, $f(v_0 v_1)$ and $f'(v_0 v_1)$ are the remaining two colors in $\{c_1, c_2, c_3\} \setminus \{f(v_1 v_2)\}$. Thus, it is easy to see that $s(f', v_1) = -s(f, v_1)$. Therefore, we have

$$n^+(f') = \begin{cases} n^+(f) + 1 & \text{if } s(f, v_1) = -1; \\ n^+(f) - 1 & \text{if } s(f, v_1) = +1. \end{cases}$$

(b) $v_i v_{i+1}$ is neither $v_0 v_1$ nor $v_{3m} v_{3m+1}$

In this case, there are two edges $v_{i-1} v_i$ and $v_{i+1} v_{i+2}$ which are adjacent with $v_i v_{i+1}$. Since all the edges of P have the same list $\{c_1, c_2, c_3\}$, both $v_{i-1} v_i$ and $v_{i+1} v_{i+2}$ must be colored with the same color; otherwise, we cannot recolor $v_i v_{i+1}$. Then, v_i and v_{i+1} have different signs, that is, $s(f, v_i) = -s(f, v_{i+1})$ and $s(f', v_i) = -s(f', v_{i+1})$. On the other hand, it is easy to see that the recolor step swaps the signs of v_i and v_{i+1}, that is, $s(f', v_i) = -s(f, v_i)$ and $s(f', v_{i+1}) = -s(f, v_{i+1})$. We thus have $n^+(f) = n^+(f')$ although $S(f) \ne S(f')$. From the viewpoint of the "+1"s' placement, only one "+1" was shifted to the right (from v_i to v_{i+1}) if $s(f, v_i) = +1$; otherwise, to the left.

By Eqs. (5) and (6) any reconfiguration sequence between f_0 and f_t is required to increase the number of "+1"s by recolor steps of Type (a) and to deliver "+1"s from either v_1 or v_{3m} to the vertices v_i, $1 \le i \le 3m$, by recolor steps of Type (b). Since one recolor step of Type (b) can shift one "+1" only to its adjacent vertex, the number of recolor steps of Type (b) required for delivering one "+1" from either v_1 or v_{3m} to a vertex v_i, $1 \le i \le 3m$, is at least

$$\min\{\mathrm{dist}(v_1, v_i), \mathrm{dist}(v_{3m}, v_i)\} = \min\{i - 1, 3m - i\},$$

where $\mathrm{dist}(v, v_i)$ is the number of edges between v and v_i. Note that these recolor steps of Type (b) are not necessarily executed consecutively, but they must be executed in any reconfiguration sequence for delivering one "+1" to v_i. The total number of recolor steps of both types to transform $S(f_0)$ into $S(f_t)$ is thus at least

$$\sum_{i=1}^{3m}\bigl(1+\min\{i-1, 3m-i\}\bigr) = \Omega(n^2).$$

Therefore, any reconfiguration sequence between f_0 and f_t is of length $\Omega(n^2)$.

This completes the proof of Theorem 3. □

4 Concluding Remarks

A reconfiguration sequence can be represented by a sequence of "recolor steps" (e, c), where a pair (e, c) denotes one recolor step which recolors an edge e of a tree T to some color $c \in L(e)$. Let $N = n + \sum_{e\in E(T)} |L(e)|$, where n is the number of vertices in T, then N denotes the input size. It is easy to see that our algorithm in Section 3.1 can be easily implemented so that it runs in time $O(nN)$: the algorithm stores and computes a sequence of recolor steps (e, c) together with only the current L-edge-coloring of T; then, each step of the algorithm can be executed in time $O(N)$, since we recolor each edge e of T at most once and hence the list $L(e)$ is checked at most once. Since there are $(n-1)$ steps, the algorithm takes time $O(nN)$ in total. Remember that $|L(e)| = 3$ for all edges e of the path P in Section 3.2. Then, our algorithm takes time $O(n^2)$ to find a reconfiguration sequence between the two L-edge-colorings f_0 and f_t defined by Eqs. (3) and (4), respectively. On the other hand, Theorem 3 suggests that it is difficult to improve the time-complexity of the algorithm if we wish to find an actual reconfiguration sequence explicitly.

One may expect that our sufficient condition for trees holds also for some larger classes of graphs, such as bipartite graphs, bounded treewidth graphs, etc. However, consider the following even-length cycle, which is bipartite and whose treewidth is 2. For an even integer m, let C be the cycle of $3m$ edges obtained by identifying the edge v_0v_1 with the edge $v_{3m}v_{3m+1}$ of P in Section 3.2, and let f_0 and f_t be L-edge-colorings of C defined similarly as in Eqs. (3) and (4), respectively. (See also Fig. 9.) Then, we cannot recolor any edge in

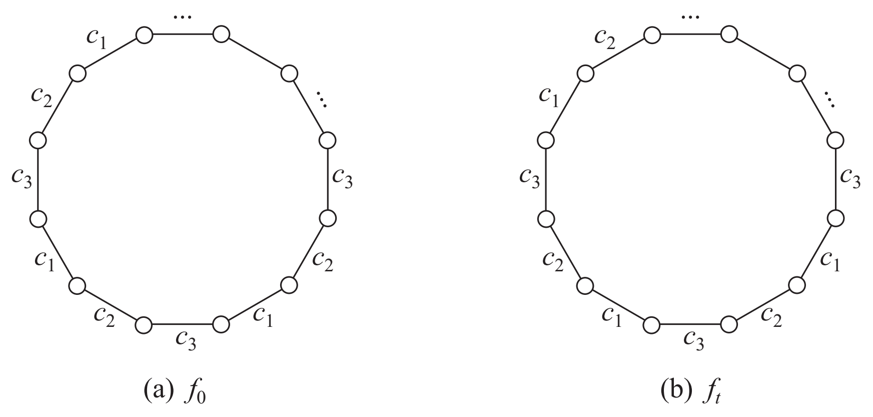


Fig. 9. Two L-edge-colorings f_0 and f_t of even-length cycle C.

the cycle, and hence f_0 and f_t are not connected even though $|L(e)| = \max\{d(v), d(w)\} + 1$ holds for each edge $e = vw$.

Acknowledgments

We thank Martin Demaine of MIT and the Algorithms Research Group of Université Libre de Bruxelles, especially Jean Cardinal and Raphaël Jungers, for fruitful discussions.

References

1. Bonsma, P., Cereceda, L.: Finding paths between graph colourings: PSPACE-completeness and superpolynomial distances. Theoretical Computer Science 410, 5215–5226 (2009)
2. Borodin, O.V., Kostochka, A.V., Woodall, D.R.: List edge and list total colourings of multigraphs. J. Combinatorial Theory, Series B 71, 184–204 (1997)
3. Călinescu, G., Dumitrescu, A., Pach, J.: Reconfigurations in graphs and grids. SIAM J. Discrete Mathematics 22, 124–138 (2008)
4. Cereceda, L., van den Heuvel, J., Johnson, M.: Finding paths between 3-colourings. In: Proc. of IWOCA 2008, pp. 182–196 (2008)
5. Fujino, T., Zhou, X., Nishizeki, T.: List edge-colorings of series-parallel graphs. IEICE Trans. Fundamentals E86-A, 1034–1045 (2003)
6. Gopalan, P., Kolaitis, P.G., Maneva, E.N., Papadimitriou, C.H.: The connectivity of Boolean satisfiability: computational and structural dichotomies. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4051, pp. 346–357. Springer, Heidelberg (2006)
7. Hearn, R.A., Demaine, E.D.: PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. Theoretical Computer Science 343, 72–96 (2005)
8. Ito, T., Demaine, E.D., Harvey, N.J.A., Papadimitriou, C.H., Sideri, M., Uehara, R., Uno, Y.: On the complexity of reconfiguration problems. In: Hong, S., Nagamochi, H., Fukunaga, T. (eds.) ISAAC 2008. LNCS, vol. 5369, pp. 28–39. Springer, Heidelberg (2008)
9. Ito, T., Kamiński, M., Demaine, E.D.: Reconfiguration of list edge-colorings in a graph. In: Dehne, F., Gavrilova, M., Sack, J.-R., Tóth, C.D. (eds.) WADS 2009. LNCS, vol. 5664, pp. 375–386. Springer, Heidelberg (2009)
10. Jensen, T.R., Toft, B.: Graph Coloring Problems. Wiley-Interscience, New York (1995)
11. Kamiński, M., Medvedev, P., Milanič, M.: Shortest paths between shortest paths and independent sets. In: Iliopoulos, C.S., Smyth, W.F. (eds.) IWOCA 2010. LNCS vol. 6460, pp. 56–67. Springer, Heidelberg (2011)
12. Savitch, W.J.: Relationships between nondeterministic and deterministic tape complexities. J. Computer and System Sciences 4, 177–192 (1970)