M1 Internship Report

On Recoloring of d-Degenerate Graphs

Ni Luh Dewi SINTIARI
Laboratoire de GSCOP (Institut Polytechnique de Grenoble)
May 22 - August 25, 2017

Supervised by: Nicolas BOUSQUET, Aurelie LAGOUTTE

Abstract

Graph recoloring problem consists in transforming a given proper (vertex) coloring of a graph into another given proper coloring by changing the color of exactly one vertex in each step, while still maintaining the properness of the coloring. On the class of d-degenerate graph, it is known that this is always possible when the number of colors is $k \geq d + 2$. However, the numer of steps to do such a tranformation can be exponential. It has been proved that when $k \geq 2(d + 1)$, there exists a transformation using $\mathcal{O}(n)$ steps where n is the number of vertices in the graph. This article presents a method of tranformation on a specific class of d-degenerate graphs, namely the class of d-paths. We aim to achieve the better lower bound on k while maintaining the linear number of steps on this class of graphs. In particular, we will prove the following: any two colorings of a d-path using $k \geq 2d + 1$ colors can be transformed one into each other in at most $((2d+1)^2+18)n$ steps. This method can be extended to a larger subclass of d-degenerate graphs, namely the class of d-trees, which is a superclass of d-paths.

1 Introduction

1.1 Coloring, Recoloring, and Connectivity of Recoloring Graph

Reconfiguration problems consist in finding step-by-step transformations between two feasible solutions such that all intermediate results are also feasible. One of the graph problems that can be investigated as a reconfiguration problem is vertex coloring. Cereceda et al. [1] formally initiated the investigation of the reconfiguration of graph colorings.

We recall some basic notions of graphs. In this paper, we restrict our discussion to the class of simple undirected graphs. An *undirected graph* G is an ordered pair $G = (V, E)$ where V is the set of *vertices* (some authors may call it nodes) together with a set E of *edges* which is a subset of unordered pairs of vertices. An undirected graph G is called *simple* if it does not contain any loop or multiple edge. Moreover, for any vertex u and v, we say that u is *adjacent* to v if $uv \in E$. In this case, u is a *neighbor* of v and vice versa. The set of neighbors of v is denoted by $N(v)$. For a set $S \subset V$, $N(S) = \bigcup_{s \in S} N(s) \backslash S$ is the set of neighbors of the vertices in S. Some other terminologies about graphs that are used in this paper are given in the Appendix.

Given a graph G, an integer k, and a set of colors $\{1, 2, ..., k\}$, a *k-color assignment* of G maps every vertex of G to a color in $\{1, ..., k\}$. A k-color assignment is said to be a *(proper) k-coloring*, if no two adjacent vertices receive the same color. If G has a k-coloring, then G is called *k-colorable*. The *chromatic number* $\chi(G)$ of a graph G is the smallest value of k such that G admits a k-coloring. The graphs having chromatic number equals to k is called *k-chromatic graphs*.

Definition 1.1. Let G be a k-colorable graph. The *k-recoloring graph* of G, denoted by $R_k(G)$, is the reconfiguration graph whose set of vertices consists of all possible k-coloring of G, with two k-colorings joined by an edge in $R_k(G)$ if and only if they differ in color on precisely one vertex of G.

Roughly speaking, for two vertices u_α, u_β of $R_k(G)$ which correspond to two k-colorings α and β, u_α and u_β are *adjacent* in $R_k(G)$, if α and β differs on exactly one vertex. Note that two colorings equivalent up to color permutation are distinct vertices in the recoloring graph. Two vertices u_α and u_β are connected in $R_k(G)$ if there exists a sequence of step-by-step transformation from α to β. This sequence of transformation is called *reconfiguration sequence*, and it corresponds to a path that goes from u_α to u_β. Recall that a graph G is said to be *connected* if any two of its vertices are linked by path. In terms of connectivity, we say that G is *k-mixing* if $R_k(G)$ is connected. Having that $R_k(G)$ is k-mixing means that for any k-coloring α and β, there exist a reconfiguration sequence transforming α into β, and vice versa.

A simple way to prove that a graph G is not k-mixing is to exhibit a *frozen* k-coloring of G, namely a k-coloring of G in which all vertices are adjacent to vertices of all other colors. Such a coloring becomes an isolated vertex in $R_k(G)$. Deciding whether a graph is k-mixing is **PSPACE**-complete for $k \geq 4$. Indeed, Cereceda et al. [1] gave two equivalent characterizations of 3-mixing bipartite graphs and proves that deciding if a given bipartite graph is 3-mixing is **coNP**-complete. In addition, Cereceda et al. [1] show that when $k = \chi(G)$, then G is not k-mixing for $k \in \{2, 3\}$. On the other hand, for all $k \geq 4$ there are k-chromatic graphs that are k-mixing, and k-chromatic graphs that are not k-mixing.

The complexity of finding a reconfiguration sequence for given two k-colorings also receives a considerable attention. This problem turns out to be solvable in polynomial time for $k \leq 3$. Bonsma and Cereceda give a polynomial time algorithm for exhibiting a path between the two 3-colorings, if it exists. For any $k \geq 4$, examples have been explicitly constructed where any reconfiguration sequence between two colorings has exponential length. In a recent work, Johnson et al. [5] showed that even finding the shortest reconfiguration sequence, is solvable in polynomial time for $k \leq 3$. The case $k \leq 3$ is, in some sense, is a little bit surprising since the underlying decision problem of determining whether a graph is 3-colorable is NP-complete.

This problem leads us to the question of determining the *recoloring diameter* of $R_k(G)$ (see Appendix for graph diameter), which will be denoted by $diam(R_k(G))$ for a given k-mixing graph G. This diameter gives an upper bound on the length of the best reconfiguration sequence for any pair of k-colorings α and β. Nevertheless, $diam(R_k(G))$ may be polynomial for some well-structured class of graphs, for example when we restrict to graphs of *bounded degeneracy*.

1.2 Background and Motivation

The question of connectivity of a k-recoloring graph has been observed by researchers in the statistical physics community under the name of *Glauber dynamics* of an anti-ferromagnetic Potts model at zero temperature. This research has a tight relation with *rapid mixing* of Markov chains used to obtain efficient algorithms for almost uniform sampling of k-colorings of a given graph. Consider a Markov chain (P, Ω, Π) where P denotes the transition matrix, Ω denotes the state space, and Π denotes the unique stationary distribution. Recall that the variation distance between two distributions μ, ν on Ω is defined as $d_{TV}(\mu, \nu) = \frac{1}{2} \sum_{x \in \Omega} |\mu(x) - \nu(x)|$. We are interested in the mixing time T_{mix} defined as the time to get close to the stationary distribution, $T_{mix} = \max_{x \in \Omega} \min t : d_{TV}(P^t(x, \cdot), \Pi(\cdot)) \leq \frac{1}{e}$.

For a graph G and a value of k, denote the Glauber dynamics for the k-colorings of G by $M_k(G) = (X_t)_{t=0}^{\infty}$. The state space of $M_k(G)$ is the set of k-colorings of G, the initial state X_0 is an arbitrary coloring, and its transition probabilities are determined by the following procedure.

1. Select a vertex v of G uniformly at random.

2. Select a colour $c \in \{1, 2, ..., k\}$ uniformly at random.

3. If recoloring vertex v with color c yields a proper coloring, then set X_{t+1} to be this new coloring. Otherwise, set $X_{t+1} = X_t$.

The relation between $M_k(G)$ and k-recoloring graph of G is that a simulation of the chain corresponds to a walk in $R_k(G)$, since two k-colorings α, β of G form an edge of $R_k(G)$ if and only if $Pr(X_{t+1} = \beta | X_t = \alpha) > 0$, in which case $Pr(X_{t+1} = \beta | X_t = \alpha) = 1 \cdot k|V|$. In this case, $M_k(G)$ is irreducible if and only if G is k-mixing. Thus the fact that a graph is k-mixing is a necessary condition for its Glauber dynamics Markov chain to be rapidly mixing. For further discussion about this topic, the readers can refer to [1].

1.3 Some Known Results

Chordal graph is a graph containing no induced cycle of length more than 3 (see Appendix). A graph G has degeneracy d if and only if there is an (elimination) ordering $v_1, v_2, ..., v_n$ of its vertices such that for $1 \leq i \leq n$, the vertex v_i has at most d neighbors in $\{v_{i+1}, v_{i+2}, ..., v_n\}$. In this case, we say that G is *d-degenerate*. In this paper we will call such an ordering as *vertex ordering*. Moreover, in the following discussion, the naming of the vertices always based on its vertex ordering.

Theorem 1.2. (Cereceda et. al., [1]) *For a d-degenerate graph G with n vertices and $k \geq d + 2$, recoloring graph $R_k(G)$ is k-mixing.*

The theorem says that for any two given k-colorings of G using at least $(d + 2)$ colors, there always exists a recoloring between them. However this method gives an exponential number of steps [1]. In particular, Cereceda et. al. [1] gives an upper bound for the diameter of recoloring graph, as stated in the following lemma.

Lemma 1.3. (Cereceda et. al., [1]) *For any d-degenerate graph G with n vertices, and every $k = d + 2$, $diam(R_k(G)) \leq n(d + 1)^n$.*

Proof. By induction on the number of vertices, we will prove that any given two k-colorings α, β of G can be transformed one into each other by recoloring every vertex at most $(d+1)^n$. Since G is d-degenerate, then there exists a vertex ordering $v_1, v_2, ..., v_n$. Let G' be the subgraph induced by $V \backslash v_1$. Note that G' is still d-degenerate. Let α', β' be the restriction of α, β respectively to G'. By induction hypothesis, α' can be transformed into β' by recoloring every vertex of G' at most $(d + 1)^{n-1}$. Hence, when including the vertex v_1 of G, everytime we need to change the color of a vertex $u \in N(v_1)$, then we may have to change the color of v_1 beforehand. Hence, during the transformation of α' into β', since the degree of v_1 in $\{v_2, v_3, ..., v_n\}$ is at most d, then we would have to change the color of v_1 at most $d(d+1)^{n-1} < (d + 1)^n$, and in the last step, we may have to change the color of v into $\beta(v)$ if necessary. Thus, to transform α into β, every vertex have to be recolored at most $(d + 1)^n$. Since there are n vertices, the number of steps is at most $n(d + 1)^n$.

□

We are interested in obtaining a better upper bound on the number of required steps for transforming any k-coloring of d-degenerated graphs. The conjecture by Cereceda et. al. [1] below is well-known.

Conjecture 1.4. (Cereceda et. al., [1]) *For any d-degenerate graph G and every $k \geq d + 2$, $diam(R_k(G)) = O(n^2)$*

No general result is known so far on this conjecture, but this conjecture was proven to be true on some particular classes of graphs, for example on d-degenerate chordal graphs, as stated in Theorem 1.5.

Theorem 1.5. (Bonamy et. al., [2]) *For every d-degenerate chordal graph and every $k \geq d+2$,*

$diam(R_k(G)) = O(n^2)$

Sketch of proof. In their proof, Bonamy et. al. introduced the notion of *k-color-dense* graphs, that is a class C of k-colorable graphs in which every graph $G \in C$ satisfies either (i) or (ii) of the following (again, the readers can check Appendix for undefined terminologies).

(i) G is the disjoint union of cliques, each of which has at most k vertices

(ii) G has a separator S, and $G - S$ has components D and D' with vertices $u \in D$ and $v \in D$ s.t.

  (a) $|D| = 1$ or $|D \cup S| \leq k$

  (b) $S \subseteq N(v)$, and

  (c) identifying u and v in G results in a k-color-dense graph G'

Then they showed that for $k \geq 1$ and $l \geq k+1$, any k-color-dense graph G on n vertices satisfies $diam(R_k(G)) \leq 2n^2$. Furthermore, they also showed that for fix $k \geq 1$, every k-colorable chordal graph is k-color dense. Hence knowing the fact that a chordal graph has degeneracy d if and only if it is $(d+1)$-colorable, we can conclude the proof. The readers should refer to [2] for the complete proof.

The lower bound on the number of colors k of Theorem 1.5 is necessary. Indeed, on a complete graph with d vertices, and $k = d+1$, then $R_k(G)$ is not k-mixing, since all vertices of $R_k(G)$ are isolated. Moreover, the upper bound on the diameter cannot be improved, for instance on paths (Bonamy et. al. [2]). In other words, one may need a quadratic number of steps to transform a 3-coloring of a path into another 3-coloring.

In addition, Bousquet and Perarnau [4] gives a lower bound on k that yields linear diameter of the recoloring graph, for the class of d-degenerate graph, as stated in the following theorem.

Theorem 1.6. Bousquet, Perarnau, [4] *For every d-degenerate graph G and every $k \geq 2(d+1)$, then $diam(R_k(G)) \leq (d+1)n$.*

Proof. The proof goes by induction on the number of vertices. Let G be d-degenerate graph, and α, β be two k-colorings of G. Since G is d-degenerate, then there exists a vertex ordering $\{v_1, v_2, ..., v_n\}$. Furthermore, let G' be the subgraph of G induced by $V \backslash v_1$,. Clearly, G' is still d-degenerate, and with vertex ordering $\{v_2, v_3, ..., v_n\}$. Let α', β' be the restriction of α, β on G'. By induction hypothesis, α', β' can be transformed one into each other by recoloring every vertex at most $(d+1)$ times.

Now consider again the initial graph G, and include v_1 during the transformation from α' into β'. At each step of the recoloring process, let some vertex u is recolored from color a into b. If $u \notin N(v_1)$ or v_1 is not currently colored with b then we are done, as we do not need to perform any recoloring on v_1, so the coloring is still proper in G. The worse case, if $u \in N(v_1)$ and v_1 is currently colored with b, then we have to add some recoloring steps to v_1 before we can recolor u with b. Let $t_1, t_2, ..., t_l$ be the steps in the recoloring sequence when a vertex $u \in N(v_1)$ is recolored, and let c_t be the new color assigned to it at time t. Assume that the bad case happens at time t_i, with $i \leq l$. Note that, in order to maintain the proper coloring, we have to assign to v_1, a color distinct from the colors in $N(v_1)$ which contains at most d colors (because the degree of v_1 is at most d). So, we have at least $k - d \geq d + 1$ choice of colors for v_1. Hence we can assign to v_1, a color $c \notin \{c_{t_i}, c_{t_{i+1}}, ..., c_{t_{i+d-1}}\}$. By this recoloring, then v_1 will not require any recoloring before time t_{i+d} in the sequence.

Let $t_{i_1}, ..., t_{i_s}$ be the recoloring steps of v_1 in the original sequence. Obviously, $i_{j+1} - i_j \geq d+1$ for all $j < s$. Note that $l \leq (d+1)d$ because $\deg(v_1) \leq d$ and every vertex in $V \backslash v_1$ requires at most $(d+1)$ recolorings. Moreover, $i_s \leq l$ because the recoloring times of v_1 can not exceed the recoloring times of $N(v_1)$. It yields that $s \leq d$. Therefore, during the transformation of α' into β', we will only recolor v_1 at most d times. At the end of the procedure, we may need to recolor v_1 into $\beta(v_1)$ if necessary, which means that v_1 is recolored at most $(d+1)$ times during the transformation from α into β.

□

Cereceda et al. [1] also observed the graph recoloring diameter on classes of graphs with *bounded treewidth*, namely d-degenerate graphs of treewidth d are $(d+2)$-mixing. Then Bonamy and Bousquet [3] proved that the recoloring diameter for bounded treewidth graphs is polynomial, as stated in the following theorem.

Theorem 1.7. Bonamy, Bousquet, [3] *For every graph G having treewidth $tw(G)$, and every $k \geq tw(G)+2$, then $diam(R_k(G)) \leq 2(n^2+n)$.*

1.4 Problems, Results, and Outline

Knowing that for $k \geq 2(d+1)$, the graph recoloring of d-degenerate graph G has a linear diameter, we may be curious to know whether the diameter is still linear if we reduce the number of colors k. Determining precisely for which value the diameter becomes linear is a central question for physicists who want to generate a coloring at random. In this paper, we are going to analyse if we can achieve a better lower bound on the number of colors k, in which the number of steps needed to transform any pair of k-coloring of d-degenerate graphs into any other is linear. A simple class of d-degenerate graphs that can be a good start is the so-called class of *d-paths*.

In this paper, we discuss some algorithms for recoloring a d-path G. We try to improve the lower bound of k in terms of d in which the number of steps needed for the recoloring is linear in n. In particular, we prove that for the class of d-paths, there exists step-by step transformation to transform a k-coloring into another k-coloring within a linear number of steps. We state it formally in Theorem 1.8. Furthermore, we also show that this result can be extended to the superclass of d-paths, namely the class of d-trees, as we will state in Theorem 1.9.

Theorem 1.8. *Let G be a d-path of n vertices and $k = 2d+1$, and γ, δ be any k-coloring of G. Then γ, δ can be transformed each other in at most $(4d^2+4d+13)n$ steps.*

Theorem 1.9. *Let G be a d-tree of n vertices and $k = 2d+1$, and γ, δ be any k-coloring of G. Then γ, δ can be transformed each other in at most $(4d^2+16d+11)n$ steps.*

We find a good lower bound for some "special" colorings of d-paths (we will explain it more clearly in Section 3). However, the lower bound on k that we obtain in the Theorem 1.8 is still not good enough, as it is only one value lower than the bound given by Bousquet and Perarnau in [4]. In addition, the proof of Theorem 1.8 and Theorem 1.9 is algorithmic, so we have a step-by step transformation to transform any two k-colorings one into each other.

This paper is organized as follows. Section 1, as the readers have seen, contains the introduction of our problems. In Section 2, we give some basic terminologies about graphs and some notations that will be used in the whole paper. We also define some recoloring technics and present some algorithms to recolor certain types of recolorings on d-paths. Later in Section 3, we give our main results that have been described previously. In the end, the Appendix provides some basic concepts about graph theory. We also provide some examples to illustrate the recoloring technics that we have implemented.

2 Preliminaries

In this section we give some basic terminologies and notations that are used in the whole paper. We also discuss some basic properties that are important to prove our result.

2.1 Basic Terminologies and Notations

We will start by analysing our problem on subclasses of d-degenerate graphs, namely the class of *d-degenerate paths* (or *d-paths*). An example of d-path can be seen in Figure 1.

Definition 2.1. A *d-path* is a graph created from a single path (or 1-path) by adding edges between each two vertices which are at distance at most d on the path.

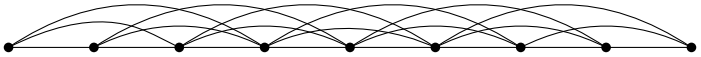


Figure 1: A 3-path with 9 vertices

In the following discussion, we will assume that G is k-colorable for some value of k that will be specified later on. Moreover, we can assume that G is connected, otherwise we can just consider it as a disjoint union of connected graphs, then do the recoloring on its connected components. For a k-coloring α of G, the color of a vertex v in G is denoted by $\alpha(v)$. We say that a k-coloring β is a *target coloring* if we want to recolor G with β. In this case, for any vertex $v \in V$, the color $\beta(v)$ is called the *target color* of v. Now fix a vertex $u \in V$. We say a vertex $v \in N(u)$ is *u-good* if it is not currently colored with $\beta(u)$, the target color of u. Otherwise, we say v is *u-bad*. Furthermore, we also define a notion of *frozen vertex*. That is, a vertex v that can not be recolored by any color (because $N(v) \cup \{v\}$ uses all the colors). Otherwise, we call it *unfrozen*. In this case, a color that can be used to recolor a vertex v is called *v-free color*. Note that a d-path G is $(d+1)$-colorable as following the vertex ordering of G, we can always color vertex v_i by the color that does not appear in its d neighbors of $\{v_1, v_2, ..., v_{i-1}\}$.

Let G be a d-path. First, for simplification, assume that $|V| = n = kl$ for some integer k, l. We give an order to the vertices of G based on the degeneracy. We take a vertex v_1 with degree at most d, then in each step, we take a vertex v_i that is adjacent to v_{i-1} s.t. $|N(v_i) \cap \{v_1, ..., v_{i-1}\}| \leq d$. For the vertex ordering $(v_1, v_2, ..., v_n)$, we will say that v_{i+1} is on the right of v_i and v_{i-1} is on the left of v_i. Based on its ordering, we denote the vertices as $V = \{u_1^1, u_2^1, ..., u_k^1, u_1^2, u_2^2, ..., u_k^2, ..., u_1^{i-1}, u_2^{i-1}, ..., u_k^{i-1}\}$. In this case, the subset $\mathbf{w}_i = \{u_1^i, u_2^i, ..., u_k^i\}$ of V for $i \in \{1, 2, ..., l\}$ is called a *k-block* (or just a block) of G. Hence, a d-path with kl vertices has l blocks of length k, and vertex u_j^i is the j^{th} *vertex* of the block $\mathbf{w}_i$. Furthermore, two vertices $v_t, v_{t'}$ are said to have *ordering distance* equals to D if $t' - t = D$. For example, the ordering distance of vertex u_1^i and u_k^i is k. In addition, $d(\alpha)$ denotes the minimum over the ordering distance in the vertex ordering of any pair of two vertices that are colored with the same color in α.

Consider a block $\mathbf{w}_i$ of G. A sub-sequence of vertices of $\mathbf{w}_i$ is called *sub-block* of $\mathbf{w}_i$. For any block $\mathbf{w}_i$, and any integer $m \in \{1, 2, ..., k\}$, the m leftmost (resp. rightmost) vertices of $\mathbf{w}_i$ is called the *m-prefix* (resp. *m-suffix*) of $\mathbf{w}_i$. Furthermore, for any vertex u_j^i of G, $N_L(u_j^i)$ and $N_R(u_j^i)$ denote the set of neighbors of u_j^i before and after u_j^i (respectively) in the vertex ordering. Let α be a k-coloring of G, we can write α as $w_1|w_2|...|w_l$ where $w_1, w_2, ..., w_l$ are the coloring of the blocks $\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_l$ respectively. Let α be a k-coloring of a d-path G, S be the sequence of the vertex ordering of G, and S' be a sub-sequence of S. The sequence of colors in α that are used to color the subgraph H induced by S' is called the *pattern of α on H*. For instance, w_i is the pattern of the block $\mathbf{w}_i$ for $i \in \{1, 2, ..., l\}$. In the Definition 2.2, we present two types of "special" colorings on a d-path.

Definition 2.2. Let G be a d-path that is currently colored with a k-coloring α, where every k-block of G is colored with exactly k colors.

(i) α is said to be *symmetric* if every block of G has the same pattern.

(ii) α is said to be *almost symmetric* if the d-suffix of every block of G has the same pattern.

An example of almost symmetric and symmetric coloring on a d-path is given respectively in Figure 2 and Figure 3 (the dashed lines indicate the blocks).

Definition 2.3. Let G be a d-path.

(i) Two symmetric colorings α, β is said to *differ by one transposition*, if $\alpha(u_j^i) = \beta(u_{j+1}^i)$ and $\alpha(u_{j+1}^i) = \beta(u_j^i)$ for every $i \in \{1, 2, .., l\}$ and a fixed value $j \in \{1, 2, ..., k-1\}$, and they agree on all other vertices.

(ii) A block $\mathbf{w}$ of G is said to has a *nice coloring*, if the d-suffix of $\mathbf{w}$ has pattern $(k-d+1, k-d+2, ..., k)$.

An example of two symmetric colorings that differ by one transposition is given in Figure 3 and Figure 4. In this example, the symmetric 6-colorings α_1 and α_2 on 3-path differ by one transposition as the color 3 and 5 are switched.

Remark 2.4. Up to the color permutation, in the next following sections, we will assume that in symmetric coloring, every block of α has pattern $(1, 2, ..., k)$, and in almost symmetric coloring, the d-suffix of every block of α has pattern $(k-d+1, k-d+2, ..., k)$.

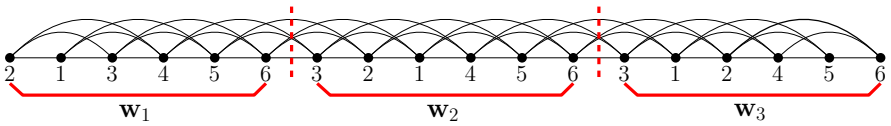


Figure 2: A 3-path colored with an almost symmetric 6-coloring

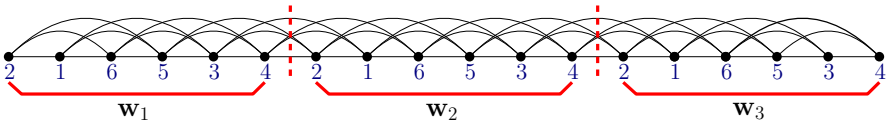


Figure 3: A 3-path colored with a symmetric 6-coloring α_1

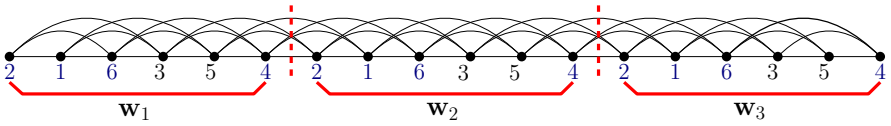


Figure 4: A 3-path colored with a symmetric 6-coloring α_2

2.2 Tools: Some Recoloring Methods on d-Paths

In the following subsection, we present some recoloring algorithms that will be used to recolor a d-path. We say that a k-recoloring method is *proper* if in each step of the recoloring, the coloring is a proper k-coloring of G. Note that a k-recoloring is proper on a d-path only if $k \geq d+1$.

Definition 2.5. A *right-chain-recoloring* up to a vertex v_{end} (similarly, left) on a d-path G (that is currently colored with α) is a recoloring that is started by recoloring the vertex v_j of G with a v_j-free color, then following the vertex ordering $(v_1, v_2, ..., v_n)$ of G, in each step, vertex v_{j+1} which is the vertex after v_j in the ordering, is recolored with $\alpha(v_j)$ which is the former color of the vertex v_j, and the recoloring stops in vertex v_{end} (i.e. this vertex is the last vertex recolored). Similarly, the *left-chain-recoloring* starts by recoloring a vertex v_j then following the ordering, at each step, v_{j-1} is recolored with the former color of v_j.

Remark 2.6. We say that we apply a *right-chain-recoloring (similarly, left) by skipping a vertex* v_t if after recoloring the vertex v_{t-1}, we recolor the vertex v_{t+1} (instead of the vertex v_t) with the former color of v_{t-1}. Note that it is possible to skip more than one vertex.

An example to illustrate this recoloring is given in Example A.1.

Proposition 2.7. *Let G be currently colored with α. The right-chain-recoloring up to a vertex v_t (similarly, left) is a proper recoloring method if $d(\alpha) \geq d+2$.*

Proof. First of all, note that this bound is necessary. Imagine that if G has a pair of vertices $v_j, v_{j'}$ of ordering distance less than or equal to $(d+1)$ that are colored the same, then we will not be able to recolor v_{j+1} with $\alpha(v_j)$ because $v_{j'} \in N(v_{j+1})$ is currently colored with $\alpha(v_j)$. Furtermore, if $d(\alpha) \geq d+2$, then everytime we recolor a vertex v_{j+1} with $\alpha(v_j)$, we will only reduce the value of $d(\alpha)$ by at most one, i.e. the ordering distance of any pair of vertices that are colored the same is at least $(d+1)$. Hence, it still gives a proper coloring. □

Proposition 2.8. *Let α be a k-coloring of G and v_i, v_{i+D} be two vertices that are colored with a same color c and no vertex $v_i + t$ with $0 < t < D$ colored with c. Then when applying right(left)-chain-recoloring, we can skip the vertices $v_{i+1}, v_{i+2}, ..., v_{i+D-(d+2)}$.*

Proof. Based on Proposition 2.7, in order to maintain the proper coloring of G, in each step, we have to preserve the value of $d(\alpha)$ to be at least $(d+1)$. In this case, recoloring v_j with $\alpha(v_i)$ still preserves the property, hence the proposition follows.

□

We recall one result of Bonamy and Bousquet [3] as stated in the following lemma. Let us call this recoloring method a *clique-recoloring*.

Lemma 2.9. Bonamy, Bousquet, [3] *For $k \geq n+1$, any two k-coloring α, β of a complete graph K_n can be transformed one into each other by recoloring every vertex at most twice.*

Proof. Let α, β be two colorings of the complete graph K_n, where the graph is initially colored with α. Our goal is to recolor it into β in such a way that every vertex is recolored at most twice. We build a digraph D with $V(D) = V(K_n)$ and for any edge $uv \in E(K_n)$, uv is an arc of D if $\alpha(v) = \beta(u)$, i.e. if v prevents the recoloring of u into its target color. Note that as all the vertices of K_n are colored differently both in α and β, then for any $v \in V(K_n)$, its out-degree $d^+(v) \leq 1$ (resp. in-degree $d^-(v) \leq 1$), otherwise there will be more than one vertex that are colored with $\beta(v)$ (resp. $\alpha(v)$). Hence, D is a disjoint union of directed paths and circuits.

In order to recolor a directed path $P = (v_0, v_1, ...v_m)$, we start by recoloring v_m with its target color (note that as $d^+(v_m) = 0$ then no vertex is being colored with $\beta(v_m)$). It yields that $d^+(v_{m-1}) = 0$ because v_m is no longer colored with $\beta(v_{m-1})$. We do it iteratively until we finish the recoloring on P. In the end we will have that every vertex of P is an isolated vertex in D. Now, to recolor a circuit $C = (v_0, v_1, ...v_m, v_0)$, as we have $k \geq n+1$ then there exist a v_0-free color c to recolor vertex v_0. Thus now we have a directed path $C' = (v_0, v_1, ...v_m)$ and we can apply the similar strategy as on the path P. Note that in this case, we will recolor the vertex v_0 twice. In the end of the Algorithm, we will have recolored all the directed paths and circuits of D. It yields that K_n is colored with β. Moreover, in this transformation, every vertex is recolored at most twice.

□

Remark 2.10. Lemma 2.9 holds for any graph G of n vertices in which the initial coloring α and the target coloring β are colored with exactly n colors of $k \geq n+1$ available colors. This is obvious as the coloring α and β on G will be a proper coloring of K_n. We can even extend the Algorithm into the coloring of an induced subgraph of G having some particular properties, as stated in Proposition 2.11.

Proposition 2.11. *Let H be an induced subgraph of a graph G, α, β be two k-colorings of G, and α', β' be the restriction of α, β in H. Let $\alpha'(H), \beta'(H)$ be the set of colors used in α, β respectively, and $C = \alpha'(H) \cup \beta'(H)$. Assume that in both α' and β', all vertices are colored differently and no vertex in $N(H)$ use any color of C. If $|C| \geq |H| + 1$ then α', β' can be transformed one into each other by recoloring every vertex at most twice.*

Proof. As no vertex in $N(H)$ uses any color of C, and $V(H)$ are colored differently in α', β', then we can obey the colors of $N(H)$ and apply simple coloring as in Lemma 2.9. □

In the following, we provide a method to transform an almost symmetric k-coloring α into a symmetric coloring β. In order to transform it, we will first transform α into an intermediate k-coloring γ by applying Algorithm 1, then we will transform γ into β by applying Algorithm 2. We remind you that when transforming γ into β, we will make every block has a nice coloring, consecutively from the leftmost block. We will explain roughly what we are doing here, then we will explain it more carefully in the next discussion. This intermediate coloring has a nice property, that when a block $\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_{i-1}$ have pattern $1, 2, ..., k$, then we will be able to apply clique-recoloring on the $(k-d-1)$-prefix of $\mathbf{w}_i$ such that the pattern becomes $1, 2, ..., k-d-1$. Then, we will have $k-d$ as a u^i_{k-d}-free color, and we can recolor u^i_{k-d} with $k-d$ and apply right-chain-recoloring up to the vertex u^i_k so that $\mathbf{w}_i$ will have pattern $1, 2, ..., k$.

Now note that as α is almost symmetric, then it might be the case that $d(\alpha) = d+1 < d+2$, i.e. there exists some pairs of vertices colored with the same color within ordering distance $d+1$. Let call such pair of vertices a *disturbing pair*.

Definition 2.12. An *intermediate k-coloring* of a d-path G is a k-coloring such that every block $\mathbf{w}_i$ has pattern: $(c^i_1, ..., c^i_{k-d-1}, (k-d+1), ..., k, c^i_k)$ where $(c^i_1, ..., c^i_{k-d-1})$ are $(k-d-1)$ different color of $\{1, 2, ..., k-d\}$, and $c^i_k \in \{1, 2, ..., k-d-1\}$.

Algorithm 1: Transforming an Almost Symmetric k-Coloring into an Intermediate k-Coloring on a d-Path G

Input: Almost symmetric k-coloring of G
Output: Intermediate k-coloring of G
Recolor u^l_k with 1, then apply left-chain-recoloring starting from u^l_k up to u^1_{k-d} by skipping some vertices according to the following rules.

```
for i ∈ {2, 3, ..., l} do
    if (u^{i-1}_{k-d}, u^i_1) is a disturbing pair then
        skip vertex u^i_1
        if α(u^i_2) = (k − d) then
            skip also u^i_2 and recolor v with α(u^i_3)
        end
    else
        if α(u^i_1) = (k − d) then
            skip vertex u^i_1
            if α(u^i_2) ≠ α(u^{i-1}_{k-d}) then
                skip also u^i_2
            end
        end
    end
end
```

Theorem 2.13. *Algorithm 1 is a proper recoloring method for d-path with n vertices. Moreover, each vertex is recolored at most once.*

Proof. We will prove the first statement. Let G is initially colored with an almost symmetric k-coloring. Let $\mathbf{w}_{i-1}, \mathbf{w}_i$ be two consecutive blocks in G containing a disturbing pair. We recall that $\mathbf{w}_{i-1}||\mathbf{w}_i$ is numbered as follows

$$u^{i-1}_1, ..., u^{i-1}_{k-d}, u^{i-1}_{k-d+1}, ..., u^{i-1}_k || u^i_1, ..., u^i_{k-d}, u^i_{k-d+1}, ..., u^i_k$$

Notice that the only possibility for a disturbing pair is (u^{i-1}_{k-d}, u^i_1) since the d-suffix of $\mathbf{w}_{i-1}$ and $\mathbf{w}_i$ has a fixed pattern $((k-d+1), (k-d+2), ..., k)$, and all the colors in $\mathbf{w}_{i-1}$ (and $\mathbf{w}_i$) are distinct.

If (u^{i-1}_{k-d}, u^i_1) is a disturbing pair, then $\alpha(u^{i-1}_{k-d}) = \alpha(u^i_1)$. In this case, we can not recolor u^{i-1}_k with $\alpha(u^{i-1}_1)$ (because its neighbor and u^{i-1}_{k-d} is colored with $\alpha(u^{i-1}_1)$). Moreover, it also can not be colored with $(k-d)$ (based on the definition of intermediate coloring). Hence we

have to recolor u_k^{i-1} with a color $c \in \{1, 2, ..., k-d\} \setminus \{\alpha(u_{k-d}^{i-1}), (k-d)\}$. We analyse the following cases.

(i) If (u_{k-d}^{i-1}, u_1^i) is a disturbing pair, then $\alpha(u_{k-d}^{i-1}) = \alpha(u_1^i)$, so we have to skip vertex u_1^i. If then $\alpha(u_2^i) \neq (k-d)$, we can recolor u_k^{i-1} with $\alpha(u_2^i)$. Otherwise, we should also skip u_2^i. Obviously, $\alpha(u_3^i) \neq \alpha(u_{k-d}^{i-1}), (k-d)$, so recoloring u_k^{i-1} with $\alpha(u_3^i)$ still gives a proper coloring.

(ii) If (u_{k-d}^{i-1}, u_1^i) is not a disturbing pair, and $\alpha(u_1^i) \neq (k-d)$, then we can recolor u_k^{i-1} with $\alpha(u_1^i)$, otherwise we have to skip vertex u_1^i. If then $\alpha(u_2^i) \neq \alpha(u_{k-d}^{i-1})$, then recolor u_k^{i-1} with $\alpha(u_2^i)$, otherwise we have to skip also this vertex. As in the previous case, $\alpha(u_3^i) \neq \alpha(u_{k-d}^{i-1}), (k-d)$, so recoloring u_k^{i-1} with $\alpha(u_3^i)$ gives a proper coloring.

Note that by this recoloring, we obtain our target coloring, by recoloring every vertex at most once. An example to illustrate this algorithm is provided in Example A.4. □

Algorithm 2:
Transforming an Intermediate k-Coloring into a Symmetric k-Coloring on a d-Path

Input: Intermediate k-coloring of G
Output: Symmetric k-coloring of G
for $i = 1$ *to* l **do**
  Apply clique-recoloring on the $(k-d-1)$-prefix of $\mathbf{w}_i$
   with target coloring $(1, 2, ..., k-d-1)$
  Recolor u_{k-d}^i with $(k-d)$
  Apply right-chain-recoloring starting at u_{k-d}^i up to u_k^i
end

Theorem 2.14. *Algorithm 2 is a proper recoloring method for the d-path with n vertices. Moreover, each vertex is recolored at most twice.*

Proof. Let G be currently colored with an intermediate coloring γ, and our target coloring is a symmetric coloring β. Let γ', β be the restriction of γ, β respectively in H, the subgraph of G induced by $\{u_1^i, ..., u_{k-d-1}^i\}$. Notice that $C = \gamma(H) \cup \beta(H) = \{1, 2, ..., k-d\}$ and every vertex in $N(H)$ is colored with a color from $\{k-d+1, ..., k\}$. Hence Proposition 2.11 holds on H, i.e. γ' can be transformed into β' by recoloring every vertex of H at most twice.

Once we have finished the recoloring of H, then we will have $k-d$ as a u_{k-d}^i-free color on the block $\mathbf{w}_i$ since no vertex in $N(u_{k-d}^i$ is being colored with $k-d$. Hence we can recolor u_{k-d}^i with $(k-d)$. This still gives proper coloring as no neighbor of u_{k-d}^i is currently colored with $(k-d)$. Then, we can apply right-chain-recoloring on the $(d+1)$-suffix of G to recolor it into $((k-d), (k-d+1), ..., k)$. Note that this is possible because the current coloring of G ensures that $d(\alpha') \geq d+2$, where α' is the current coloring.

By repeating this step for all the block of G, we will obtain the coloring β. The number of steps needed is at most $2n$ because every vertex is recolored at most twice. An example to illustrate this recoloring method is provided in Example A.2.

□

3 Main Results

3.1 Recoloring d-Paths

We know that for any d-degenerate graph G, and a set of k colors where $k \geq d+2$, $R_k(G)$ is connected. Moreover, if G is chordal graph, then the diameter of $R_k(G)$ is quadratic [2] Bousquet et al. [4] give a lower bound on k in order for the diameter of $R_k(G)$ to be linear, namely $k \geq 2(d+1)$. We are interested in improving this lower bound. We want to look whether this bound can be improved on a small class of graphs, namely the class of d-path.

In Theorem 1.8, we prove that we can still achieve a linear number of steps when the number of colors is reduced by one, i.e. when $k = 2d + 1$.

We build the proof by doing a sequence of transformation. First, we transform an arbitrary k-coloring of G into almost symmetric k-coloring, then we transform it into symmetric k-coloring. In the following section, we show that any two special colorings (symmetric and almost symmetric) on a d-path can be transformed into any other in $f(k)n$ steps where $f(k)$ is a polyomial function with parameter k. Then we will see that we can transform any k-coloring of a d-path into special coloring in linear number of steps.

Lemma 3.1. *Let G be a d-path, $k \geq d + 3$, and α be an almost symmetric k-coloring of G. The coloring α can be transformed into a symmetric k-coloring β by recoloring every vertex of G at most 3 times.*

Proof. We recall that in α, the d-suffix of every block of G have pattern $k-d+1, ..., k$, and up to the color permutation, we can assume that the pattern of every block of G in β is $(1, 2, .., k)$. In order to transform α into β, we can do a sequence of transformation: $\alpha \to \gamma \to \beta$ where γ is an intermediate coloring. Based on Theorem 1 and Theorem 2, transforming α into γ and γ into β requires that every vertex to be recolored at most once and twice respectively. Hence to transfor α into β, every vertex have to be recolored at most 3 times.

□

Lemma 3.2. *Let G be a d-path on n vertices, $k \geq d + 3$, and α, β be two symmetric k-colorings of G. If α and β differ by one transposition, then α can be transformed into β by recoloring every vertex in G at most twice.*

Proof. The summary of this method can bee seen in Algorithm 3, and an example to illustrate how the method works is provided in Example A.3. Let $\mathbf{w}_1$,$\mathbf{w}_2$,...,$\mathbf{w}_l$ be the l consecutive k-blocks of G, where $\mathbf{w}_i = (u_1^i, u_2^i, ..., u_k^i)$. Assume that for any block $\mathbf{w}_i$, the coloring α, β differ in vertices u_t^i and u_{t+1}^i for a fixed $t \in \{1, 2, ..., k-1\}$. Note that, $u_k^1 \notin N(u_1^1)$, so we can recolor u_1^1 with $\alpha(u_k^1)$. Furthermore, since α is a symmetric coloring, then $d(\alpha) = k \geq d + 2$, so we can do a right chain recoloring up to vertex u_{t+1}^l. Now we obtain the coloring α' in which $\alpha'(u_j^i) = \alpha(u_{j-1}^i)$ for $j \in \{2, ..., k\}, i \in \{1, 2, ..., l\}$ and $\alpha'(u_1^i) = \alpha(u_k^{i-1})$ for every $i > 1$. Roughly speaking, we shift the coloring one step to the right. Consider the subgraph G' induced by $\{u_1^1, ..., u_k^1, u_1^2, ..., u_k^2, ..., u_1^{i-1}, ..., u_{t+1}^{i-1}\}$ with the coloring α'. Note that in G', $d(\alpha') \geq d+3$ so we can do a left chain recoloring, by first recoloring the vertex u_t^l with $\alpha(u_{t+1}^l)$ and skip every vertex u_{t+1}^i for $i \in \{1, 2, ..., l-1\}$. This step gives us the coloring β, satisfying $\beta(u_j^i) = \alpha(u_j^i)$ for every $j \neq t, t+1$, $\beta(u_t^i) = \alpha(u_{t+1}^i)$, and $\beta(u_{t+1}^i) = \alpha(u_t^i)$. Note that, every vertex is recolored at most twice, one for each the right and the left-chain-recoloring.

□

Algorithm 3: Transforming Two Symmetric k-Colorings that Differ by One Transposition
Input: Symmetric k-colorings α of G, $t \in \{1, 2, ..., k-1\}$ **Output:** Symmetric k-colorings β of G s.t α, β differ by one transposition - Recolor u_1^1 with $\alpha(u_k^1)$, then apply right-chain-recoloring up to u_{t+1}^l - Recolor u_t^l with $\alpha(u_{t+1}^l)$ - Apply left-chain-recoloring starting at u_t^l up to u_1^1 by skipping u_{t+1}^i for every $i \in \{1, 2, ..., l-1\}$

Lemma 3.3. *Let G be a d-path and $k \geq d + 3$. Any two symmetric k-colorings α, β in G can be transformed one into each other in at most k^2n steps.*

Proof. Note that the transformation of any two symmetric k-colorings α and β can be seen as a sequence of transformation of symmetric k-colorings $\alpha = \alpha_1 \to \alpha_2 \to ... \to \alpha_m = \beta$ in which α_i and α_{i+1} are symmetric k-colorings that differ exactly by one transposition. Let $\alpha = (w|w|...|w)$ and $\beta = (w'|w'|...|w')$ where w, and w' are any permutation of $(1, 2, ..., k)$.

Furthermore, w can be step-by-step transformed into w', where in each step, we only change exactly one transposition. In the worst case, the number of step-by-step transformation needed to transform w into w' is $\frac{1}{2}k(k-1)$, for example when $w = (1, 2, ..., k)$ and $w' = (k, k-1, ..., 1)$. Thus the length m of the sequence of transformation from α into β is also at most $\frac{1}{2}k(k-1)$. By Lemma 3.2, for every $i \in \{1, 2, ..., m-1\}$ we can transform α_i into α_{i+1} in at most $2n$ steps. Hence, the number of steps needed to transform α into β is at most $2n(\frac{1}{2}k(k-1)) \leq k^2 n$.

□

In what follows, we are going to discuss a method to transform an arbitrary k-coloring of a d-path G into an almost symmetric k-coloring. Algorithm 4 gives the summary of this method. Recall the definition of *nice coloring* on a block $\mathbf{w}$ of G, as the coloring of $\mathbf{w}$ in which the d-suffix of $\mathbf{w}$ has pattern $(k-d+1, k-d+2, ..., k)$ (note that we do not care about the coloring of $(k-d)$-prefix of $\mathbf{w}$).

Algorithm 4: Transforming Arbitrary k-Coloring into Almost Symmetric k-Coloring on d-Path G

Input: Arbitrary k-colorings α of G, $t \in \{1, 2, ..., k-1\}$
Output: Almost Symmetric k-colorings β of G
Add a virtual block $\mathbf{w}_{l+1}$ colored with a nice coloring
for $i = l$ *downto 1* **do**
  for $c = k - d + 1$ *to* k **do**
    $u = u_c^i$
    $v_{bad}^R = v \in N(u)$ s.t. $\alpha(v) = c$ if any
    if $\exists!\ v_{bad}^R \in N_R(u)$ **then**
      do Procedure 1
    end
    if $\exists\ v_{bad}^R \in N_R(u) \wedge v_{bad}^L \in N_L(u)$ **then**
      do Procedure 2
      do Procedure 1
    end
    if $\exists!\ v_{bad}^L \in N_L(u)$ **then**
      do Procedure 3
      do Procedure 2
      do Procedure 1
    end
  end
end
Procedure 1
1 Recolor u_{p+1}^R with a u_{p+1}^R-free color
2 Recolor v_{bad}^R with $\alpha(u_{p+1}^R)$
3 Recolor u with c
Procedure 2
4 Recolor u with a u-free color
5 Recolor v_{bad}^L with $\alpha(u)$
Procedure 3
6 Recolor u_p^R with c

Lemma 3.4. *Let G be a d-path and $k = 2d+1$. Consider two consecutive k-blocks $\boldsymbol{w}_{i-1}$ and $\boldsymbol{w}_i$ in G, where $\boldsymbol{w}_i$ has a nice coloring. We can recolor those blocks in such a way that both of them will have a nice coloring, by only recoloring the vertices of $\boldsymbol{w}_{i-1}$ and $(k-d)$-prefix of $\boldsymbol{w}_i$, in at most $6d$ steps.*

Proof. Let G be initially colored with an arbitrary k-coloring α (which is not almost symmetric), and recall that $\mathbf{w}_{i-1}||\mathbf{w}_i$ is written as

$$u_1^{i-1}, u_2^{i-1}, ..., u_k^{i-1}||u_1^i, u_2^i, ..., u_k^i$$

Assume that $\mathbf{w}_i$ has a nice coloring, i.e. the vertices $u_{d+2}^i, u_{d+3}^i, ..., u_k^i$ are colored with $d+2, d+3, ..., k$ respectively. We will iteratively and consecutively recolor the vertices $u_{d+2}^{i-1}, u_{d+3}^{i-1}, ..., u_k^{i-1}$ s.t. the block $\mathbf{w}_{i-1}$ becomes "nice".

Now assume that after some steps, the $(p-1)$ leftmost vertices of the d-suffix of $\mathbf{w}_{i-1}$ have been recolored with their target color. Now we want to recolor vertex u_{d+p+1}^{i-1} with $(d+p+1)$. For simplification, let $u = u_{d+p+1}^{i-1}$ and $c = d+p+1$. Note that we will not be able to do so trivially, if and only if $N(u)$ contains at least one u-bad vertex. Moreover, we note that $N(u)$ can only contain at most two u-bad vertices (one in $N_L(u)$ and one in $N_R(u)$), as the ordering distance of the leftmost and the rightmost neighbor of u is at most $2d$.

We have to consider three cases. In all cases, we assume that any u-bad vertex v_{bad} is frozen, otherwise we can recolor it with a v_{bad}-free color. Our goal is to "kill" every u-bad vertex in $N(u)$, i.e. recoloring every u-bad vertex, so that $N(u)$ will not contain any u-bad vertex anymore, and then we can recolor u with c.

Case 1. There is exactly one u-bad vertex, $v_{bad}^R \in N_R(u)$

Consider the set of $q \leq (2d+1)$ vertices, $X = \left\{v_{bad}^R, v_{bad+1}, ..., u_{2d+1}^{i-1}, u_1^i, ..., u_{d+p+1}^i\right\}$ (note that v_{bad}^R can be in $\mathbf{w}_i$). In this set X, there are two vertices that are colored with $(d+p+1)$ (namely, the vertex v_{bad}^R and u_{d+p+1}^i). Take the $|X|-d$ leftmost vertices of X, namely the set $M = \left\{v_{bad}^R, v_{bad+1}, ..., u_{2d+1}^{i-1}, u_1^i ... u_{p+1}^i\right\}$. Note that M can not be empty, since v_{bad}^R is on the left of u_{p+1}^i. Moreover, it should contains at least one unfrozen vertex. In particular, the vertex u_{p+1}^i is absolutely unfrozen, because its left neighbor v_{bad}^R and its right neighbor u_{d+p+1}^i are colored the same. Take an unfrozen vertex v_{uf} in M, and recolor it with a v_{uf}-free color (notice that this color is not c). Since $v_{bad}^R, v_{uf} \in M$ then they are adjacent (because $|M| \leq d+1$), then recoloring v_{uf} would yield v_{bad}^R to be no longer frozen. Furthermore, as we assumed that v_{bad}^R is frozen, then no other vertex in $N(v_{bad}^R)$ is colored with $\alpha(v_{uf})$, the former color of v_{uf}. So we can recolor v_{bad}^R with $\alpha(v_{uf})$, and thus, $N_R(u)$ does not contain any u-bad vertex anymore, and we can recolor u with c. Therefore, to recolor u we need at most 3 steps.

Case 2. There are two u-bad vertices, $v_{bad}^R \in N_R(u)$ and $v_{bad}^L \in N_L(u)$.

Now, as the ordering distance of the leftmost vertex of $N_L(u)$ and the rightmost vertex of $N_R(u)$ is $(2d+1)$ then the subset of $\leq d+1$ vertices $M = \left\{v_{bad+1}^L, v_{bad+2}^L, ..., u_{d+p+1}^{i-1} = u\right\}$ contains at least one unfrozen vertex. In particular, u is unfrozen, because it has two neighbors which are colored the same. As in the previous case, we can take an unfrozen vertex $v_{uf} \in M$, recolor it by a v_{uf}-free color (notice that this color is not $(d+p+1)$). Since $v_{bad}^L, v_{uf} \in M$ and $|M| \leq d+1$ then they are adjacent, so recoloring v_{uf} yields v_{bad}^R to be no longer frozen. Moreover, as we assumed that v_{bad}^L is frozen, then no other vertex in $N(v_{bad}^L)$ is colored with $\alpha(v_{uf})$, the former color of v_{uf}. So we can recolor v_{bad}^L with $\alpha(v_{uf})$, and thus, $N_L(u)$ does not contain any u-bad vertex, and the only bad vertex is $v_{bad}^R \in N_R(u)$. So we are now in the Case 1. Notice that the number of steps we need to kill v_{bad}^L is 2. Hence, to recolor u, we have to do the same way as in the Case 1, so the number of steps needed is at most 5.

Case 3. There is only one u-bad vertex, $v_{bad}^L \in N_L(u)$

Consider the set of $2d+1$ vertices, $\left\{u_{d+p+1}^{i-1}, ..., u_{2d+1}^{i-1}, u_1^i, ..., u_{d+p}^i\right\}$. Based on our assumption, no vertex in this set is colored with c. Hence, we can recolor vertex u_p^i with c, because none of its neighbor is currently colored with $(d+p+1)$. Note that $u_p^i \in N_R(u)$, hence now we are in the Case 2. Notice that we only do at most one step recoloring to be in the Case 2. Therefore, to recolor u, we will need at most 6 steps. As we have to recolor d-suffix of $\mathbf{w}_{i-1}$, so the number of steps is at most $6d$.

□

Remark 3.5. The bound of $k \geq 2d+1$ in Lemma 3.4 is necessary, i.e. the recoloring method given in the Lemma 3.4 does not work for any $k \leq 2d$. We provide a counter-example in Example A.6 of the Appendix. In this case, eventhough there are two vertices in the set $\{u^{i-1}_{d+p+2}, ..., u^{i-1}_{2d+1}, u^i_1, ..., u^i_{d+p+1}\}$ colored the same, it does not guarantee that it contains an unfrozen vertex, because the number of colors k is less than the number of neighbors $|N(v)| = 2d$ that a vertex v may have.

Lemma 3.6. *Let G be a d-path and $k \geq 2d+1$. Any k-coloring of G can be transformed into an almost symmetric coloring in at most $3n$ steps.*

Proof. We will first consider the case for $k = 2d+1$. Let α be any k-coloring of G. To transform α into a coloring in which all blocks have nice coloring, we have to repeat the algorithm in the Lemma 3.4 at most l times where l is the number of blocks in G. Moreover, in order to apply the recoloring method as in Lemma 3.4, we have to first recolor $\mathbf{w}_l$, the rightmost block of G. Here we can add one virtual block $\mathbf{w}_{l+1}$ with some virtual vertices and edges, and assume that the virtual block has a nice coloring. For any recoloring of block $\mathbf{w}_i$, we have to recolor the d-suffix of $\mathbf{w}_i$. Moreover, since G consists of $l = \frac{n}{k}$ blocks, thus we will have to call the algorithm in Lemma 3.4 $\frac{n}{k}$ times. Hence, overall, the total number of steps is at most $\frac{6dn}{k} < 3n$. Note that in this case, when all the blocks of G have nice coloring, it turns that the coloring of G is almost symmetric, because H, the $(d+1)$-prefix of $\mathbf{w}_i$ forms a clique, and $N(H)$ uses d colors of $(d+2), (d+3), ..., (2d+1)$, then every vertex v in the $(k-d)$-prefix of a block would be colored differently.

The case for $k > 2d+1$ is trivial, because $|N(v) \cup \{v\}| = 2d+1$ for every $v in V(G)$, and we have $k > 2d+1$ colors. Therefore, every u-bad vertex is unfrozen, and we can recolor it with a u-free color, which yields that $N(u)$ no longer contains any u-bad vertex. In this case, the number of steps needed to recolor u is at most 3 (at most two steps to recolor the u-bad vertices and one step to recolor u). By the same reasoning as above, the number of steps needed to make all the blocks have nice coloring is at most $\frac{3dn}{k} < \frac{3n}{2}$. However, eventhough all the blocks are colored with a nice coloring does not turn that the coloring becomes almost symmetric, as the number of colors $k > 2d+1$ then the $(k-d)$-prefix of some block may contain vertices that are colored the same. In this case, we can transform the coloring into an almost symmetric coloring by simply changing the color of vertices that are colored the same such that every block uses exactly k colors. The number of steps to do this transformation is at most $\frac{n}{k}(k-d) < n$. Hence, overall we need at most $\frac{3n}{2} + n < 3n$ steps to transform the coloring into almost symmetric.

□

Now we will prove our main theorem, namely Theorem 1.8.

Proof of Theorem 1.8

Proof. To transform the k-coloring γ into the k-coloring δ, we will do a sequence of transformation:

$$\sigma \to \alpha' \to \alpha \to \beta \to \beta' \to \phi$$

where α', β' are almost symmetric k-coloring, and α, β are symmetric k-coloring.

Lemma 3.6 says that we can transform σ into α' (resp. β' into ϕ) in at most $3n$ steps. Then by Lemma 3.1, α' can be transformed into α (resp. β into β') in at most $3n$ steps. Now by Lemma 3.3, α can be transformed into β in at most $k^2 n$ steps. Therefore, overall, the number of steps to perform the recoloring α into β is at most $k^2 n + 2(3n) + 2(3n) = (k^2+12)n = ((2d+1)^2+12)n = (4d^2+4d+13)n$.

□

So far, we assumed that $n = kl$. In case that $n = kl + c$ where $c < k$ is a constant, we can just add some virtual vertices and edges to simplify the recoloring, then we can omit them later.

3.2 Recoloring d-Trees

Now we will try to see whether we can extend this method to a larger class, namely the class of d-trees.

Definition 3.7. A *(complete) d-tree* G is built from a rooted tree (or 1-tree) by adding edges from each vertex to its father, the father of its father, etc. up to d predecessors.

It is obvious that a d-path is an induced subgraph of a d-tree. Notice that we can see a d-tree as a non-disjoint union of d-paths, i.e. for any pair of paths (P_1, P_2), there exist $j \geq 0$ such that $\{v_1, v_2, ..., v_j\}$ are the common vertices of P_1 and P_2. In this case, vertex v_1 is the *root* of the d-tree.. Let the d-tree branches into some paths $P_1, P_2, ..., P_m$. In most of our discussion, we will denote the vertex in which a path makes some branches by u_t^X, and denote the block containing this vertex by $\mathbf{w}_X^{P_i}$ (the superscript P_i indicates that this block is in path P_i and the subscript X denotes the numbering of the block based on the vertex ordering of path P_i. Moreover, sometimes we will put a numbering P_i as a subscript/superscript to indicate that a vertex of a block belongs to the path P_i.

We provide an example of d-tree graph with a $(2d+1)$-coloring in Figure 5 (the dashed lines in every figure are the border of the blocks). In the following section, G always refer to a d-tree Beforehand, the definition of symmetric coloring and almost symmetric coloring still hold on d-trees. In this case, two blocks are considered as different blocks if they contain at least one uncommon vertex. We provide an example of a d-tree with a symmetric and an almost symmetric coloring in Figure 5 and Figure 6 respectively. Some lemmas that are used in the proof of Theorem 1.9 follows.

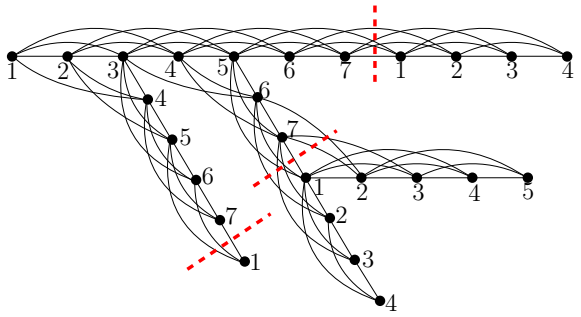


Figure 5: A 3-tree with a symmetric 7-coloring

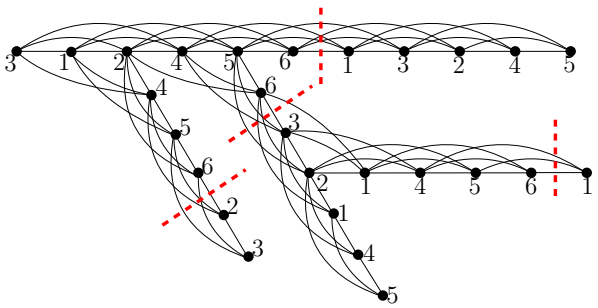


Figure 6: A 3-tree with an almost symmetric 6-coloring

Lemma 3.8. *Let G be a d-tree and $k \geq d+3$. Any two symmetric k-colorings α, β in G can be transformed one into each other in at most $k^2 n$ steps.*

Proof. We adapt the proof of Lemma 3.2. Let G be a d-tree graph that is colored with k-coloring α. Consider a pair of paths $(P-1, P_2)$ that branch at a vertex v_t, in particular let $P_1 = (v_0, ..., v_t, v_{t+1}, ..., v_m)$ and $P_2 = (v_0, ..., v_t, v'_{t+1}, ..., v'_{m'})$ of G. The method is exactly the same as in the Lemma 3.3, except that at some vertices where a path branches. Based

on the definition of the symmetric coloring on d-tree, for any $j \geq 1$, $\alpha(v_{t+j}) = \alpha(v'_{t+j})$ where v_{t+j} and v'_{t+j} are the vertices after v_t in the vertex ordering of P_1 and P_2 respectively. In this case, everytime we recolor the vertex v_{t+j}, then in the next step we recolor the vertex v'_{t+j}. By this algorithm, we will maintain the coloring properties of any path contained in G to be the same as in Lemma 3.2. Moreover, by the similar reason as in Lemma 3.3, thus, any symmetric k-coloring α and β can be transformed one into each other in at most k^2n steps. □

The fact in the proposition and the remark below will be used in the proof of the next lemmas.

Proposition 3.9. *An almost symmetric k-coloring of a d-tree G for $d+3 \leq k \leq 2d+1$ satisfies the following property. Consider two paths P_1 and P_2 of G that intersect in vertex u_t^X of the block $\boldsymbol{w}_X^{P_1}$ of the path P_1 (similarly, the block $\boldsymbol{w}_X^{P_2}$ of the path P_2). Let the common vertices of these two paths in this block are $\{u_1^X, ..., u_t^X\}$. Hence the $(k-t)$-suffix of the block $\boldsymbol{w}_X^{P_1}$ and the block $\boldsymbol{w}_X^{P_2}$ use the same set of colors. Otherwise, there are at least two vertices in $\{u_1^x, ..., u_t^x\}$ that are colored the same.*

Remark 3.10. Consider two consecutive blocks $\mathbf{w}_i, \mathbf{w}_{i+1}$ from a d-path that is colored with an almost symmetric coloring. When applying recoloring within the $(k-d)$-prefix of $\mathbf{w}_{i+1}$ using the set of colors $\{1, 2, ..., k-d\}$, we can skip as many vertices as we need. This is possible because the d-suffix of $\mathbf{w}_i$ and $\mathbf{w}_{i+1}$ are respectively colored with $\{k-d+1, ..., k\}$.

Lemma 3.11. *Let G be a d-tree with an almost symmetric k-coloring of α where $k \geq d+3$. The coloring α can be transformed into a symmetric k-coloring by recoloring every vertex of G at most 5 times.*

Proof. We will adapt the proof of Lemma 3.1. Let G be a d-tree with an almost symmetric k-coloring of α. Recall that up to the color permutation, and without loss of generality, we can assume that the pattern of $(k-d)$-suffix of every block in G is $k-d+1, ..., k$. First, we will transform the coloring α of P into an intermediate k-coloring γ. The following method explains the step-by-step transformation to transform α into γ. The method is exactly the same as in Algorithm 1, except at some vertices in which a path has branches.

1. Apply Algorithm 1 on every path P in G. We do it consecutively from the leaves of the tree until finally we reach the root of the tree. Everytime we have branches, we do the following.

   (i) Consider a set of paths $\mathbf{P} = \{P_1, P_2, ..., P_m\}$ that intersect on a vertex u_t^X of the block $\mathbf{w}_X^{P_i}$ of the path P_i for $i \in \{1, 2, ..., m\}$. Apply Algorithm 1 to every path of $P_i \in \mathbf{P}$ up to the block $\mathbf{w}_{X+1}^{P_i}$.

   (ii) Now we will recolor the block $\mathbf{w}_X^{P_i}$. Let S be the set of common vertices of the block $\mathbf{w}_X^{P_i}$ for every $P_i \in \mathbf{P}$. Note that when we want to recolor a vertex $v \in S$ with c, the color c has to be a v-free in every path $P_i \in \mathbf{P}$ (indeed, Proposition 3.9 ensures that this is always possible, since the set of colors of the $(k-t)$-suffix of block $\mathbf{w}_X^{P_i}$) are the same for every $P_i \in \mathbf{P}$. Let v_S be the "first" such vertex in S. On the path $P_1 \in \mathbf{P}$, continue Algorithm 1 until we have an appropriate v_S-free color c that we will use to recolor v_S.

   (iii) On every path $P_i \in \mathbf{P}$ for $i \in \{2, 3, ..., m\}$, apply Algorithm 1 up to the vertex u_{P_i} that is colored with c, and skip the remaining vertices on the $(k-t)$-suffix of $\mathbf{w}_X^{P_i}$. Note that this is always possible based on Remark 3.10. As soon as we have that color c that is v_S-free in every path $P_i \in \mathbf{P}$, then recolor v_S with c and continue Algorithm 1 until we finish the recoloring of the block $\mathbf{w}_X^{P_i}$.

2. Repeat steps (ii) and (iii) above everytime we have branches, until we finish the recoloring of G. In the end, G will be colored with γ.

Notice that the method above requires the recoloring of every vertex of G at most once. Now, we will explain how to transform γ into a symmetric coloring β.

1. Starting from the root of the tree, apply Algorithm 2, by first transforming the leftmost block of the tree (which is also the leftmost block of every path in the tree) such that the pattern becomes $(1, 2, ..., k)$. We do it consecutively to the right as in Algorithm 2.

2. Similar to the previous method, the exception is when we have branches in the path. Everytime we have branches, we do the following. Assume that we will recolor the block $\mathbf{w}_X^{P_i}$ in which the path branches at vertex u_t^X into some set of paths $\mathbf{P} = \{P_1, P_2, ..., P_m\}$. We consider two cases.

   (i) If the pattern of $(k-t)$-suffix of the block $\mathbf{w}_X^{P_i}$ is not the same in all the paths of $\mathbf{P}$ (this might be the case when $t \in \{1, ..., (k-d-3)\}$) then first, we have to make the pattern to be the same in every $P_i \in \mathbf{P}$. Otherwise, go to step (ii). To do so, apply clique-recoloring to the $(k-t)$-suffix of every path $P_i \in \mathbf{P}$. In fact, we can obey the d-suffix of every $P_i \in \mathbf{P}$ as they already have the same pattern, and just implement the clique-recoloring on sub-block $W = (u_{t+1}^X, u_{t+2}^X, ..., u_{k-d}^X)$ in every path $P_i \in \mathbf{P}$.

   (ii) Now the pattern of $\mathbf{w}_X^{P_i}$ is the same for every $P_i \in \mathbf{P}$. Apply Algorithm 2 on $\mathbf{w}_X^{P_i}$ for every $P_i \in \mathbf{P}$. In this case, everytime we have to recolor a vertex of $\mathbf{w}_X^{P_i}$, then we do it consecutively for every P_i.

   (iii) As soon as the pattern of the block $\mathbf{w}_X^{P_i}$ is $(1, 2, ..., k)$, for every $P_i \in \mathbf{P}$, continue the recoloring as in Algorithm 2 to the right.

3. Repeat step 2 above everytime we have a branches until we finish the recoloring of G. In the end, G will be colored with β.

By this method, we can transform γ into β by recoloring every vertex at most four times. Hence, to transform α into β every vertex has to be recolored at most 5 times, i.e. the number of steps needed is at most $5n$. An example to illustrate this recoloring is provided in Example A.7 and Example A.8.

□

Lemma 3.12. *Let G be a d-tree and $k \geq 2d+1$. Consider two consecutive k-blocks $\boldsymbol{w}_{i-1}$ and $\boldsymbol{w}_i$ in G, where $\boldsymbol{w}_i$ has a nice coloring. We can recolor those blocks in such a way that both of them will have a nice coloring, by only recoloring the vertices of $\boldsymbol{w}_{i-1}$ and $(k-d)$-prefix of $\boldsymbol{w}_i$.*

Proof. The recoloring is exactly the same as in d-paths except at some vertices in which the paths of G have branches. Let $\mathbf{P} = \{P_1, P_2, ..., P_m\}$ be the set of paths that branch at vertex u_t^X of the block $\mathbf{w}_X^{P_i}$ for $P_i \in \mathbf{P}$. Assume that the blocks $\mathbf{w}_{X+1}^{P_i}$ on the right of $\mathbf{w}_X^{P_i}$ in path P_i has nice coloring. Our goal is to recolor $\mathbf{w}_X^{P_i}$ and $(k-d)$-prefix of $\mathbf{w}_X^{P_i}$ in such a way that $\mathbf{w}_X^{P_i}$ has a nice coloring. Assume now that the $(p-1)$ leftmost vertices of the d-suffix of $\mathbf{w}_X^{P_i}$ have been colored with the target coloring, we plan to recolor the vertex $u = u_{d+p+1}^X \in \mathbf{w}_X^{P_i}$ with $c = d + p + 1$.

Let S be the set of common vertices of every path $P_i \in \mathbf{P}$ in the block $\mathbf{w}_X^{P_i}$. We assume that we can not recolor u trivially because $N(u)$ contains at least one u-bad vertex, and these u-bad vertices are frozen at some paths. Otherwise, it can be recolored trivially so that it is no longer u-bad. Let us denote v_{bad}^L and v_{bad}^R the u-bad vertices that belong to $N_L(u)$ and $N_R(u)$ respectively. Moreover, we also assume that u, v_{bad}^R (resp. v_{bad}^L), and the unfrozen vertex v_{uf} are not together in S or $V(\mathbf{w}_X^{P_i}) \backslash S$. Otherwise, we can consider it as a case of d-paths. To simpifly our discussion, we will only consider two branches, namely P_h and P_j. In the following section, we analyse several cases that we could have in a pair of branches $P_h, P_j \in \mathbf{P}$ of a d-tree, and how to recolor the v_{bad}^L and v_{bad}^R.

Case 1 (see Figure 7). The block only contains $v^R_{bad} \in N_R(u)$.

This case is similar to Case 1 of Lemma 3.4. We remind you that as in d-paths, there exists at least one unfrozen vertex $v_{uf} \in N_R(v^R_{bad})$. Note that here we assume that $u \in S$, otherwise we can consider it as the case of Lemma 3.4 as $u, v^R_{bad}, v_{uf} \in V(\mathbf{w}^{P_h}_X) \backslash S$.

(i) *Case 1.1.* $u \in S$ and $v^R_{bad} \notin S$ (see Figure 7 (a)).

As $v^R_{bad} \notin S$, then each branch contains one v^R_{bad}, namely $v^R_{bad_h}$ in $\mathbf{w}^{P_h}_X$ and $v^R_{bad_j}$ in $\mathbf{w}^{P_j}_X$. Moreover, the unfrozen vertex $v_{uf} \notin S$, i.e. there exists v_{uf_h} in $\mathbf{w}^{P_h}_X$ and v_{uf_j} in $\mathbf{w}^{P_j}_X$. Hence we can recolor v_{uf_h} and v_{uf_j} with a v_{uf_h}-free and v_{uf_j}-free color respectively, then recolor v_{bad_h} and v_{bad_j} with $\alpha(v_{uf_h})$ and $\alpha(v_{uf_j})$, respectively.

(ii) *Case 1.2.* $u \in S$, $v^R_{bad} \in S$ (see Figure 7 (b)).

This case is a bit more complicated. Assume that the unfrozen vertex $v_{uf} \notin S$, otherwise we can consider it as the coloring of d-path as u, v^R_{bad}, and v_{uf} are in S. Hence, there exists an unfrozen vertex v_{uf_h} in $\mathbf{w}^{P_h}_X$ and v_{uf_j} in $\mathbf{w}^{P_j}_X$. First we look at the block $\mathbf{w}^{P_h}_X$. Since vertex v_{uf_h} is unfrozen in $\mathbf{w}^{P_h}_X$, thus we can recolor it with a v_{uf_h}-free color. Hence, $f = \alpha(v_{uf_h})$ is now a v^R_{bad}-free color in $\mathbf{w}^{P_h}_X$.

Now look at the block $\mathbf{w}^{P_j}_X$. If no vertex of $N_R(v^R_{bad})$ in this block is colored with f, it means that f is a v^R_{bad}-free color in $\mathbf{w}^{P_j}_X$. Otherwise, if f is not a v^R_{bad}-free color, i.e. it is used by a vertex $y \in N_R(v^R_{bad})$, then:

- If y is unfrozen, recolor it with a y-free color so that f becomes v^R_{bad}-free.
- Otherwise if y is frozen, then recolor v_{uf_j} with a v_{uf_j}-free color, then recolor y with $\alpha(v_{uf_j})$, so that f will be v^R_{bad}-free.

Now f is v^R_{bad}-free in both $\mathbf{w}^{P_h}_X$ and $\mathbf{w}^{P_j}_X$. Hence we can recolor v^R_{bad} with f.

Case 2 (see Figure 8). The block contains both $v^L_{bad} \in N_L(u)$ and $v^R_{bad} \in N_R(u)$.

This case is similar to Case 2 of Lemma 3.4. Our goal is to recolor v^L_{bad} so that we can be in the Case 1. We remind you that as in d-paths, u is unfrozen. Note that here we assume that $v^R_{bad} \notin S$, otherwise we can consider it as the case of Lemma 3.4 as $v^L_{bad}, u, v^R_{bad} \in S$.

(i) *Case 2.1.* $v^L_{bad} \in S$, $u \in S$, and $v^R_{bad} \notin S$ (see Figure 8 (a)).

Note that as $u \in S$, then $v_{uf} \in S$ (which implies $v^L_{bad}, v_{uf}, u \in S$). So the case becomes trivial as we can consider the recoloring of v^L_{bad} as in a d-path.

(ii) *Case 2.2.* $v^L_{bad} \in S$, $u \notin S$, and $v^R_{bad} \notin S$ (see Figure 8 (b)).

As $u \notin S$ then $\exists u_{P_h} \in \mathbf{w}^{P_h}_X$ and $\exists u_{P_j} \in \mathbf{w}^{P_j}_X$. First we look at the block $\mathbf{w}^{P_h}_X$. Vertex u_{P_h} is unfrozen in P_h, thus we can recolor it with a u_{P_h}-free color. Hence, $f = \alpha(u_{P_h})$ is a v^L_{bad}-free color in P_h. Now look at the block $\mathbf{w}^{P_j}_X$. If no vertex of $N_R(v^L_{bad})$ in P_j is colored with f, it means that f is also v^L_{bad}-free color in P_j. Otherwise, if f is not a v^L_{bad}-free color, i.e. it is used by a vertex $y \in N_R(v^R_{bad})$, then:

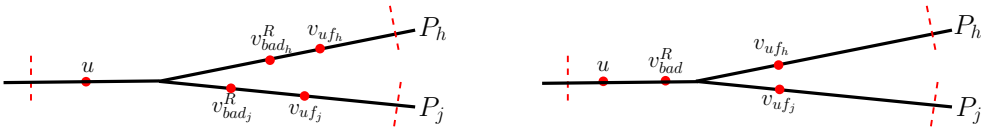
- If y is unfrozen, recolor it with a y-free color, so that f becomes v^R_{bad}-free.
- Otherwise if y is frozen, then recolor u_{P_j} with a u_{P_j}-free color, then recolor y with $\alpha(u_{P_j})$, so that f will be also v^L_{bad}-free color in P_j.

Now f is v^R_{bad}-free in both $\mathbf{w}^{P_h}_X$ and $\mathbf{w}^{P_j}_X$, so then, we can recolor v^L_{bad} with f.

Hence we can consider the current condition as in *Case 1* above.

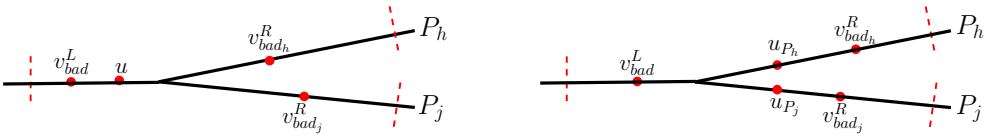
Case 3 (see Figure 9). There exists a branch $\mathbf{w}^{P_h}_X$ that only contains $v^L_{bad} \in N_L(u)$, and the other branches contain both $v^L_{bad} \in N_L(u)$ and $v^R_{bad} \in N_R(u)$).

We will prove that we can make the block $\mathbf{w}^{P_h}_X$ contains both $v^L_{bad} \in N_L(u)$ and $v^R_{bad} \in N_R(u)$). If we can do so, then we can consider the case as in *Case 2*. Look at the path P_j, there are two cases to consider. So our goal is to recolor a vertex $v \in N_R(u)$ with c.



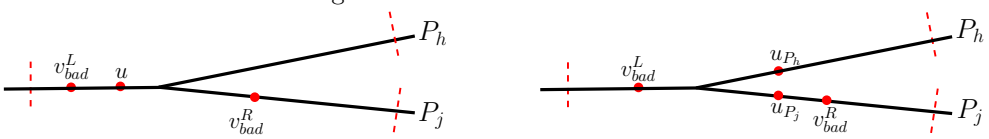
(a) Case 1.1 (b) Case 1.2

Figure 7: Case 1 of Lemma 3.12



(a) Case 2.1 (b) Case 2.2

Figure 8: Case 2 of Lemma 3.12



(a) Case 3.1 (b) Case 3.2

Figure 9: Case 3 of Lemma 3.12

(i) *Case 3.1.* $v^L_{bad} \in S$, $u \in S$ (see Figure 9 (a)).
This case is similar to Case 3 of Lemma 3.4 (we suggest you to look again at Lemma 3.4). In this case, there exists a vertex $v \in N_R(u)$ s.t. no vertex in $N(v) \cup \{v\}$ is colored with c. Hence we can recolor v with c.

(ii) *Case 3.1.* $v^L_{bad} \in S$, $u \notin S$ (see Figure 9 (b)).
(similar to Case 3.1)

□

Remark 3.13. Note that the case when the block $\mathbf{w}_X^{P_h}$ only contains $v^R_{bad} \in N_R(u)$, and the block $\mathbf{w}_X^{P_j}$ contains both $v^L_{bad} \in N_L(u)$ and $v^R_{bad} \in N_R(u)$ can not be happen, since if the block $\mathbf{w}_X^{P_h}$ contains v^L_{bad} then so do every path in P. Otherwise, $v^L_{bad} \notin S$, that yields that $v^L_{bad}, u, v^R_{bad}, u \in V(\mathbf{w}_X^{P_i}) \backslash S$, and the problem becomes trivial.

Lemma 3.14. *Let G be a d-tree that is colored with a k-coloring α with $k = 2d + 1$. Then we can transform α into an almost symmetric k-coloring β, in at most $6dn$ steps.*

Proof. Note that in order to transform α into β, the coloring of every block of G must be a nice coloring. In each step, if a block does not have any branch, then it requires at most $6d$ steps as in Lemma 3.4. If the block has some branches (let us say the block $\mathbf{w}_X^{P_i}$, then we consider Case 1, 2, 3 of Lemma 3.12. In order to recolor a vertex of the d-suffix of the block, Case 1 of Lemma 3.12 requires at most 3 steps. Case 2 requires 2 more steps to be in Case 1, so it needs at most 5 steps. Meanwhile, Case 3 requires one more step to be in Case 2, so it needs at most 6 steps to recolor one vertex of the d-suffix of $\mathbf{w}_X^{P_i}$. As we have to recolor d-suffix of the block, hence to transform the coloring of the block $\mathbf{w}_X^{P_i}$ into a nice coloring, we need at most $6d$ steps (the same result as in d-paths).

Note that the number of blocks in a d-tree can not be greater than the number of vertices n. Hence, to transform an arbitrary k-coloring of a d-tree into an almost symmetric coloring, we need at most $6dn$ steps.

□

Now we are ready to prove our main theorem, namely Theorem 1.9.

Proof of Theorem 1.9

Proof. By the same reasoning as in Theorem 1.8, we can do a sequence of transformation

$$\sigma \to \alpha' \to \alpha \to \beta \to \beta' \to \phi$$

where α', β' are almost symmetric k-coloring, and α, β are symmetric k-coloring. Now by Lemma 3.14, to transform σ into α' (resp. β' into ϕ) we need at most $6dn$ steps. Then by Lemma 3.11, α' can be transformed into α (resp. β into β') in at most $5n$ steps. Now by Lemma 3.8, α can be transformed into β in at most k^2n steps. Therefore, overall, the number of steps to perform the recoloring α into β is at most $k^2n+2(6dn)+2(5n) = (k^2+12d+10)n = ((2d+1)^2+12d+10)n = (4d^2+16d+11)n$.

□

4 Conclusion and Further Works

We have seen that given a set of k colors with $k \geq 2d+1$, we have a step-by-step transformation to transform any two k-colorings of a d-path (or d-trees in general) within a linear (in n) number of steps. This result improves the lower bound on k given by Bousquet and Perarnau [4]. However, while the transformation between *special colorings* (symmetric/almost symmetric k-colorings) of a d-path have a good lower bound, namely $k \geq d+3$, the lower bound on k of transforming an arbitrary k-coloring into an almost symmetric k-coloring is not good enough, since we have $k \geq 2d+1$, which is lower only by one compared to the result of Bousquet and Perarnau [4].

Hence finding a better lower bound on the number of colors k to transform an arbitrary k-coloring into an almost symmetric k-coloring on a d-path while preserving the number of steps to be linear in n, is one goal of further research. Furthermore, as our discussion in this paper is restricted to a small class of d-degenerate graphs, we are also interested in analysing whether the methods that we use in the recoloring of d-paths can be extended into larger classes of d-degerate graphs.

References

[1] L. Cereceda. *Mixing Graph Colourings*. PhD thesis, London School of Economics and Political Science, 2007.

[2] M. Bonamy, M. Johnson, I. Lignois, V. Patel, and D. Paulusma. *Reconfiguration Graphs for Vertex Colourings of Chordal and Chordal Bipartite Graphs*. Journal of Combinatorial Optimization, pages 1-12, 2012.

[3] M. Bonamy and N. Bousquet. *Recoloring Graphs via Tree Decompositions*. 2016.

[4] N. Bousquet and G. Perarnau. *Fast Recoloring of Sparse Graphs*. European Journal of Combinatorics, 52:1-11, 2016.

[5] M. Johnson, D. Kratsch, S. Kratsch, V. Patel, and D. Paulusma. *Finding Shortest Paths between Graph Colourings*. CoRR, 2014. arXiv:1403.6347.

[6] P. Bonsma and L. Cereceda. *Finding Paths between Graph Colourings: PSPACE-Completeness and Superpolynomial Distances*. Theoretical Computer Science, 410(50):5215–5226, 2009.

A Appendix

A.1 Basic Definitions and Concepts in Graph Theory

A graph $G(V, E)$ is a set V of *vertices* and a set E of *edges*. In an *undirected* graph, an edge is an unordered pair of vertices, and an edge connecting vertex u and v is denoted by uv. An ordered pair of vertices is called a directed edge (arc). A graph with directed edges is called *directed* graph. If we allow multi-sets of edges, i.e. multiple edges between two vertices, we obtain a *multigraph*. A self-loop or loop is an edge between a vertex and itself. An undirected graph without loops or multiple edges is known as a *simple* graph. In this paper, we assume graphs to be simple and undirected.

If vertices u and v are endpoints of an edge, we say that they are *adjacent*. If vertex v is one of edge e's endpoints, v is incident to e. The *degree* of a vertex is the number of edges incident to it. A *walk* is a sequence of vertices $v_1, v_2, ..., v_t$ such that $(v_i, v_{i+1}) \in E$. A *path* is a walk where $v_i \neq v_j \ \forall i, j$. In other words, a path is a walk that visits each vertex at most once. A closed walk is a walk where $v_1 = v_t$. A *cycle* is a closed path, i.e. a path combined with the edge (v_t, v_1). Similarly, in a directed graph D, we use the terms of *directed path* and *directed cycle*. Consider a vertex u of a D, the number edges going in u is called the *in-degree* of u and the number of edges going out of u is its *out-degree*. Most of the times in this report, we will discuss an undirected graph. Thus, G always refer to undirected graph.

A graph is connected if for each pair of vertices, there exists a path connecting them. Consider a pair of vertices (u, v) in G. The length of a path connecting u, v is the number of edges traversed by the path. A path connecting u, v having the minimum length is called *shortest path between u and v*, and the length is denoted by $d(u, v)$ and is called the *distance* between u and v. The *diameter* of G is the maximum length of a shortest path between any pair of vertices in G. A disconnected graph has infinite diameter. A *tree* is a connected graph with no cycles. An *internal* vertex (or inner vertex) is a vertex of degree at least 2. A vertex with degree 1 is called a *leaf*. A *forest* is a graph where each connected component is a tree, in other words, the graph consists of a disjoint union of trees. A *rooted tree* is a tree in which one vertex has been designated the *root*.

1

A.2 Some Examples

Example A.1. Consider two consecutive blocks $\mathbf{w}_{i-1}$ and $\mathbf{w}_i$ of a 2-path G colored with a 5-coloring α. We provide some example in Table 1 and 2 as you can find below. Every row shows the step-by-step transformation of the recoloring, where the circled number denotes the color that is currently changed. Table 1 is an example of a right-chain-recoloring applied on those blocks, where it is initially colored with $(2, 3, 1, 4, 5 \mid\mid 3, 1, 2, 4, 5)$. The coloring starts from vertex u_1^{i-1} and stops at vertex u_3^i. Meanwhile Table 2 is an example of a right-chain-recoloring applied on those blocks by skipping some vertices. Note that the blocks are initially colored with $(2, 3, 1, 4, 5 \mid\mid 2, 3, 1, 4, 5)$ and we can skip at most one vertex in each step because $d(\alpha) = 5 = d + 3$. In this example, we skip the vertex u_3^{i-1} and u_5^{i-1}.

Example A.2. Table 3 is an example of clique-recoloring applied on a block $\mathbf{w}_i$ of a 5-path that is initially colored with an intermediate 9-coloring. This table shows a step-by-step recoloring based on Algorithm 2.

Example A.3. Table 4 is an example of recoloring as in Lemma 3.2. Consider a 3-path on 12 vertices having 6-symmetric colorings α_1, α_2 that differ by one transposition. Assume that the block patterns of α_1 and α_2 are $(2, 1, 6, 5, 3, 4)$ and $(2, 1, 6, 3, 5, 4)$ respectively. The step by step transformation from α_1 into α_2 is shown in Table 4.

Example A.4. Figure 10 is an example of the implementation of Algorithm 1 applied on a 5-path of 27 vertices, and $k = 9$. Figure 10(a) shows the initial coloring α (which is almost symmetric) of the path, and Figure 10(b) is the intermediate coloring γ obtained after applying Algorithm 1. The underlined number indicates that we skip the coloring of the

Vertex	u_1^{i-1}	u_2^{i-1}	u_3^{i-1}	u_4^{i-1}	u_5^{i-1}	u_1^i	u_2^i	u_3^i	u_4^i	u_5^i
Step 0	2	3	1	4	5	3	1	2	4	5
Step 1	⑤	3	1	4	5	3	1	2	4	5
Step 2	5	②	1	4	5	3	1	2	4	5
Step 3	5	2	③	4	5	3	1	2	4	5
Step 4	5	2	3	①	5	3	1	2	4	5
Step 5	5	2	3	1	④	3	1	2	4	5
Step 6	5	2	3	1	4	⑤	1	2	4	5
Step 7	5	2	3	1	4	5	③	2	4	5
Step 8	5	2	3	1	4	5	3	①	4	5

Table 1: Right-chain-recoloring up to vertex u_3^i on 2-path

Vertex	u_1^{i-1}	u_2^{i-1}	u_3^{i-1}	u_4^{i-1}	u_5^{i-1}	u_1^i	u_2^i	u_3^i	u_4^i	u_5^i
Step 0	2	3	1	4	5	2	3	1	4	5
Step 1	⑤	3	1	4	5	2	3	1	4	5
Step 2	5	②	1	4	5	2	3	1	4	5
Step 3	5	2	1	③	5	2	3	1	4	5
Step 4	5	2	1	3	5	④	3	1	4	5
Step 5	5	2	1	3	5	4	②	1	4	5
Step 6	5	2	1	3	5	4	2	③	4	5

Table 2: Right-chain-recoloring up to vertex u_3^i on 2-path by skipping some vertices

Vertex	u_1^i	u_2^i	u_3^i	u_4^i	u_5^i	u_6^i	u_7^i	u_8^i	u_9^i
Step 0	4	3	2	5	6	7	8	9	3
Step 1	4	①	2	5	6	7	8	9	3
Step 2	4	1	③	5	6	7	8	9	3
Step 3	4	②	3	5	6	7	8	9	3
Step 4	①	2	3	5	6	7	8	9	3
Step 5	1	2	3	④	6	7	8	9	3
Step 6	1	2	3	4	⑤	7	8	9	3
...	...	...	...	...	...	...	...	...	...
Step 10	1	2	3	4	5	6	7	8	⑨

Table 3: Clique-recoloring on block $\mathbf{w}_i$ of a 5-path with $k = 9$

Vertex	u_1^1	u_2^1	u_3^1	u_4^1	u_5^1	u_6^1	u_1^2	u_2^2	u_3^2	u_4^2	u_5^2	u_6^2
Step 0	2	1	6	5	3	4	2	1	6	5	3	4
Step 1	④	1	6	5	3	4	2	1	6	5	3	4
Step 2	4	②	6	5	3	4	2	1	6	5	3	4
Step 3	4	2	①	5	3	4	2	1	6	5	3	4
...	...	...	...	...	...	...	...	...	...	...	...	...
Step 9	4	2	1	6	5	3	4	2	1	6	⑤	4
Step 10	4	2	1	6	5	3	4	2	1	③	5	4
Step 11	4	2	1	6	5	3	4	2	⑥	3	5	4
...	...	...	...	...	...	...	...	...	...	...	...	...
Step 19	②	1	6	3	5	4	2	1	6	3	5	4

Table 4: Step by step transformation of two symmetric 6-colorings α_1, α_2 that differ by one transposition on a 3-path

vertex. Figure 11 shows the implementation of Algorithm 2 to the intermediate coloring γ. Figure 11(a) is the coloring γ. Figure 11(b), (c), (d) show the coloring obtained after each iteration of the loop of the clique-recoloring, consecutively from the left (see Table 3 for more details about clique-recoloring)

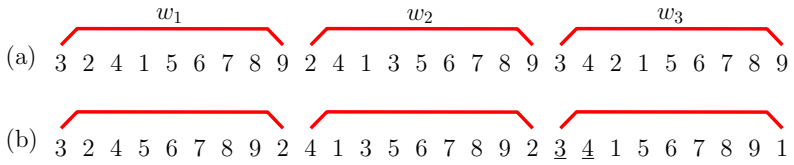


Figure 10: Transforming an almost symmetric 9-coloring into an intermediate 9-coloring on a 5-path

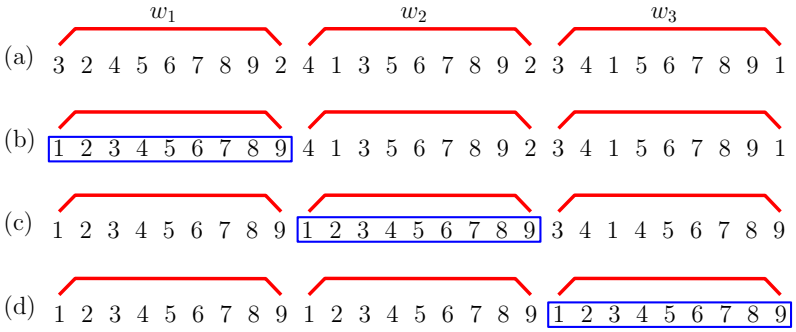


Figure 11: Transforming an intermediate 9-coloring into a symmetric 9-coloring on a 5-path

Example A.5. Consider 3-path on 12 vertices having 6-symmetric coloring α, and β with block pattern $(2,1,6,5,3,4)$ and $(6,2,1,4,3,5)$ respectively. Hence we can transform α into β by a sequence of transformation of symmetric colorings:

$$\alpha = \alpha_0 \to \alpha_1 \to \alpha_2 \to \alpha_3 \to \alpha_4 \to \alpha_5 = \beta$$

where: $\alpha = \alpha_0 = (2,1,6,5,3,4)$; $\alpha_1 = (2,6,1,5,3,4)$; $\alpha_2 = (6,2,1,5,3,4)$; $\alpha_3 = (6,2,1,5,4,3)$; $\alpha_4 = (6,2,1,4,5,3)$; $\alpha_5 = (6,2,1,4,3,5) = \beta$. In this example, each transformation needs at most $2n$ steps (by using the method as in Example A.3). Hence to transform α into β, we need at most $5(2n) = 10n < 36n = k^2n$.

Example A.6. In Figure 12, the block $\mathbf{w}_{i-1}, \mathbf{w}_i$ are parts of the 4-path G, and the number of colors is 7. Assume that vertex u_4^{i-1} of $\mathbf{w}_{i-1}$ has been colored with its targer color 4, and our goal is to recolor u_5^{i-1} with 5. This is a counter-example of Case 1 of Lemma 3.4. As we can see in the figure, the vertices between u_7^{i-1} and u_5^i (which are colored with 5), namely u_1^i, u_2^i, and u_3^i of $\mathbf{w}_i$ are frozen, and we do not want to recolor any vertex of the d-suffix of $\mathbf{w}_i$. Hence, the argument of Case 1 does not hold in this case.

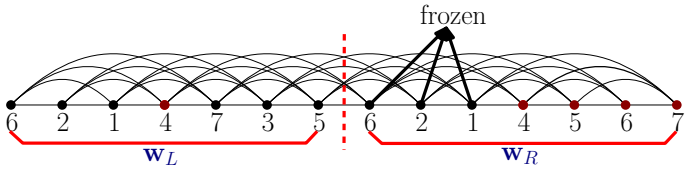


Figure 12: Counter example of Remark 3.5

Example A.7. Let we have a 3-tree with branches at a vertex u_2^X of the block $\mathbf{w}_X$ as shown in Figure 13. The tree is initially colored with an almost symmetric 7-coloring and we aim to

transform it into a symmetric 7-coloring. In Figure 13 we give a step-by-step transformation of transforming the almost symmetric 7-coloring into an intermediate 7-coloring. Meanwhile in Figure 14 we give a step-by-step transformation of transforming the intermediate 7-coloring into a symmetric 7-coloring. In each figure, the numbers in red indicate the corresponding vertices that are recolored in that step, ant the encircled numbers indicate that we skip the corresponding vertices during the recoloring.

Step-by-step transformation of Figure 13

(a) G is initially colored with an almost symmetric 7-coloring. Remember that in applying Algorithm 1, our restriction is that for every block $\mathbf{w}_i$, we can not recolor vertex u_k^i with $k-d$ (in this example, we can not recolor u_7^i with 4). Moreover, if (u_{k-d}^{i-1}, u_1^i) is a disturbing pair, then we have to skip the vertex u_1^i.

(b) The coloring of G after applying Algorithm 1 up to the block $\mathbf{w}_{X+1}$ of both paths. We skip vertex u_1^{X+1} in path P_1 because u_7^X can not be recolored with 4 in the next step.

(c) Now we will recolor the block $\mathbf{w}_X$. Apply Algorithm 1 on the block $\mathbf{w}_X$, remember that when recoloring vertex u_2^X, the color that we will use must be u_2^X-free in both paths. Hence, "free" the color 3 in $\mathbf{w}_X$ of P_1. As 3 must be u_2^X-free in $\mathbf{w}_X$ of P_2, so skip the recoloring of u_3^X of P_2. By this way, 3 is u_2^X-free in P_2, and we can recolor u_2^X with 3.

(d) Apply Algorithm 1 on the block $\mathbf{w}_{X-1}$ up to vertex u_4^{X-1}. Now G is colored with an intermediate 7-coloring.

Step-by-step transformation of Figure 14

(a) Now we will transform the coloring into a symmetric coloring. The figure shows the coloring of G after applying Algorithm 2 to the block $\mathbf{w}_{X-1}$. Now the pattern of $\mathbf{w}_{X-1}$ becomes $(1, 2, 3, 4, 5, 6, 7)$.

(b) Now we will recolor the branches, namely block $\mathbf{w}_X$. As the vertices of 5-suffix of $\mathbf{w}_X$ agree in both paths, then we apply directly the clique-recoloring on 3-prefix of the block, continued by right chain (as in Algorithm 2

(c) The coloring of G after applying Algorithm 2 to the block $\mathbf{w}_{X+1}$ of both paths. Now G is colored with a symmetric 7-coloring with block pattern $(1, 2, 3, 4, 5, 6, 7)$.

Example A.8. We provide another example of recoloring a branch of a 5-tree G, as shown in Figure 15. Let the tree branches at vertex u_2^X of the block $\mathbf{w}_X$, and it is initially colored with an almost symmetric coloring α as shown in Figure 15(a). We aim to transform the coloring into symmetric coloring. e will first transform into an intermediate 9-coloring, followed by the transformation into a symmetric 9-coloring. The steps of transformation is described below. The readers can refer to Figure 15. In each figure, the numbers in red indicate the corresponding vertices that are recolored in that step.

(a) G is initially colored with an almost symmetric 9-coloring. Our restriction is that for every block $\mathbf{w}_i$, we can not recolor u_9^i with 4, and if (u_{k-d}^{i-1}, u_1^i) is a disturbing pair, then we have to skip the vertex u_1^i.

(b) The coloring of G after applying Algorithm 1 up to the block $\mathbf{w}_{X+1}$ of both paths. We skip vertex u_1^{X+1} because (u_4^X, u_1^{X+1}) is a disturbing pair.

(c) Now we will recolor the block $\mathbf{w}_X$, by applying Algorithm 1. Note that we have to skip vertex u_2^X, because (u_4^{X-1}, u_1^X) is a disturbing pair and u_2^X is curently colored with 4. Moreover, note that when recoloring vertex u_1^X, the color that we will use must be u_1^X-free in both paths. Hence, "free" the color 3 in $\mathbf{w}_X$ of P_1. As 3 must be u_1^X-free in $\mathbf{w}_X$ of P_2, so skip the recoloring of u_3^X of P_2. By this way, 3 is also u_1^X-free in P_2, and we can recolor u_1^X with 3.

(d) The coloring of G after applying Algorithm 1 to the block $\mathbf{w}_{X-1}$ up to vertex u_4^{X-1}. Now G is colored with an intermediate 9-coloring.

(e) Now we will transform the coloring into a symmetric coloring. The figure shows the coloring of G after applying Algorithm 2 to the block $\mathbf{w}_{X-1}$. Now the pattern of $\mathbf{w}_{X-1}$ becomes $(1, 2, 3, 4, 5, 6, 7, 8, 9)$.

(f) Now we will recolor the branches, namely block $\mathbf{w}_X$. We first make the vertices of 7-suffix of $\mathbf{w}_X$ agree in both paths. The figure shows the coloring of G after applying the clique-recoloring on the vertices $\left\{u_3^X, u_4^X\right\}$.

(g) The coloring of G after applying Algorithm 2 to the block $\mathbf{w}_X$. Now the pattern of $\mathbf{w}_X$ becomes $(1, 2, 3, 4, 5, 6, 7, 8, 9)$ in both paths.

(h) The coloring of G after applying Algorithm 2 to the block $\mathbf{w}_{X+1}$ of P_1. Now G is colored with a symmetric 9-coloring with block pattern $(1, 2, 3, 4, 5, 6, 7, 8, 9)$.

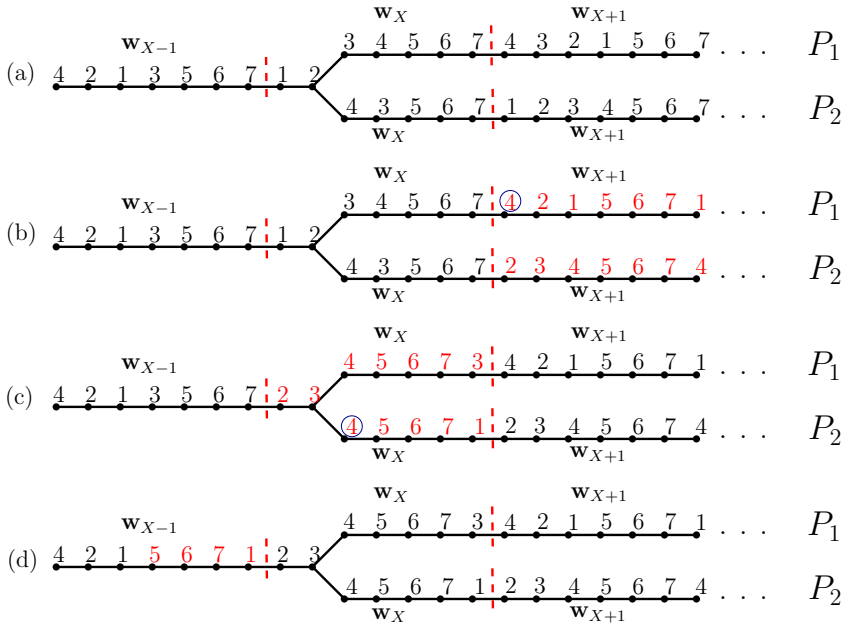


Figure 13: Step-by-step transformation of an almost symmetric 7-coloring into an intermediate 7-coloring on a 3-tree

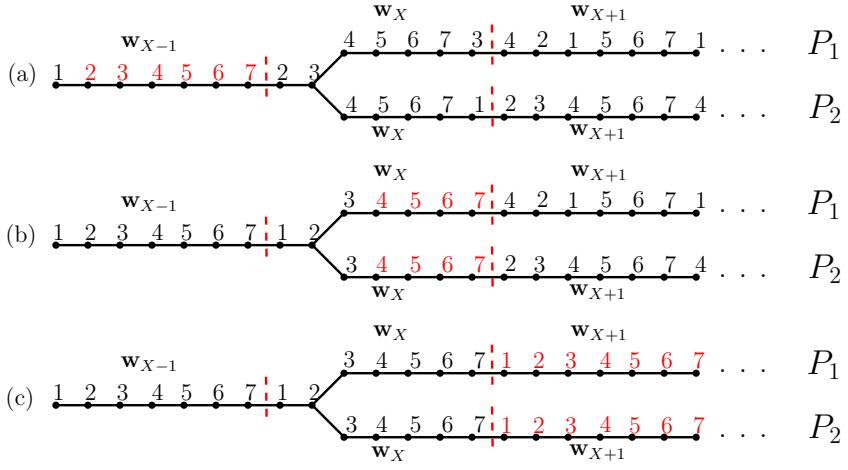


Figure 14: Step-by-step transformation of an intermediate 7-coloring into a symmetric 7-coloring on a 3-tree

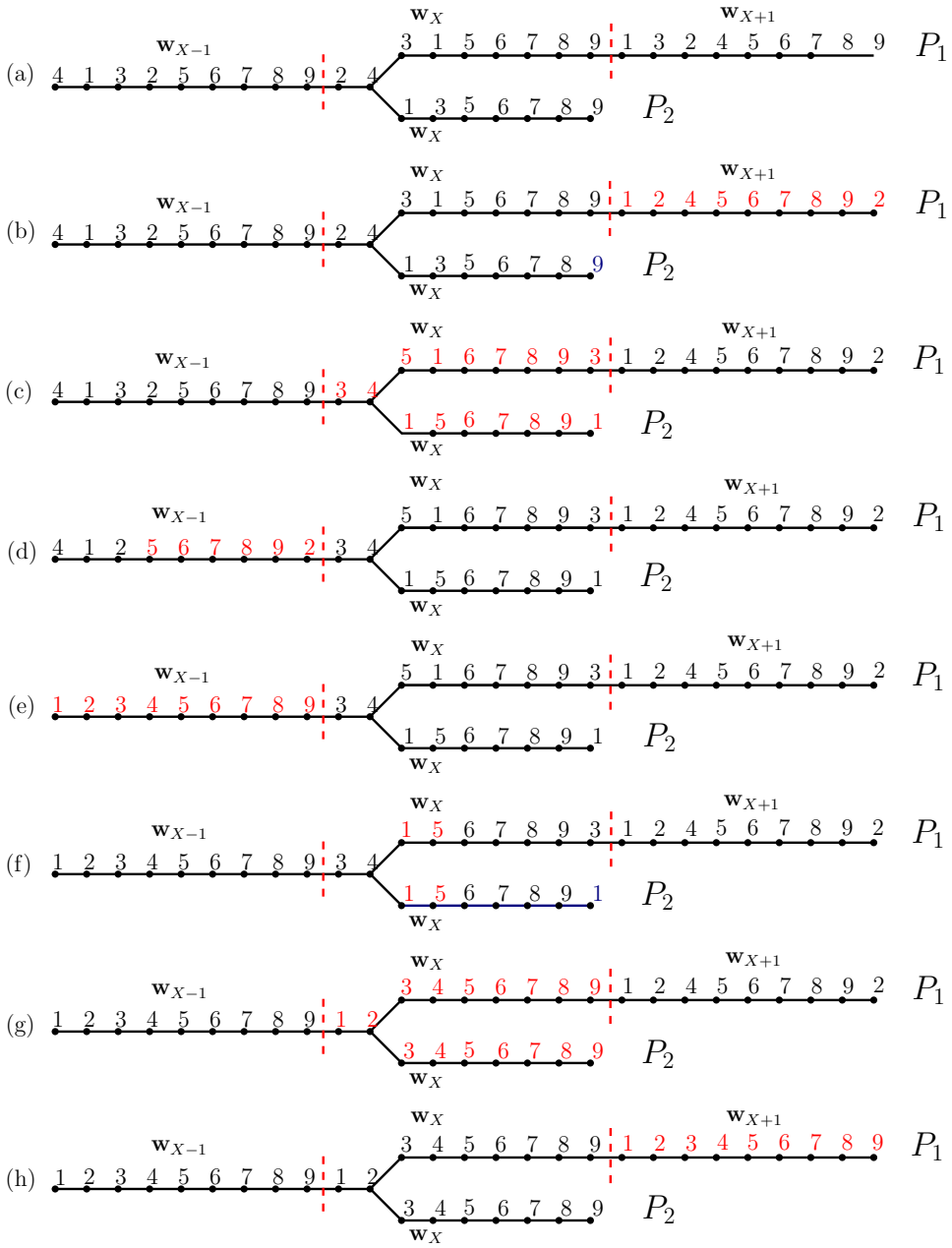


Figure 15: Step-by-step transformation of an almost symmetric 9-coloring into a symmetric 9-coloring on a 5-tree