Convex recoloring of leaf-colored trees

Emgad H. Bachoore

Hans L. Bodlaender

Department of Information and Computing Sciences,
Utrecht University

Technical Report UU-CS-2006-010

www.cs.uu.nl

ISSN: 0924-3275

Convex recoloring of leaf-colored trees *

Emgad H. Bachoore and Hans L. Bodlaender

Institute of Information and Computing Sciences, Utrecht University,
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
bachoore@cs.uu.nl hansb@cs.uu.nl

Abstract. A coloring of the leaves of a tree T is called *convex*, if it is possible to give each internal node a color, such that for each color, the set of nodes with that color forms a subtree of T. Motivated by a problem from phylogenetic reconstruction, we study the problem, when given a tree with a coloring of its leaves, to recolor as few as possible leaves to obtain a convex coloring. We present first a linear time algorithm for verifying whether or not a given leaf colored tree is convex colorable. Then, we give a number of preprocessing rules for reducing the size of the given tree or splitting it into two or more subtrees. Finally, we introduce a branching algorithm for solving the problem in $4^{OPT} \cdot n$, where OPT is the optimal solution for solving the problem, and show that the problem is fixed parameter tractable.

1 Introduction

The problem to obtain a 'likely' evolutionary tree from biological or linguistic data (the phylogenetic reconstruction problem) gives rise to several interesting combinatorial problems. One of these is the problem of convex recoloring. The most studied variant of this problem is the following: we are given a tree T with a coloring of the nodes of the tree. We ask how we can change as few as possible colors of a node, such that we obtain a coloring where for each color, the nodes with that color form a connected subtree. The colors represent a value for some characteristic for the items represented by the nodes; from the phylogenetic application, one expects that with perfect data, each color class forms a connected subtree. As the data may contain errors, the input possibly does not have this connectivity property, and we ask how we can 'correct' as few as possible colors / values, to get a tree with this connectivity property. This problem was extensively studied by Snir [5], see also [3]. An approximation algorithm (for a generalized weighted variant) was obtained by Moran and Snir [4]. A recent improvement was obtained by Bar Yehuda et al. [1].

An interesting variant is when only the values for the leaves of the tree are given. E.g., the leaves represent species that are presently known, and values for internal nodes (representing possibly hypothetical extinct species) are not known. This variant leads to the following combinatorial problem: given is a tree T with a coloring of the leaves of T. We ask to recolor as few as possible leaves of T, such that the resulting coloring has the property that we can assign a color to each leaf of T and obtain a total coloring of T where for each color, the set of nodes with that color form a connected subtree of T.

This CONVEX RECOLORING OF LEAF-COLORED TREES problem was first studied by Snir in [5], who showed that the problem is NP-complete. In this paper, we obtain several further results on the problem. After some preliminaries in Section 2, we give in Section 3, a linear time algorithm for verifying whether or not a leaf colored tree is convex colorable. In Section 4, we present a number of safe preprocessing for reducing the size of the tree to an equivalent smaller one, or splitting the tree into two or more subtrees. Finally, in Section 5, we use a branching technique to develop an algorithm for the problem to determine, given a leaf-colored tree, what is the minimum number of leaves that must be recolored to obtain a convex colorable tree, and this shows that the problem is fixed parameter tractable.

2 Preliminaries

All trees used in this paper are rooted. Whenever we refer to paths, we assume these are simple. To avoid the ambiguity in the meanings of some terms, we use some of the terminology, introduced in [5].

* This work has been supported by the Netherlands Organization for Scientific Research NWO (project TACO: 'Treewidth And Combinatorial Optimization').

If $C_1 : V_1 \to X$ and $C_2 : V_2 \to X$ are functions with disjoint domains ($V_1 \cap V_2 = \emptyset$), then $C_1 + C_2 : V_1 \cup V_2 \to X$ is the function with for $x \in V_1$: $(C_1 + C_2)(x) = C_1(x)$, and $x \in V_2$: $(C_1 + C_2)(x) = C_2(x)$.

A *tree* or *free tree* is a graph connecting n nodes with $n - 1$ edges such that there is exactly one path between any two given nodes. A *rooted tree* is defined as a free tree in which a particular node has been specified as the root. In a rooted tree, each node v that is not the root has a *parent*, namely the unique neighbor of v that is on the path from v to the root. The parent of v is denoted parent(v). If w is the parent of v, then v is a *child* of w. Let for all $v, w \in V$, p_{vw} denote the path in T from v to w. If v is on the path from w to the root of the tree ($v \neq w$), then w is a *descendant* of v, and v is an *ancestor* of w. A node is a *leaf* in a rooted tree, if it has no children.

A *total colored tree* (or, for short: *colored tree*) is a pair (T, C) where T= (V, E) is a tree and C is a coloring of T, i.e., a function $V \to \mathcal{C}$ from V onto a set of colors $\mathcal{C}$. A *partial colored tree* is a pair (T, C) where $T = (V, E)$ is a tree, and C is a coloring of some of the nodes of T, i.e., a function $W \to \mathcal{C}$ from a subset of nodes $W \subseteq V$ onto a set of colors $\mathcal{C}$. A *leaf colored tree* is a partial colored tree (T, C) in which a node has a color, if and only if it is a leaf.

A *block* in a colored tree is a maximal set of nodes induces a monochromatic subtree, i.e., a maximal set of nodes of the same color that induce a connected subtree of the tree. A d*-block* is a block of color d. A coloring C is said to be *convex* (or, also sometimes called: *connected*) if the number of d-blocks equals 1 for every color $d \in \mathcal{C}$. In other words, the coloring is convex, if for each color d, the set of nodes with color d forms a connected subtree. A *total convex colored tree* is a total colored tree whose coloring is convex.

A total coloring C' of a tree T is an *extension* of a partial coloring C of T, if for every node v with a color in C, $C'(v) = C(v)$. A partial colored tree (T, C) is *convex colorable*, if and only if there is a convex total coloring that extends it. The same definitions apply also for leaf colored trees; i.e., a a leaf colored tree is convex colorable, if we can transform it to a total convex colored tree without changing the colors of its leaves.

A colored tree $(T = (V, E), C')$, partial or total, is viewed as a *recoloring* of a given colored tree (T, C), if there is at least one node $v \in V$, $C(v) \neq C'(v)$. We say that the recoloring C' of C *retains* the color of node v, if $C(v) = C'(v)$, otherwise C' *overwrites* v. For a recoloring C' of C, $\mathcal{X}_{\mathcal{C}}(C')$ (or $\mathcal{X}$ (C')) is the set of nodes overwritten by C', i.e., $\mathcal{X}_{\mathcal{C}}$ $(C') = \{v \in V | C(v)$ is defined and $C(v) \neq C'(v)\}$.

The *degree of a recoloring* C' of C, $\delta_C(C')$, is the number of nodes overwritten by C', i.e., $\delta_C(C') = |\mathcal{X}_C(C')|$. A coloring C^* is an *optimal convex recoloring* of C, or in short an *optimal recoloring* of C, and $\delta_C(C^*)$ is denoted by $OPT(T, C)$, if C^* is a convex coloring of T for which $\delta_C(C^*) = min_C\{\delta_C(C')\}$, where the minimum is taken over all convex recolorings of C.

For a rooted tree $T = (V, E)$ with root r, and a node x, the tree T_x is the subtree of T rooted at x, i.e., the subtree of T induced by x and all descendants of x. If C is a (partial) coloring of T, then C_x is the restriction of C to T_x.

A recoloring $\mathcal{C}^*$ is an *optimal leaf convex recoloring* of C for a leaf colored tree (T, C), or in short an *optimal leaf recoloring* of C, if the number of recolored leaves is the minimum over all convex colorable recolorings of C. This number is then denoted as $OPT(T, C)$.

A *weighted colored tree* is a colored tree to whose nodes labels (usually number) are assigned. The word "weight" also has a more specific meaning when applied to colored trees, namely, the weight of a node v of a color c in a colored tree (T, C), $w_c(v)$, is the cost of using color c for coloring node v in (T, C). The *degree of a recoloring* C' of C, $\delta_C(C')$, in a weighted colored tree, is the sum of the weights of the nodes overwritten by C', i.e., $\delta_C(C') = \sum w(v_i)$, where the sum is taken over all nodes v_i whose color is overwritten in the tree.

Lemma 1. *Let (T, C) be a leaf colored tree such that every leaf in T has a different color. Then C is convex colorable. Hence, $OPT(T, C) = 0$.*

Lemma 2. *Let (T, C) be a leaf colored tree. If there is only one color $c \in \mathcal{C}$ that is assigned to more than one leaf of the tree and each of the other colors $d \in \mathcal{C}$, $d \neq c$ is assigned to one leaf of the tree, then the leaf colored tree is convex colorable.*

Lemma 3. *Let m be the number of leaves in a leaf colored tree (T, C) and d be the number of different colors used for coloring the leaves of T. $OPT(T, C) \leq m - d$.*

Proof. For each color $c \in \mathcal{C}$, we give each node of color c, except one of these, a new color. Now, every leaf in the tree has a different color from the others. Thus, the tree is convex colorable, and we have recolored $m - d$ leaves. □

Corollary 1. *Let (T, C) be a leaf colored tree with m leaves and let $n_1, n_2, \cdots, n_l$ be the numbers of leaves with colors $c_1, c_2, \cdots, c_l$, such that $n_1 + n_2 + \cdots + n_l = m$. The minimum number of leaves that should be recolored to transform (T, C) to a convex colorable tree is at most $k = m - \max\{n_1, n_2, \cdots, n_l\} - (l - 1)$.*

Proof. Suppose $n_r = \max\{n_1, n_2, \cdots, n_l\}$. For each color c_s, $s \neq r$, we give each node of color c_s, except one of these, a new color. Now, only one color (c_r) is given to possibly more than one leaf; each other color is given to at most one leaf. Thus the tree is convex colorable, and we have recolored k leaves. □

Definition 1. *A tree $T = (V, E)$ is a* star, *if T has the following form: There is one special node (root) r in T with zero or more children, and every other node in the tree except r has at most one child.*

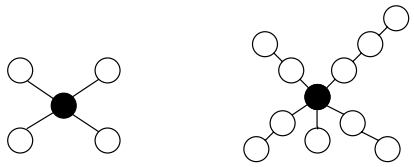


Fig. 1. Two trees that are a star

Lemma 4. *Let (T_1, C_1) and (T_2, C_2) be two leaf colored trees such that the number of leaves in T_1 equals the number of leaves in T_2, and for each color c, the number of nodes with color c in T_1 equals the number of nodes with color c in T_2, and T_1 is a star. Then $OPT(T, C_1) \geq OPT(T_2, C_2)$.*

Lemma 5. *Let $(T = (V, E), C)$ be a total convex colorable tree, $v_i, v_j \in V$, each of v_i and v_j is not a descendant of the other, $color(v_i) = color(v_j)$. Then color(parent(v_i) = color(parent(v_j)).*

Lemma 6. *Let (T_s, C_s) be a leaf colored subtree of a leaf colored tree (T, C). Then $OPT(T_s, C_s) \leq OPT(T, C)$.*

Lemma 7. *Let $(T = (V, E), C)$ be a colored tree, P_{vw} be a path between nodes v and w, x be a node on the path P_{vw}. If color(v)=color(w) $\neq$ color(x), then the tree (T, C) is not convex colorable.*

3 Verifying Convex Colorability

In this section, we show that we can test in $O(n)$ time if a leaf colored tree is convex colorable. We use a simple characterization of convex colorability in terms of crossing paths, and also show that we can find in $O(n)$ time a pair of crossing paths for a leaf colored tree that is not convex colorable.

Definition 2. *Let p_1, p_2 be paths in a leaf colored tree (T, C). We call p_1 and p_2 a pair of* crossing paths, *if the following conditions hold.*

- *The endpoints of p_1 are leaves with the same color.*
- *The endpoints of p_2 are leaves with the same color.*
- *The color of the endpoints of p_1 is different from the color of the endpoints of p_2.*
- *p_1 and p_2 intersect: $V(p_1) \cap V(p_2) \neq \emptyset$.*

Theorem 1. *Let (T, C) be a leaf colored tree. Let p_1 and p_2 be crossing paths, with p_1 a path between leaves v_1 and v_2, and p_2 a path between leaves w_1 and w_2. Then in a convex colorable recoloring of (T, C), at least one node from $\{v_1, v_2, w_1, w_2\}$ is recolored.*

Proof. Let v_1, v_2, w_1, and w_2 be the leaves of the crossing paths p_1 and p_2 in a leaf colored tree (T, C). Let x be the first common ancestor of these leaves. If we assign the color $c = C(v_1) = C(v_2)$ to the node x, then either w_1 or w_2 must be recolored, otherwise the set of vertices with the color of w_1 is not convex. Similarly, when we assign the color of nodes w_1 and w_2 to x. When we assign a color $c \notin \{c, d\}$ to x, then c 'breakes' the color of v_1 and v_2 as well as the color of w_1 and w_2, and we have to recolor at least two of the leaves. □

Theorem 2. *Let (T, C) be a leaf colored tree. (T, C) is convex colorable, if and only if there is no pair of crossing paths in (T, C).*

Proof. If there is a pair of crossing paths, then by Theorem 1, at least one leaf must be recolored, so the tree is not convex colorable.

If there is no pair of crossing paths, then we can obtain a convex coloring of T as follows. For each internal node x, if x is on a path between two leaves with the same color c, then color x with c. If x is on no such path, we give x a new color, not given to any other node c_x. One can easily verify that this is a convex coloring. □

Definition 3. *Let v be a node in a leaf colored tree (T, C).*

$$n_v(i) = \begin{cases} 0 & \text{if } v \text{ has a child } w \text{ such that all leaves with color } i \text{ are a descendant of or equal to } w, \text{ or there is no leaf with color } i \text{ in the subtree rooted at } v, \\ \sum_{w \in children(v)} n_w(i) & \text{otherwise.} \end{cases}$$

Lemma 8. *$n_v(i) \geq 1$, if and only if there is a path between two leaves with color i that uses v.*

Proof. By the definition of $n_v(i)$, $n_v(i) = 0$, if and only if v has a child w such that all leaves with color i are a descendant of or equal to w, or there is no leaf with color i in the subtree rooted at v, otherwise $n_v(i) \geq 1$. Hence, if $n_v(i) = 0$, then there is no path between any two leaves with color i that uses v, otherwise $n_v(i) \geq 1$.

If $n_v(i) \geq 1$, then by the definition of $n_v(i)$, $n_v(i)$ equals the sum of the $n_w(i)$, where $w \in children(v)$. If v has two (or more) children w_1, w_2, with $n_{w_1}(i) \geq 1$ and $n_{w_2}(i) \geq 1$, then there is a descendant y_1 of w_1 with color i and a descendant y_2 of w_2 with color i, and the path from y_1 to y_2 uses v. If v has only one child w with $n_w(i) \geq 1$, then there must be a vertex y with color i that are not a child of w, and hence also not a child of v. Also, v has a descendant z with color i, and a path from y to z uses v. □

Lemma 9. *Let v_1, v_2 be two leaves associated with colors c_1, c_2 respectively, in a leaf colored tree (T, C) such that $c_1 \neq c_2$. Let w be an internal node in T such that v_1 and v_2 belong to the subtree rooted at the node w. If $n_w(c_1) \geq 1$ and $n_w(c_2) \geq 1$, then (T, C) is not convex colorable.*

Proof. Suppose $n_w(c_1) \geq 1$ and $n_w(c_2) \geq 1$. By Lemma 8, there is a path between two leaves with color c_1 that uses w, and a path between two leaves with color c_2 that uses w. These are crossing paths, and hence, by Theorem 1, at least one leaf must be recolored to obtain a convex colorable tree. □

Lemma 10. *Let x, y be two nodes in a total convex colored tree (T, C). Suppose y is a descendant of the parent of x, but y is not a descendant of x. If $color(x) = color(y)$ then for every node $z = ancestor(y)$, $z \neq ancestor(parent(x))$, $color(z) = color(x) = color(y)$.*

Lemma 11. *Let S_v be the set of the colors of the children of a node v in a convex colorable tree (T, C). There is at most one color in the set S_v which is repeated for more than one child of v.*

Corollary 2. *Let v and w be two nodes in a convex colorable tree (T, C), $parent(v) = parent(w) = x$. If $color(v) = color(w)$, then $color(x) = color(v)$.*

Theorem 3. *Let $(T_1 = (V_1, E_1), C_1 : V_1 \rightarrow \mathcal{C}_1), \cdots, (T_k = (V_k, E_k), C_k : V_k \rightarrow \mathcal{C}_k)$ be color connected trees rooted at nodes $v_1, \cdots, v_k$ respectively. Let $B = (\mathcal{C}_1 \cap \mathcal{C}_2) \cup \cdots \cup (\mathcal{C}_1 \cap \mathcal{C}_k) \cup \cdots \cup (C_{k-1} \cap \mathcal{C}_k)$, and let $T_x = (V_x, E_x), V_x = V_1 + \cdots + V_k + \{x\}, E_x = E_1 + \cdots + E_k + \{x, v_1\} + \cdots + \{x, v_k\}$ as in Figure 2, i.e., x is the parent of nodes $v_1, \cdots, v_k$. If $|B| = 0$, then T_x is colored connected whatever color we use for coloring the node x. If $|B| = 1$, then T_x is colored connected if and only if $color(x) = (C_1(v_1) \cap C_2(v_2)) \cup \cdots \cup (C_{k-1}(v_{k-1}) \cap C_k(v_k)) = b \in B$. If $|B| > 1$, then T_x is not colored connected whatever color we use for coloring the node x.*

Proof. If $|B| = 0$, then there is no common color between the colors of the leaves of the trees rooted at $v_1, \cdots, v_k$. Now, if we assign any color $c \notin \mathcal{C}$ to x, then the tree rooted at x will not include any pair of crossing paths. Therefor, T_x is convex colorable, (see Theorem 2).

If $|B| = 1$, then there is one common color between, at least, two leaves of two different trees of the trees rooted at $v_1, \cdots, v_k$. Therefore the tree rooted at x is convex colorable if and only if the color we assign to x is the same color that is common between the colors of the leaves of $T_{v_1}, \cdots, T_{v_k}$ and is also the same common color between the colors of $v_1, \cdots, v_k$, (see Lemma 11).

Finally, if $|B| > 1$, then there are at least two common colors between the leaves of two or more trees $T_{v_1}, \cdots, T_{v_k}$. This means that the tree T_x includes at least one pair of crossing paths. Therefor, whatever color we assign for x the tree T_x will not be a convex colorable, (see Theorems 1 and 2). □

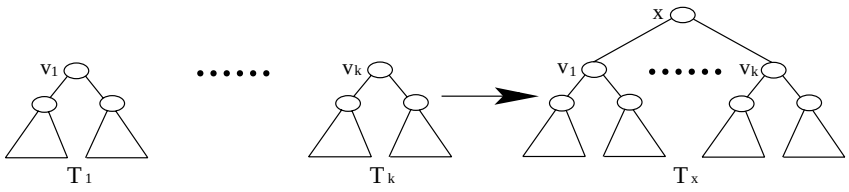


Fig. 2. Connected colored trees combination

In Figure 3, we give the pseudo-code of the algorithm Is-Convex-Colorable for testing whether a leaf colored tree $(T = (V, E), C : V \rightarrow \mathcal{C})$ is convex colorable or not. Let $tot(1), \cdots, tot(|\mathcal{C}|)$ be the numbers of leaves of colors $1, \cdots, |\mathcal{C}| \in \mathcal{C}$ in (T, C), respectively. In this algorithm, we traverse the nodes of the tree in postorder. We compute $n_v(i)$, for each visited node v, as it is given in Definition 3. In other words, for each visited leaf $v \in V$, we have $n_v(c) = 1$ for the color c of v, and for all colors $c' \neq c$, we have $n_v(c') = 0$. Whereas, for each visited internal node $v \in V$, $n_v(i)$ equals the sum of the values $n_w(i)$ over all children w of v if none of these $n_w(i)$ equals $tot(i)$, otherwise it is 0.

At each internal node v, we test whether or not the subtree rooted at v, in T, is convex colorable, as follows: For each node v, if there are two colors i, j, with $n_v(i) \geq 1$ and $n_v(j) \geq 1$, then the subtree rooted at v is not convex colorable and therefore (T, C) is also not convex colorable. Thus, (T, C) is convex colorable, if and only if there is no $i, j \in \mathcal{C}$, $i \neq j$ and $n_r(i) \geq 1$ and $n_r(j) \geq 1$, where r is the root of the tree.

The pseudo-code given suggest an algorithm that runs in $O(nk)$ time when we have a tree with n nodes and k colors were used for coloring the leaves. However, if we store at each node only the values $n_v(c)$ that are not zero, then at each node that is processed except possibly the last one, we store only one value (the algorithm stops as soon as we have a node where two values $n_v(c)$, $n_v(c')$ both are non-zero). In this way, the algorithm takes linear time.

Theorem 4. *Let (T, C) be a leaf colored tree with n nodes. There is an algorithm to test whether (T, C) is convex colorable or not that uses $O(n)$ time.*

It is also possible to find in $O(n)$ time a pair of crossing paths if the tree is not convex colorable. We first run Algorithm Is-Convex-Colorable. Suppose it returns no when for node x, we have $n_x(i) \geq 1$ and $n_x(j) \geq 1$. Then, there are crossing paths with leaves of colors i and j that intersect at node x. A simple search can give us these paths: for each neighbor y, look if there is a path from y to a leaf with color i avoiding x. If we have two such paths from neighbors y_1 and y_2, these paths, together with x form a path between two leaves with color i that uses x. We can do the same for color j, and we have the desired crossing paths. Standard techniques make this run in linear time.

Theorem 5. *Let (T, C) be a leaf colored tree with n nodes that is not convex colorable. There is an algorithm that finds in $O(n)$ time a pair of crossing paths.*

Algorithm Is-Convex-Colorable (x)

Input: A leaf colored tree $(T_x, C : V_x \rightarrow \mathcal{C})$, rooted by node x, $T_x = (V, E)$, $V \leftarrow \{v_1, \cdots, v_n\}$, $C \leftarrow \{c_1, \cdots, c_k\}, n, k \in \mathbb{N} - \{0\}$, and $tot(1), \cdots, tot(|\mathcal{C}|)$ be the number of leaves in (T_x, C) of colors $1, \cdots, |\mathcal{C}| \in \mathcal{C}$.
Output: "yes" if (T_x, C) is convex colorable, and "no" otherwise.

```
1  if (isleaf(x)) then n_x(i) = 1 such that, i is the color of x
2  else
3      n_x(i) ← 0, i ← 1, ··· , |C_x|;
4      foreach y ∈ children(x)
5          Is-Convex-Colorable (y);
6          if (n_y(i) < tot(i)) then n_x(i) ← n_x(i) + n_y(i);
7      if (∃i, j ∈ C_x, i ≠ j: n_x(i) ≥ 1 and n_x(j) ≥ 1) then return "no" , exit;
8  if (x = root) then return "yes";
```

Fig. 3. Pseudo-code of the Algorithm Is-Convex-Colorable.

4 Leaf Colored Tree Preprocessing Rules

There are several methods for preprocessing a leaf colored tree before running an algorithm on it for finding the minimum number of leaves that should be recolored to transform it into an optimal convex colorable tree. With preprocessing,we hope to decrease the size of the input tree. The algorithm for finding the minimum number of recolored leaves thus often runs on a smaller instance, and hence can be much faster. For example, we first preprocess the tree, and then run a slow exact algorithm on the reduced instance. We consider two types of preprocessing rules in this section. These are *Reduction Rules (Simplification)* and *Splitting Rules (Divide and Conquer)*. With the reduction rules, we change the given tree into a smaller 'equivalent' one. Whereas, with the splitting rules, the given tree is split into two or more smaller parts.

Definition 4. *A* **splitting rule** R*,*

$$R : (T, C) \rightarrow (T_1, C_1), \cdots, (T_m, C_m)$$

is a rule for breaking down a leaf colored tree (T, C) into two or more subtrees $(T_1, C_1), \cdots, (T_m, C_m)$, $m > 1$, such that,

1. $V(T_1) \cup \cdots \cup V(T_m) = V(T)$.
2. $V(T_1) \cap \cdots \cap V(T_m) = \phi$.
3. $E(T_1) \cap \cdots \cap E(T_m) = \phi$.
4. $E(T_1) \cup \cdots \cup E(T_m) \cup \{x_1, y_1\} \cup \cdots \cup \{x_m, y_m\} = E(T)$, *where* $\{x_1, y_1\}, \cdots, \{x_m, y_m\} \in E(T)$, $\{x_1, y_1\}, \cdots, \{x_m, y_m\} \notin E(T_1), \cdots, E(T_m)$.

*We call a tree splitting rule R **safe**, if and only if*

$$OPT(T_1, C_1) + \cdots + OPT(T_m, C_m) = OPT(T, C).$$

Definition 5. *A* **reduction rule** R*,*

$$R : (T, C) \rightarrow (T', C')$$

*is a rule for transforming leaf colored trees (T, C) into smaller trees (T', C'), namely, $|V(T')| < |V(T)|$ and $|E(T')| < |E(T)|$. We call a tree reduction rule R **safe**, if and only if*

$$OPT(T', C') = OPT(T, C).$$

In the following, we present three preprocessing rules. The first preprocessing rule is a splitting rule with two special cases, while the second and the third preprocessing rules are reduction rules. Furthermore, we show how to transform some of the subtrees obtained from splitting or reduction rules to optimal convex colorable subtrees.

Preprocessing Rule 1: Independent Leaf Colored Subtree (ILCS)

Definition 6. *Let T_s be a subtree of a tree T, rooted at node s. $T \ominus T_s$ is the tree $T - E(T_s)$ obtained by deleting from T the edges of T_s and the resulting isolated nodes, if they exist, i.e., let $\zeta(T) = (\{v \in V(T) | \, deg(v) \geq 1\}, E(T))$. $T \ominus T_s = \zeta(V(T), (E(T) \setminus E(T_s))$.*

Definition 7. *Let $(T_s = (V_s, E_s), C_s : V_s \rightarrow \mathcal{C}_s)$ be a leaf colored subtree of a leaf colored tree $(T = (V, E), C)$. We call (T_s, C_s) an **independent leaf colored subtree** of (T, C) if $T_t = T \ominus T_s$ and $\mathcal{C}_s \cap \mathcal{C}_t = \phi$.*

ILCS Rule:
let (T_s, C_s) be a leaf colored subtree of a leaf colored tree (T, C);
if (T_s, C_s) is independent
then
  let $T_1 = T_s$; let C_1 be C restricted to T_1;
  let $T_2 = T \ominus T_s$; let C_2 be C restricted to T_2;
  split (T, C) into (T_1, C_1) and (T_2, C_2).

An optimal total convex colorable tree (T, C^*) can be obtained by combining optimal convex colorable trees (T_1, C_1^*) and (T_2, C_2^*).

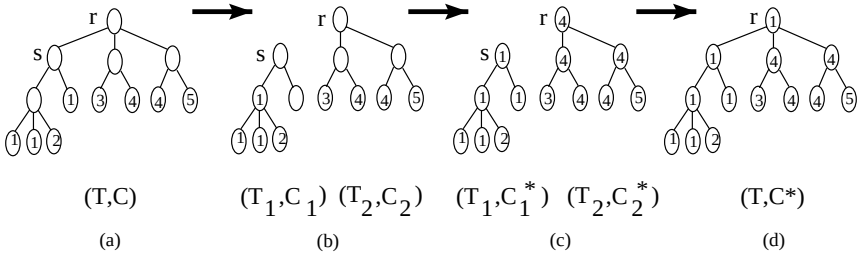


Fig. 4. An example for splitting and recoloring a leaf colored tree by using the Independent Leaf Colored Subtree (ILCS) rule.

Lemma 12. *The Independent Leaf Colored Subtree (ILCS) rule is safe.*

Proof. We claim that $OPT(T_1, C_1) + OPT(T_2, C_2) = OPT(T, C)$. Take an optimal convex recoloring C_1^* of T_1 and an optimal convex recoloring C_2^* of T_2. We can always find such recolorings such that new colors in C_1^* are not used in C_2^*. Thus, no color used in C_1^* is used in C_2^* and hence $C_1^* + C_2^*$ is convex colorable for T. Hence, $OPT(T_1, C_1) + OPT(T_2, C_2) \leq OPT(T, C)$.

Suppose we have an optimal convex recoloring C^* of T. Let C_1^* be obtained by restricting $\mathcal{C}$ to T_1, and C_2^* be obtained by restricting $\mathcal{C}$ to T_2. Now, $OPT(T, C) = \delta_C(C^*) = \delta_{C_1}(C_1^*) + \delta_{C_2}(C_2^*) \leq OPT(T_1, C_1) + OPT(T_2, C_2)$. □

In the following two subsections, we describe two special cases of the Independent Leaf Colored Subtree (ILCS) rule. These are: the Leaf Of a Unique Color (LOUC) rule and the Parents Of the Same Leaves (POSL) rule.

Leaf of a Unique Color (LOUC)

Suppose that the given leaf colored tree has a leaf whose color is different from the colors of other leaves. Then, this leaf can be considered as an independent leaf colored subtree of the given leaf colored tree.

Definition 8. *Let v be a leaf node in a leaf colored tree (T, C). If no leaf $w \neq v$ in (T, C) has the same color as v, then we call v a **leaf of a unique color (LOUC)**.*

LOUC Rule:
let x be a leaf of a unique color in a leaf colored tree (T, C) rooted at r;
while $(|children(parent(x))| = 1)$ **set** $parent(x) \leftarrow x$;
let T_1 be the subtree of T rooted at node x,
T_2 be the subtree of T obtained by removing T_1 from T, with root r,
C_1 be obtained by restricting coloring C to T_1,
C_2 be obtained by restricting coloring C to T_2;
split (T, C) into (T_1, C_1) and (T_2, C_2);

Note that the tree that is split off is obtained by taking a leaf whose color is not given to any other leaf, and then following the path from that leaf up to the tree, just before we find a node which has at least two children.

After we split a leaf colored tree (T, C) into two leaf colored subtrees (T_1, C_1) and (T_2, C_2) by using the LOUC splitting rule, it is easy to transform (T_1, C_1) to an optimal total convex colorable tree (T_1, C_1^*). We can do that by coloring every node in T_1 by the color of the leaf of a unique color. If (T_2, C_2) can be transformed to an optimal total convex colorable subtree, then an optimal total convex colorable tree (T, C^*) can be obtained by combining (T_1, C_1^*) with (T_2, C_2^*), namely, $(T = (V, E), C^*) = ((V(T_1) + V(T_2)), (E(T_1) + E(T_2) + \{e | e \in E,\ e \notin E_1,\ e \notin E_2\}), (C_1^* + C_2^*))$.

Lemma 13. *The Leaf Of a Unique Color Rule is safe.*

Proof. This follows from the fact that the LOUC rule is a special case of the ILCS rule. □

Parents Of the Same Leaves (POSL) rule

Definition 9. *We call two leaf colored trees $(T_1 = (V, E), C)$ and $(T_2 = (W, F), D)$* equivalent, *if there is a bijection $f : V \to W$, such that for all $x, y \in V$: $\{x, y\} \in E \Leftrightarrow \{f(x), f(y)\} \in F$, and for all $v \in leaves(V)$, $C(v) = D(f(v))$.*

For rooted trees, we require additionally that f maps the root of the first tree to the root of the second tree. Clearly, when leaf colored trees are equivalent, the solution to the problem to find an optimal leaf convex recoloring in one tree can directly be translated to a solution for the other tree.

Definition 10. *Let v and w be two non leaves in a leaf colored tree T. We call v and w **parents of the same leaves** if they have the same number of children, all their children are leaves, and the color frequencies of their children are equal, namely, the number of children of v with the color i equals the number of children of node w with the color i, for every color i used for coloring the children of v and w.*

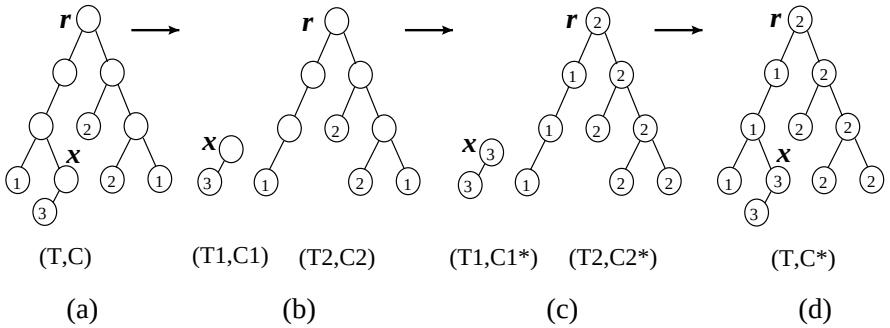


Fig. 5. An example for splitting and recoloring a leaf colored tree by using the LOUC rule. The trees shown in (b) are the result of performing the LOUC rule on the tree in (a). The trees shown in (c) are the results of performing recoloring operations on the trees in (b). The tree shown in (d) is obtained by combining the trees in (c).

POSL Rule:
let $(T = (V, E), C)$ be a leaf colored tree rooted at node r,
$v, w \in V$ be two *parents of the same leaves in* T,
$\mathcal{C}_v$, $\mathcal{C}_w$ be the sets of the colors of the children of v and w, respectively,
T_s be a subtree of T, rooted at the first common parent, s, of v and w,
$T_t = T \ominus T_s$ and
C_s and C_t be the coloring functions of T_s and T_t, respectively;
if each leaf in T_s is a child of v or a child of w and $C_s \cap C_t = \phi$
then split (T, C) into (T_s, C_s) and (T_t, C_t);

We can use the following easy procedure to transform the subtree rooted at node s to a convex colorable tree.

let $d \in \mathcal{C}_s$ be the color that is assigned to most leaves of the subtree T_s,
amongst other colors in $\mathcal{C}_s$;
foreach $v_i \in leaves(T_v)$
if ($color(v_i) \neq d$) and ($color(v_i) = color(v_j)$, $v_j \in leaves(T_v), v_j \neq v_i$
then $color(v_i) \leftarrow d$;
foreach $w_i \in leaves(T_w)$ **if** color(w_i) $\neq d$ **then** $color(w_i) \leftarrow d$;

Lemma 14. *The Parents Of the Same Leaves rule (POSL) is safe.*

Proof. This follows from the fact that the POSL rule is a special case of the ILCS rule. □

Preprocessing Rule 2: The Unary Path Rule (UP)

Definition 11. *Let p_{ab} be the path between nodes a and b in a leaf colored tree $(T = (V, E), C)$, such that $|p_{ab}| \geq 3$. We call the path p_{ab} a* unary path, *if for each node v on p_{ab}, $v \notin \{a, b\}$, $|children(v)| = 1$.*

UP Rule:
let $p_{ab} = (a, \cdots, b)$ be a unary path in a leaf colored tree (T, C), rooted at
node r such that b is a descendant of a;
let T' be the tree, obtained by removing all nodes on p_{ab} except a and b
from T, with their incident edges, and adding an edge between a and b; $C' = C$;

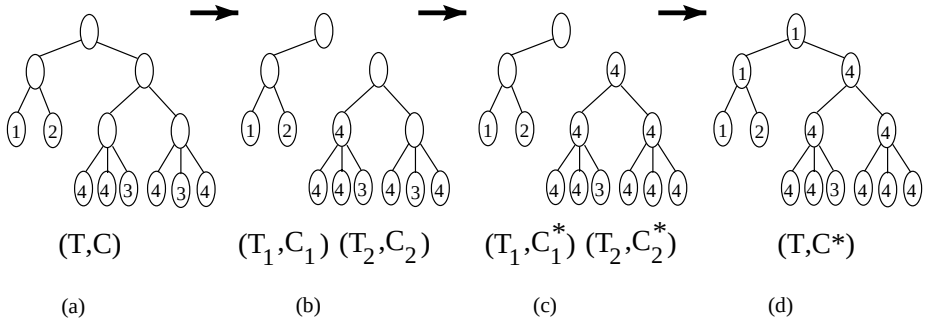


Fig. 6. An example for splitting and recoloring a leaf colored tree by using the Parents Of The Same Leaves (POSL) rule.

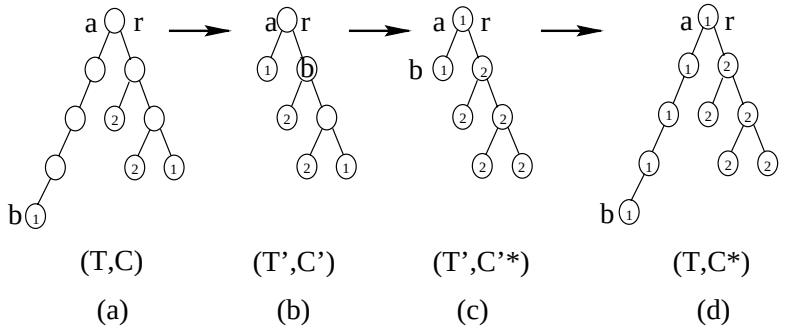


Fig. 7. An example for reducing and recoloring a leaf colored subtree by using the The Unary Path rule (UP).

In order to transform the given tree (T, C) to an optimal convex colorable tree (T, C^*), we first transform the tree (T', C') to an optimal convex colorable tree (T', C'^*). Then, we set the color of each node in the tree T equals to its corresponding node in the tree T'. Next, we set the colors of all nodes $v \in V(p_{ab})$ except nodes a and b in (T, C'^*) equal to the color of the node b in (T, C'^*), namely, $C^*(v_i) = C'^*(b)$ for each $v_i \in V(p_{ab}),\ v_i \neq a$ and $v_i \neq b$.

Lemma 15. *Let Pab be a path in a convex colorable tree (T, C^*), $|V(p_{ab}| \geq 3$. Let $x,\ z$ are two nodes on the path p_{ab}, such that $C^*(x) = C^*(z)$. If there is a node y between nodes x and z on the path p_{ab}, then $C^*(y) = C^*(x) = C^*(z)$.*

Lemma 16. *The Unary Path (UP) preprocessing rule is safe.*

Proof. Let p_{ab} be a unary path between nodes a and b in a leaf colored tree $(T = (V, E), C)$ such that b is a descendant of a. Let (T', C') be the tree obtained from applying Preprocessing Rule 3 on the tree (T, C). Suppose we have an optimal convex colored tree (T', C'^*) of the leaf colored tree (T', C') with corresponding total convex coloring (T', C''). If we use the same recoloring of the leaves for T, we obtain a convex colorable leaf colored tree (T, C^*) (for each leaf v in T or T': $C'^*(v) = C^*(v)$). The corresponding total coloring C''' of T can be obtained from C'' by setting the color of each internal node on the path p_{ab} to the color of b in C''. Thus, the number of recolored leaves has not changed. Hence, $OPT(T', C') \geq OPT(T, C)$. Now, suppose we have an optimal convex colorable tree (T, C^*) and we delete the nodes and their incident edges on the path between nodes a and b, and then add an edge between a and b, then the tree we obtain is also convex colorable. Also here, the number of recolored leaves will not change. Thus, $OPT(T', C') \leq OPT(T, C)$. □

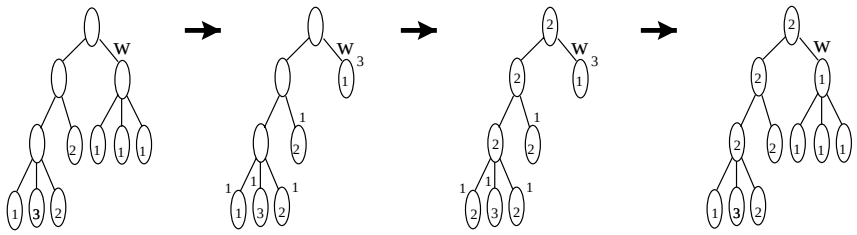
Preprocessing Rule 3: Leaf Siblings Of the Same Character (LSOSC)

Definition 12. *Let S_w be a set of leaves in T with a common parent w. We call S_w, a* ***set of leaf siblings of the same character*** *if all leaves $v \in S_w$ have the same color.*

LSOSC Rule:
let $S_w = \{w_1, \cdots, w_m\}$ be a set of leaf siblings of the same character in (T, C), $m > 1$;
let T' be a tree obtained from the tree T, such that,$V(T') = V(T) - S_w$, $E(T') = E(T) - \{\{w_i, w\} | w_i \in S_w, w$ is the parent of $w_i\}$;
foreach $v \in leaves(V(T'),\ v \neq w$,
  set $C'(v) \leftarrow C(v)$;
  set $weight(v) \leftarrow 1$;
set $C'(w) \leftarrow C(w_1))$;
set $weight(w) \leftarrow |S_w|$;

Note that this rule transforms the tree to one with weights. After we have executed the LSOSC rule on a given tree, we then transform the tree (T', C') to an optimal convex colorable tree (T', C'^*). Then, we add the nodes of the set S_w to the tree (T', C'^*) with an edge between every node $w_i \in S_w$ and the node w. Next. we can use the following coloring rule for coloring the nodes of the set S_w in the tree (T', C'^*). The result is an optimal convex colorable tree (T, C^*) of the tree (T, C).

let $D = \mathcal{C}'^* - \{C'^*(w)\}$;
if $\exists\ x \in V(T'),\ x \neq w$ and $C'^*(x) = \mathcal{C}'^*(w_i),\ w_i \in S_w,\ 1 \leq i \leq m$
then for $(i = 1$ to $m)\ C'^*(w_i) \leftarrow C'^*(w)$;
else for $(i = 1$ to $m - 1)\ C'^*(w_i) \leftarrow C'^*(w)$;



,

Fig. 8. An example for reducing and recoloring a leaf colored tree by using the Leaf Siblings Of the Same Character (LSOSC) rule. The numbers inside the nodes represent the colors of the nodes and the numbers outside the nodes represent their weights.

Lemma 17. *Let $S_w = \{w_1, \cdots, w_m\},\ m > 1$ be a set of leaf siblings of the same character in a leaf colored tree $(T = (V, E), C)$, c_1 be the color of the nodes of the set S_w and $a, b, c, d \in V,\ a, b, c \notin S_w,\ d \in S_w,\ C(a) = C(b) = c_2,\ C(c) = C(d) = c_1$,*

1. *If p_{ab} and p_{cd} are two crossing paths in (T, C), then for every $w_i \in S_w$, p_{cw_i} and p_{ab} are crossing paths in (T, C).*
2. *If p_{ab} and p_{cd} are not crossing paths in (T, C), then for every $w_i \in S_w$, p_{cw_i} and p_{ab} are not crossing paths in (T, C).*

Proof. Let $p_{ab} = (a, \cdots, b)$, $p_{cd} = (c, \cdots, w, d)$, where w is the parent of d in the tree (T, C). If p_{ab} and p_{cd} are crossing paths in the tree (T, C) and w is the parent of the set of the nodes of the same character, S_w, then $w \notin V(p_{ab}) \cap V(p_{cd})$. This means, $V(p_{ab} \cap V(p_{cw_i}) = V(p_{cd})$ for every $w_i \in S_w$. Therefore, for every $1 \leq i \leq m$, p_{ab} and p_{cw_i} are crossing paths in (T, C). In the same manner, we can prove that if p_{ab} and p_{cd} are not crossing paths in (T, C), then for every $w_i \in S_w$, p_{cw_i} and p_{ab} are not crossing paths in (T, C). □

Corollary 3. *Let (T, C^*) be an optimal convex colorable recoloring of a leaf colored tree (T, C), S_w be a set of leaf siblings of the same character in (T, C). If there is a leaf $x \in S_w$, $C^*(x) = C(x)$, then for every leaf $v \in S_w$, $C^*(v) = C(v)$.*

Proof. Suppose that we have two leaves $x, y \in S_w$, $C^*(x) = C(x)$ and $C^*(y) \neq C(y)$. Thus, we have recolored y in (T, C) to obtain an optimal convex colorable tree (T, C^*). As the recoloring is optimal, a recoloring that leaves the color of y intact is not convex, so there must be nodes a, b, c, with $C(a) = C(b) \neq C(y), C(c) = C(y)$, and p_{ab} and p_{cy} are crossing paths. Now, by Lemma 17, pab and p_{cx} are also crossing paths, hence C^* is not convex, contradiction. □

Lemma 18. *The Leaf Siblings Of the Same Character (LSOSC) rule is safe.*

Proof. Let $S_w = \{w_1, \cdots, w_m\}$ be a set of leaf siblings of the same character in $(T = (V, E), C)$, $m > 1$, w is the parent of these siblings and d be the color of these siblings. Let (T', C') be a tree obtained from (T, C) as it is described in the *LSOSC rule*. We can consider two cases for proving this Lemma.
Case 1: In the given tree, there is no leaf $v \in V$, $v \notin S_w$ and $C(v) = d$ in (T, C), namely, there is no leaf in (T, C) and not in S_w that gets assigned the same color as the leaves in the set S_w. In such a case, if we apply the LSOSC reduction rule on the tree (T, C), then the node w in the weighted leaf colored tree (T', C', w) is a leaf of a unique color. Therefore, we need not to recolor this node to transform (T', C', w) to an optimal convex colorable tree (T', C'^*, w) and thus the nodes of the set S_w in the tree (T, C^*) since according to this rule, the color of the node w is assigned to the nodes of the set S_w.
Case 2: In the given tree, there is a leaf, $v \in V$, $v \notin S_w$ and $C(v) = c(w_i) = d$ in (T, C), namely, there is a leaf v in (T, C), $v \notin S_w$ that gets assigned the same color as the leaves in the set S_w. From Lemma 17 and Corollary 3, we know that, if p_{vx} and p_{ab} are two crossing paths in (T, C), then the nodes of the set S_w in the tree (T, C^*), either, all should be recolored or no node of them should be recolored. Due to the *LSOSC rule*, the nodes of the set S_w are assigned the same color as the node w in (T, C^*) after transforming (T', C', w) into a convex colorable tree (T', C'^*, w). Moreover, the weight of a leaf v in the tree (T, C, w), $w(v)$, corresponds to the number of leaves that should be recolored in the normal leaf colored tree. Therefore, the minimum number of recolored leaves in a leaf colored tree (T, C^*) equals the total weight of recolored leaves of its corresponding (T', C^*, w). □

Corollary 4. *Let (T, C) be a leaf colored tree such that, (T, C) does not include any set of siblings of the same character. Let $x_1, x_2, y_1, y_2 \in leaves(V)$. If $color(x_1) = color(x_2)$, $color(y1) = color(y2)$ and $color(x_1) \neq color(y_1)$, then we have to recolor at least one leaf node in a leaf colored tree (T, C) to transform it to an optimal convex colorable tree (T, C^*).*

Lemma 19. *Let $(T', C' : V \leftarrow C')$ be a leaf colored tree obtained from executing the LSOSC rule on a leaf colored tree (T, C). In order to transform (T, C) to an optimal convex colorable tree (T, C^*), we have to recolor at most $k = |leaves(T)| - 1$ leaves in T, of a total weight $q = t - max(w_1, \cdots, w_{|C'|})$, where t is the total weight of all leaves of the tree (T', C'), w_{c_i} is the total weight of the leaves of color c_i in T'.*

Proof. Let d be the color of the maximum weight in a leaf colored tree (T, C). For every leaf $v \in V$ if $C(v) = d$, then $C'(v) = C(v)$. For every leaf $v \in V$, if $C(v) \neq d$, then $C'(v) = x$ such that, x is a color that is not assigned to any other leaves of T. Thus, (T, C') is a convex colorable for (T, C) and we have recolored at most k leaves of a total weight q because at most one color is repeated for more than one leaf and each of the other leaves has assigned a different color from the others. □

Conclusions

The essential purpose of preprocessing rules is to reduce the size of a problem under study, using relatively little computation time and without losing optimality. The smaller, and presumably easier, problem is subsequently solved. In this section, we discussed preprocessing rules for the problem of transforming a leaf colored tree to an optimal convex colorable tree. We introduced two types of rules for doing that, namely, splitting rules and reduction rules. Under each splitting rule, the given tree is breaking down into two or more subtrees. The combination of the optimal convex colorable subtrees is an optimal convex colorable tree of the given tree. In the second type of rules, we exploit a set of rules for stepwise reducing the problem of finding the convex colorable tree of minimum number of recolored leaves to the same problem on a smaller leaf colored tree. The smaller tree can be transformed to a convex colorable tree using an exact or heuristic algorithm, depending on the tree's size. From the optimal convex colorable tree of the smaller tree, a convex colorable tree of the original tree is obtained by reversing the reduction steps. The splitting and reduction rules are guaranteed not to destroy the optimality. For some leaf colored tree instances, it is sufficient to apply these preprocessing rules with some simple coloring rules to obtain the minimum number of recolored leaves that should be recolored to transform a leaf colored tree into an optimal convex colorable tree, for instance, a leaf colored tree with all leaves having different colors or leaf colored tree of the form star.

The tree or subtrees we obtain from applying these preprocessing rules on the given tree have the following properties: First, there are at least two leaves in each resulting subtree of the same color. Second, there is no subtree within each tree obtained from reduction rule or subtrees obtained from splitting rules, the colors of its leaves are totally different than these of other subtrees. In other words, there is at least one crossing path in each subtree. Third, the number of branches at each node of the subtrees obtained from the preprocessing rules is at least two. Fourth, in the resulted subtrees, there is no set of siblings of the same color.

5 A branching algorithm

In this section, we give an exact algorithm for the CONVEX RECOLORING OF LEAF COLORED TREES problem. Our algorithm shows that this problem is *fixed parameter tractable* when the number of recolored leaves is taken as parameter. More details on fixed parameter tractability follow.

The algorithm uses the *branching* technique. The main algorithm is a decision algorithm: it is given a leaf colored tree and an integer k, and it decides if T can be made convex colorable by giving at most k leaves a new color. We start running the algorithm for $k = 0$, and while the answer is negative, increase k, and run the algorithm for the new value of k, until we have found the optimal number of leaves to recolor. The branching algorithm operates on subinstances that are again a leaf colored tree (T with some leaves recolored with a new color) and an integer k': again, in such a subinstance, we decide if this leaf colored tree can be made convex colorable by recoloring k' leaves.

Our algorithm basically depends on Theorem 1. It is given in Figures 9 and 10 and operates as follows. Algorithm MainBranching starts with $k = 0$, tests if the optimum number of recolored leaves is k, and if not, increases k by one and repeats.

Algorithm Branching receives as input a leaf colored tree (T_x, C) and an integer k, and decides if we can make the tree convex colorable by recoloring at most k leaves. First, it checks, using the procedure from Theorem 4, if (T_x, C) is convex colorable. If so, we have found the desired solution. If (T_x, C) is not convex colorable, and $k = 0$, we know that this subinstance has no solution. Otherwise, we know by Theorem 2, that there must be a crossing pair in (T_x, C). We can find such a crossing pair in linear time (Theorem 5). At least one of the four leaves on these two paths must be recolored (Theorem 1). When recoloring a leaf, we can assume it receives a new color, not given to any other leaf. Thus, we create four subinstances: in each, we recolor one of the four leaves in the crossing paths. Thus, there is a solution with at most k recolored leaves, if and only if there is a solution with at most $k - 1$ recolored leaves in one of the subinstances. The pseudo-code is given in Figures 9 and 10.

Theorem 6. *The minimum number of recolored leaves OPT can be computed in $O(4^{OPT} \cdot n)$ time.*

```
Algorithm MainBranching(T_x, C)
1  found = false; k = 0;
2  while (not found)
3      answer = Branching (T_x, C, k);
4      if (answer = false)
5      then k = k + 1
6      else found = true; return answer;
```

Fig. 9. Pseudo-code of the Algorithm MainBranching.

```
Algorithm Branching(T_x, C, k)
    Input: A leaf colored tree (T_x, C), rooted by node x, T_x = (V, E),
    V = {v_1, ··· , v_n}, C = {c_1, ··· , c_k}, n, k ∈ ℕ − {0}.
    Output: A leaf coloring C' with at most k recolored leaves that is
    convex colorable, or "false" if no such recoloring exists.
1   if (T_x, C) is convex colorable then return C
2   else
3       if (k = 0) then return "false"
4       else
5           Find a pair of crossing paths in (T_x, C), p_1 and p_2;
6           let Y be the leaves on p_1 and p_2;
7           foreach y ∈ Y
8               Define C' as follows: C'(y) = θ_y; (θ_y is a new color);
9               for all leaves x ≠ y: C'(x) = C(x);
10              br = Branching(T_x, C', k − 1);
11              if (br ≠ "false") then return br; (exit)
```

Fig. 10. Pseudo-code of the Algorithm Branching.

Proof. The branching algorithm will continue precisely to level OPT. The kth round of the main loop of MainBranching will cause in total at most $\sum_{j=0}^{k} 4^j < 2 \cdot 4^k$ calls to Branching. So, in total, less than $\sum_{k=0}^{OPT} 2 \cdot 4^k < 4^{OPT+1}$ calls to Branching will be done. Each such call takes $O(n)$ time (see Theorem 4 and Theorem 5). □

Fixed Parameter Tractable Problems (FPT)

A problem with a parameter k is called fixed parameter tractable (FPT) if it can be solved or decided by an algorithm within a running time $O(f(k) \cdot poly(n))$, for some function f. For fixed k, this is polynomial time. For more information on fixed parameter tractability, see [2].

Corollary 5. *The* CONVEX RECOLORING OF LEAF COLORED TREES *problem* $\in$ *FPT.*

Proof. Do the branching algorithm for k levels and see if a solution is found. The running time is $O(4^k \cdot n)$. □

References

1. R. Bar-Yehuda, I. Feldman, and D. Rawitz. Improved approximation algorithm for convex recoloring of trees. In *Proceedings Third Workshop on Approximation and Online Algorithms WAOA 2005*, 2005.
2. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1998.
3. S. Moran and S. Snir. Convex recolorings of strings and trees: Definitions, hardness results, and algorithms. In F. K. H. A. Dehne, A. López-Ortiz, and J.-R. Sack, editors, *Proceedings WADS 2005: 9th International Workshop on Algorithms and Data Structures*, pages 218–232. Springer Verlag, Lecture Notes in Computer Science, vol. 3608, 2005.
4. S. Moran and S. Snir. Efficient approximation of convex recolorings. In *Proceedings APPROX 2005: 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, published with Proceedings RANDOM 2005*, pages 192–208. Springer Verlag, Lecture Notes in Computer Science, vol. 3624, 2005.
5. S. Snir. *Computational Issues in Phylogenetic Reconstruction: Analytic Maximum Likelihood Solutions, and Convex Recoloring*. PhD thesis, Department of Computer Science, Technion, Haifa, Israel, 2004.