Theory Comput Syst (2008) 43: 3–18
DOI 10.1007/s00224-007-9069-7

Improved Approximation Algorithm for Convex Recoloring of Trees

Reuven Bar-Yehuda · Ido Feldman · Dror Rawitz

Published online: 18 October 2007

Abstract A pair (T, C) of a tree T and a coloring C is called a *colored tree*. Given a colored tree (T, C) any coloring C' of T is called a *recoloring* of T. Given a weight function on the vertices of the tree the *recoloring distance* of a recoloring is the total weight of recolored vertices. A coloring of a tree is *convex* if for any two vertices u and v that are colored by the same color c, every vertex on the path from u to v is also colored by c. In the *minimum convex recoloring problem* we are given a colored tree and a weight function and our goal is to find a convex recoloring of minimum recoloring distance.

The minimum convex recoloring problem naturally arises in the context of *phylogenetic trees*. Given a set of related species the goal of phylogenetic reconstruction is to construct a tree that would best describe the evolution of this set of species. In this context a convex coloring corresponds to *perfect phylogeny*. Since perfect phylogeny is not always possible the next best thing is to find a tree which is as close to convex as possible, or, in other words, a tree with minimum recoloring distance.

We present a $(2 + \varepsilon)$-approximation algorithm for the minimum convex recoloring problem, whose running time is $O(n^2 + n(1/\varepsilon)^2 4^{1/\varepsilon})$. This result improves the previously known 3-approximation algorithm for this NP-hard problem. We also present an algorithm for computing an optimal convex recoloring whose running time is $O(n^2 + n \cdot n^* \cdot \Delta^{n^*+1})$, where n^* is the number of colors that violate convexity in

R. Bar-Yehuda · I. Feldman
Department of Computer Science, Technion, Haifa 32000, Israel

R. Bar-Yehuda
e-mail: reuven@cs.technion.ac.il

I. Feldman
e-mail: idofeld@cs.technion.ac.il

D. Rawitz (✉)
Caesarea Rothschild Institute, University of Haifa, Haifa 31905, Israel
e-mail: rawitz@cri.haifa.ac.il

the input tree, and Δ is the maximum degree of vertices in the tree. The parameterized complexity of this algorithm is $O(n^2 + n \cdot k \cdot 2^k)$.

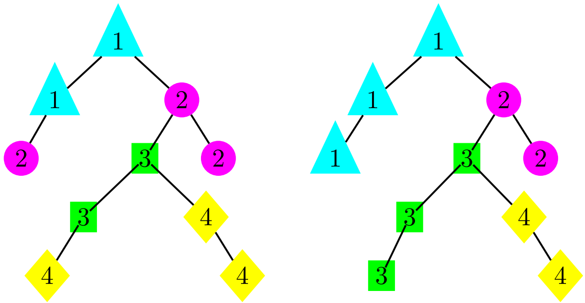
Keywords Approximation algorithms · Convex recoloring · Local ratio · Phylogenetic trees

1 Introduction

1.1 Problem Statement and Motivation

Given a tree $T = (V, E)$ with n vertices, a *coloring* of the tree is a function $C : V \to \mathcal{C}$, where $\mathcal{C}$ is a set of colors. We define $m \triangleq |\mathcal{C}|$. A pair (T, C) of a tree and a coloring is called a *colored tree*. A coloring C of a tree is *convex* if for every two vertices u and v such that $C(u) = C(v) = c$ the color of every vertex on the path from u to v is also c. That is, a coloring is convex if the set of vertices colored by c induces a (possibly empty) subtree for every color $c \in \mathcal{C}$. Examples of a non-convex coloring and a convex coloring are given in Fig. 1. Given a colored tree (T, C) any coloring C' of T is called a *recoloring* of T. A vertex u is *recolored* if $C(v) \neq C'(v)$. Given a non-negative weight function w on the vertices of T the *recoloring distance* of C' is the total weight of recolored vertices. For example, given the coloring in Fig. 1a and assuming unit weights, the recoloring cost of the coloring in Fig. 1b is 2. In the *minimum convex recoloring problem* we are given a colored tree (T, C) and a non-negative weight function w and our goal is to find a convex recoloring C' of minimum recoloring distance.

The minimum convex recoloring problem was first introduced by Moran and Snir [12], who showed that this problem arises in the context of *phylogenetic trees*. Given a set of related species the goal of phylogenetic reconstruction is to construct a tree that would best describe the evolution of this set of species. In such a phylogenetic tree the leaves represent the species, while internal vertices represent extinct species. A *character* is an attribute shared by the entire set of species represented by the tree. Such a character has different *states*, and each species is associated with one of these states. For example, the character may be as simple as the existence of wings, and the states are *wings*, and *no wings*. It is not hard to see that the states of



(a) Non-convex coloring (b) Convex coloring

Fig. 1 Transforming a non convex coloring into a convex coloring

a given character correspond to a set of colors, and that the states associated with the species correspond to a coloring of the tree. A natural biological constraint is that the tree does not contain *reverse* or *convergent* transitions with respect to every character. A reverse transition occurs when some species has a common character state with an old ancestor while its direct ancestor is associated with a different character state. In a convergent transition two species share a character state which is different from the character state of their least common ancestor. The absence of reverse and convergent transitions implies that the path in the tree connecting two species with some character state must contain only species with an identical character state. In other words, a character with respect to which there are no convergent and reverse transitions is a convex coloring of the tree. Hence, our goal is to construct a tree in which every character is a convex coloring. This problem is known as the *perfect phylogeny problem* (see, e.g., [6, 9–11]).

Since perfect phylogeny is not always possible the next best thing is to find a tree which is as close to convex as possible with respect to each character. However, the meaning of *close to convex* must be defined first. One possible measure of closeness is the *parsimony score* which is the number of mutated edges (i.e., edges whose endpoints have different states) summed over all characters [14, 15]. Another measure is the *phylogenetic number* [8] which is defined as the maximum number of connected components induced by a single state. In [12] Moran and Snir defined a natural distance from a phylogenetic tree to a convex one—the *recoloring distance*. We note that the parsimony score and the phylogenetic number do not specify a distance to an actual convex coloring of the given tree. Moreover, there are trees with large phylogenetic numbers and parsimony scores that can be made convex only by changing the color of one vertex, while other trees with small phylogenetic numbers that can become convex only by changing the color of a large number of vertices.

For more details about phylogenetic trees and other applications of the minimum convex recoloring problem we refer the reader to [12, 13].

1.2 Previous Results

The minimum convex recoloring problem was first defined by Moran and Snir [12]. They showed that the problem is NP-Hard even on strings (trees with two leaves) with unit weights. In addition, they presented a dynamic programming based algorithm for computing an optimal convex recoloring of trees. The running time of this algorithm is $O(n \cdot n^* \cdot \Delta^{n^*+2})$, where n^* is the number of colors that violate convexity in the input tree, and Δ is the maximum degree of vertices in the tree. Moran and Snir also show that a modification of their algorithm has a running time of $O(n^4 \cdot n^* \cdot \mathrm{Bell}(n^*))$, where $\mathrm{Bell}(m)$ is the number of unordered partitions of m elements to any number of non empty subsets. $\mathrm{Bell}(m)$ is asymptotically smaller than $(\frac{m}{\log m})^m$.

In a followup paper [13] Moran and Snir presented a 3-approximation algorithm based on the *local ratio technique* [1–5] whose running time is $O(n_c n^2)$, where n_c is the number of colors, and an $O(n_c n)$ time 2-approximation algorithm for strings.

1.3 Our Results

We obtain a polynomial time $(2 + \varepsilon)$-approximation algorithm for the minimum convex recoloring problem. Our algorithm depends on an accuracy parameter $k \geq 2$, where $k = \lceil \frac{2}{\varepsilon} \rceil + 1$, and consists of two phases. The first phase is a local ratio algorithm that manipulates the weights such that the original weighted colored tree is transformed into a partially colored tree with a special structure we call k-simple. The running time of this phase is $O(n^2)$. We show that this transformation has the property that any $(2 + \frac{2}{k-1})$-approximation constructed by the second phase for the k-simple tree is also a $(2 + \frac{2}{k-1})$-approximation for the original instance. The second phase is a dynamic programming algorithm that computes an optimal solution for the k-simple tree. The running time of this phase is $O(n^2 + nk^2 2^k)$. It follows that the approximation ratio of our algorithm is $2 + \frac{2}{k-1}$ and its running time is $O(n^2 + nk^2 2^k)$. For example, if we set $k = \log n/2 + 1$ we get a $(2 + 4/\log n)$-approximation algorithm whose time complexity is $O(n^2)$.

Our dynamic programming algorithm for computing an optimal convex recoloring (for general colored trees) is faster than the best previously known algorithm presented in [12], since the running time of our algorithm (in terms of n^* and Δ) is $O(n^2 + n \cdot n^* \cdot \Delta^{n^*+1})$. We also show that the parameterized complexity of this algorithm is $O(n^2 + n \cdot k \cdot 2^k)$.

1.4 Overview

The remainder of the paper is organized as follows. Section 2 contains most of our definitions and notation. Our dynamic programming algorithm is given in Sect. 3, and in Sect. 4 we study its parameterized complexity. In Sect. 5 we define k-simple trees and analyze the dynamic programming algorithm for the special case of k-simple trees. Finally, in Sect. 6 we present our $(2 + \frac{2}{k-1})$-approximation algorithm.

2 Preliminaries

In this section we focus on two main issues. First, we define the notion of *partial colorings*, and show that, given a weighted colored tree, it is sufficient to look for a convex partial recoloring instead of a convex recoloring. Next, we examine the structure of an optimal convex partial recoloring.

2.1 Partial Colorings

A *partial coloring* of a tree T is function $C : V \to \mathcal{C} \cup \{\varnothing\}$, where $\varnothing$ stands for *no color*. That is, if $C(v) = \varnothing$ then v is assumed to be *uncolored*. A pair (T, C) of a tree and a partial coloring is called a *partially colored tree*. A partial coloring C is called convex if it can be extended to a (total) convex coloring.

Observation 1 *A convex partial coloring can be completed in $O(n^2)$ time.*

Proof Let (T, C) be a partially colored tree, where C is convex. We show how to compute a total convex coloring C' for T, where $C'(u) = C(u)$ if $C(u) \neq \varnothing$. Let u be a vertex such that $C(u) = \varnothing$. Observe that if there are two vertices v_1 and v_2 such that u is on the path between v_1 and v_2 and $C(v_1) = C(v_2) = c \neq \varnothing$, then the color of u in any convex completion of C must be c. Hence, in such a case we assign $C'(u) = c$. Next, we color the remaining uncolored vertices. Let u be an uncolored vertex and let $V_u \subseteq V$ be the set of colored vertices v such that the path from u to v is uncolored, and let $C'(u) = \min_{\prec}\{C(v) : v \in V_u\}$, where $\prec$ is some (arbitrary) complete order on the colors.

Since $C'(u)$ can be obtained using BFS, for any uncolored vertex u, C can be completed in $O(n^2)$ time. □

We consider an extended version of the minimum convex recoloring problem in which both the input and output colorings may be partial. That is, we are given a tree $T = (V, E)$, and a partial coloring C of the tree and our goal is to compute a convex partial recoloring C'. We say that a recoloring C' of C *discolors* a vertex u if $C(u) \neq \varnothing$ and $C'(u) \neq C(u)$. That is, C' discolors u if it changes or removes its original color. Given a non-negative weight function w on the vertices of T the *recoloring distance* of C' is the total weight of discolored vertices. (Informally, this means that we pay for removing a color, but not for applying a color.) Hence, we may assume without loss of generality that if $w(u) = 0$ then $C(u) = \varnothing$, and vice versa. Observe that since coloring uncolored vertices costs nothing, turning a convex partial coloring into a convex coloring incurs no cost.

Observation 2 *The weight of an optimal convex partial recoloring is equal to the weight of an optimal convex recoloring.*

Given a partially colored tree (T, C), a vertex set $X \subseteq V$ is called a *cover* if there is a convex partial recoloring C' such that X is the set of vertices that are discolored by C'. For a set of vertices $U \subseteq V$ and a weight function w we define $w(U) = \sum_{u \in U} w(u)$. Hence, the cost of a cover X is defined as $w(X)$, and the *recoloring distance* of a corresponding convex partial coloring C' is $w(C') = w(X)$.

2.2 Form of an Optimal Solution

Given a subset of vertices $U \subseteq V$ we denote the set of colors that are used to color U by $C(U)$, i.e., $C(U) = \{c \in \mathcal{C} : C(u) = c \text{ and } u \in U\}$. Notice that $C(U)$ does not include $\varnothing$. Given a subtree T' of T we denote the set of colors used in T' by $C(T')$, i.e., $C(T') = C(V(T'))$.

Observation 3 *Let* (T, C, w) *be a weighted colored tree. Then there exists an optimal convex* (*partial*) *recoloring* C' *such that* $C'(T) \subseteq C(T)$.

Proof Let C' be an optimal convex recoloring that uses a minimal number of colors not from $C(T)$. We show that C' must satisfy $C'(T) \subseteq C(T)$. Assume that there exists $c \in C'(T) \setminus C(T)$. We show how to transform C' into an optimal convex recoloring C'' that uses the colors in $C'(T) \setminus \{c\}$ in contradiction to the minimality of C'. Let d be a color such that $d \in C'(T) \setminus \{c\}$ and there is an edge (v, u) such that $C'(v) = d$ and $C'(u) = c$. In this case we define $C''(u) = d$ for every u such that $C'(u) = c$. C'' is clearly convex. Also, observe that if $C'(v) = c$, then the recoloring

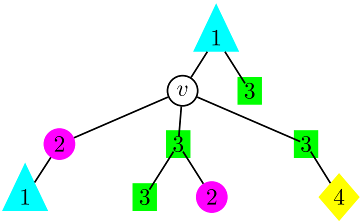


Fig. 2 $S(v)=\{1,2,3\}$, $\sigma_v=3$, and $\pi_v=\textsc{sep}_v(1)\cdot\textsc{sep}_v(2)\cdot\textsc{sep}_v(3)=2\cdot 2\cdot 3=12$

cost of v is counted in $w(C')$. Hence, $w(C'')\le w(C')$. If such a color d does not exist, then we simply define $C''(u)=d'$ for every u, where $d'\in C(T)$. □

Next, we show that there exists an optimal partial recoloring in which each vertex has a limited choice of colors, and a vertex colored by $\varnothing$ is not located on the path between two vertices that are colored by the same color.

Given a tree T we denote by $T\setminus v$ the set of subtrees obtained when v is removed from T. Given a colored tree (T,C), we say that v *separates* c (with respect to C) if there are at least two subtrees in $T\setminus v$ that contain a vertex u such that $C(u)=c$. The *separation number* $\textsc{sep}_v(c)$ of a vertex v with respect to a color $c\neq\varnothing$ is defined as the number of subtrees in $T\setminus v$ that contain a vertex u such that $C(u)=c$. Let $S(v)$ be the set of colors that are separated by v, i.e., let $S(v)$ be the set of colors for which the separation number is greater than 1. That is, $S(v)=\{c:\textsc{sep}_v(c)>1\}$. We define $\sigma_v=|S(v)|$ and $\pi_v=\prod_{c\in S(v)}\textsc{sep}_v(c)$. (If $\sigma_v=0$ then $\pi_v=1$.) An example is given in Fig. 2.

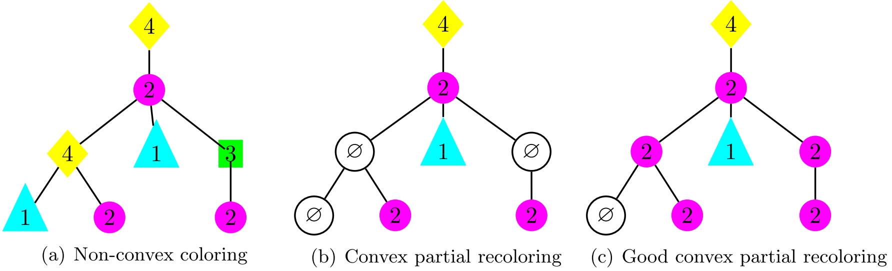
Definition 1 Given a colored tree (T,C) we define the *color set* of v by $G(v)\overset{\triangle}{=}S(v)\cup\{C(v),\varnothing\}$. (Recall that $C(v)$ may be $\varnothing$.) A partial recoloring C' is called *good* if (1) $C'(v)\in G(v)$ for every $v\in V$, and (2) if $C'(v)=\varnothing$, then v does not separate any color c with respect to C'.

We would like to emphasize that $\textsc{sep}_v(c)$ and the related definitions depend on a specific coloring of the given tree. Specifically, the coloring involved is almost always the original coloring of the tree. Hence, for reasons of simplicity, this coloring does not appear in the notation. In cases where the coloring involved D is not the original coloring of the tree, we use the term *with respect to D*.

In the next lemma we show that there exists a good optimal convex partial recoloring. Hence, we can concentrate on finding good partial recolorings, and, in particular, it is enough to design an algorithm that computes an optimal good partial recoloring in order to solve the problem.

Lemma 1 *Let (T,C,w) be a weighted colored tree. Then there exists a good optimal convex partial recoloring C'.*

Proof Let C' be an optimal convex recoloring, and let X be the corresponding cover. We construct a good optimal partial recoloring C'' that corresponds to the same cover X. First, we set $C''(v)=C(v)$ for every $v\notin X$. Next, we discolor the vertices in



(a) Non-convex coloring (b) Convex partial recoloring (c) Good convex partial recoloring

Fig. 3 Example of a good convex partial recoloring

X. Observe that, with respect to C', every $v \in X$ separates at most 1 color (since X is a cover). Hence, for every $v \in X$ that separates a color c, we define $C''(v) = C'(v) = c$, and otherwise, we define $C''(v) = \varnothing$. (See Fig. 3 for an illustration). Clearly, C'' is good. Moreover, since we only changed vertices in X, we get that $w(C'') = w(C')$. Also, C'' is convex, since it can be extended to C'. □

Lemma 2 *Let $v \in V$ be a vertex and $T' \in T \setminus v$ be a subtree of the weighted colored tree (T, C, w). If C' is a good partial recoloring, then $C'(T') \subseteq C(T') \cup \{\varnothing\}$.*

Proof Consider a vertex u in the subtree T'. If $C'(u) = C(u)$ or $C'(u) = \varnothing$, then we are done. Otherwise, it follows that $C'(u) \in S(u)$ by Definition 1. Hence, u separates $C'(u)$, which means that $C'(u) \in C(T')$. □

3 Dynamic Programming Algorithm

In this section we describe a dynamic programming algorithm for computing an optimal convex partial recoloring whose running time is $O(n^2 + \sum_{v \in V} \sigma_v \cdot (\deg(v) + \pi_v))$, where $\deg(v)$ is the degree of the vertex v. This expression becomes polynomial in n and exponential in k for the special case of k-simple trees (defined in the next section).

Throughout this section we treat the input tree T as a rooted tree. This is done by choosing an arbitrary root s. We denote the ith child of v by v_i, and the parent of v by v_0. Observe that v has $\deg(v) - 1$ children, if $v \neq s$, and that the root s has $\deg(s)$ children. As before the set of subtrees obtained by the removal of V is denoted by $T \setminus v$. We denote the subtree of the ith child by $T_i(v)$, and by $T(v)$ the tree rooted at v. We also denote by $T_0(v)$ the subtree obtained by removing $T(v)$ from T. See Fig. 4.

Let C' be a convex good partial recoloring of T, and consider a vertex v such that $C'(v) = c$. If $c \neq \varnothing$ then C' induces a partition of $\mathcal{C} \setminus \{c\}$. A color $d \neq c$ that is used in $T_i(v)$ cannot be used in $T_j(v)$ where $i \neq j$. If $c = \varnothing$ then C' induces a partition on $\mathcal{C}$. In both cases, v partitions the color set $\mathcal{C} \setminus \{c\}$ into $\deg(v)$ mutually disjoint color sets. (Recall that $\varnothing \notin \mathcal{C}$.) If $c \neq \varnothing$ then c may be used to color vertices in more than one subtree from $T \setminus v$. Obviously, the use of this color is possible only if the vertices colored by it form a subtree.

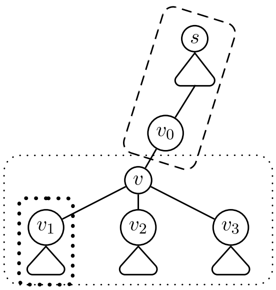


Fig. 4 $T_0(v)$ is marked by the *dashed line*, $T_1(v)$ is marked by the *thick dotted line*, and $T(v)$ is marked by the *thin dotted line*

Definition 2 Let (T, C) be a colored tree, let $c \in \mathcal{C} \cup \{\varnothing\}$, and let $v \in V$ be a vertex with r children. We say that $(D_0, \dots, D_r)$ is a *good partition with respect to* v *and* c if $D_i \subseteq C(T_i(v)) \setminus \{c\}$ for $i \in \{0, \dots, r\}$, and $(D_0, \dots, D_r)$ is a partition of $\bigcup_{i=0}^{r} C(T_i(v)) \setminus \{c\}$.

Let $\text{GOOD}(v, c)$ denote the set of all good partitions with respect to v and c.

Observation 4 *Let v be a vertex, and let c be a color. Then,* $|\text{GOOD}(v, c)| \le \pi_v$.

Proof Consider a color $d \neq c$. Clearly, d can be assigned to D_i only if $d \in C(T_i(v))$. Hence, d can be assigned to exactly $\text{SEP}_v(d)$ subtrees in $T \setminus v$. Hence, the number of possible combinations is $\prod_{d \neq c} \text{SEP}_v(d) \le \pi_v$. □

Let v be a vertex, and let c be a color. Also, let C' be a convex partial recoloring that is consistent with some good partition $(D_0, \dots, D_r)$ with respect to v and c. Then, if v is colored by c, the colors in D_0 cannot be used in $T(v)$. We refer to these colors as *forbidden* with respect to v.

Definition 3 Let (T, C) be a colored tree, and let $v \in V$ be a vertex with r children. Define,

$$\text{FORB}(v) = \{D : \exists c \in G(v)\ \exists (D_1, \dots, D_r),\ (D, D_1, \dots, D_r) \in \text{GOOD}(v, c)\}.$$

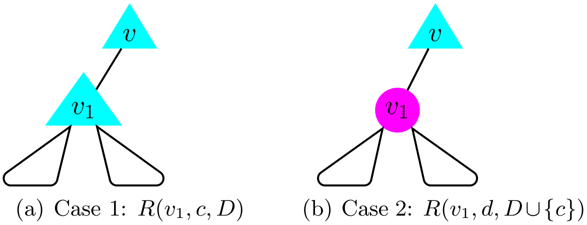
Thus, $D \in \text{FORB}(v)$ if there is a good partition, where the color set D may be used only in $T_0(v)$.

Lemma 3 $|\text{FORB}(v)| \le 2^{\sigma_v + 1}$ *for every* $v \in V$.

Proof Let v be a vertex with r children. Observe that if $d \in C(T_i(v)) \cap C(T_j(v))$, where $0 \le i < j \le r$, then v separates d. Thus, if v does not separate d and $d \neq C(v)$, then for any color $c \neq d$ either $d \in D_0$ for every $(D_0, \dots, D_r) \in \text{GOOD}(v, c)$, or $d \notin D_0$ for every $(D_0, \dots, D_r) \in \text{GOOD}(v, c)$. If follows that if $D, D' \in \text{FORB}(v)$ and $d \in D \setminus D'$, then either $d \in S(v)$ or $d = C(v)$. Therefore, $|\text{FORB}(v)| \le 2^{|S(v)|+1} = 2^{\sigma_v + 1}$. □

We now turn to design a dynamic programming algorithm for computing an optimal good convex partial recoloring of a given colored tree. We construct an optimal solution bottom-up, by storing intermediate values on the vertices of the tree.

Fig. 5 Combining a recoloring of $T_1(v)$ with a recoloring of v. Let c be a possible color of v, and let D be the set of forbidden colors for $T(v)$. The colors c and d correspond to the *triangle* and the *circle*, respectively



(a) Case 1: $R(v_1, c, D)$ (b) Case 2: $R(v_1, d, D \cup \{c\})$

Definition 4 Let v be a vertex in T, let $c \in \mathcal{C} \cup \{\varnothing\}$, and let $D \in \text{FORB}(v)$.

- A good convex recoloring C' of $T(v)$ is a (v, D)-*coloring* if it is a recoloring in which the colors in D are not used to color $T(v)$, i.e., it is a recoloring of $T(v)$ such that $C'(T(v)) \cap D = \emptyset$. $\text{OPT}(v, D)$ denotes the weight of an optimal (v, D)-coloring.
- A good convex recoloring C' of $T(v)$ is a (v, c, D)-*coloring* if it is a recoloring in which the colors in D are not used to color $T(v)$ and v is colored by c, i.e., it is a (v, D)-coloring of $T(v)$ such that $C'(v) = c$. $\text{OPT}(v, c, D)$ denotes the weight of an optimal (v, c, D)-coloring.

Observe that, for a tree T with root s, the weight of an optimal recoloring of the whole tree is $\text{OPT}(s, \emptyset)$.

We compute OPT recursively using the following rules:

1. $R(v, D) = \min_{c \in G(v) \setminus D} R(v, c, D)$.
2. If $C(v) = c$ then

$$R(v, c, D) = \min_{(D, D_1, \ldots, D_r) \in \text{GOOD}(v,c)} \left\{ \sum_{i=1}^{r} R'(v_i, c, \mathcal{C} \setminus D_i) \right\}$$

otherwise

$$R(v, c, D) = w(v) + \min_{(D, D_1, \ldots, D_r) \in \text{GOOD}(v,c)} \left\{ \sum_{i=1}^{r} R'(v_i, c, \mathcal{C} \setminus D_i) \right\},$$

where $R'(v, c, D) \overset{\triangle}{=} \min\{R(v, D \cup \{c\}), R(v, c, D)\}$.

where v is a vertex with r children, $D \in \text{FORB}(v)$, and $c \in G(v) \setminus D$. (Recall that v_i is the ith child of v for every $i \in \{1, \ldots, r\}$.)

In Rule 1 we go through all (v, c, D)-colorings and find the best one that colors the subtree $T(v)$. In Rule 2 we try to glue colorings of the subtrees $T_1(v), \ldots, T_r(v)$ to a coloring of v. Notice that, for every $i \in \{1, \ldots, r\}$, if v is colored by c then either c is not used in $T_i(v)$, or it is used to color v_i and maybe some other vertices in $T_i(v)$. (See Fig. 5.)

Theorem 1 *Let v be a vertex with r children, let $D \in \text{FORB}(v)$ be a set of colors, and let $c \in G(v) \setminus D$. Then, $R(v, c, D) = \text{OPT}(v, c, D)$, and $R(v, D) = \text{OPT}(v, D)$.*

Proof We prove the theorem by induction on the tree. In the induction base v is a leaf, and in this case

$$R(v,c,D)=\begin{cases} w(v) & C(v)\neq c,\\ 0 & \text{otherwise,}\end{cases}$$

and

$$R(v,D)=\begin{cases} w(v) & C(v)\in D\\ 0 & \text{otherwise,}\end{cases}$$

as required.

Next, for the inductive step we assume that $R(v_i,D)=\text{OPT}(v_i,D)$ and $R(v_i,c,D)=\text{OPT}(v_i,c,D)$ for every $i\in\{1,\ldots,r\}$.

We first prove that $R(v,c,D)\leq\text{OPT}(v,c,D)$. Let C' be an optimal (v,c,D)-coloring. That is, C' is a good convex recoloring of $T(v)$ in which the colors in D are not used to color $T(v)$ and v is colored by c. For every $i\in\{0,\ldots,r\}$, let D'_i be the set of colors that are used by C' to color $T_i(v)$. That is, $D'_i=C'(T_i(v))\setminus\{c\}$ for every $i\in\{0,\ldots,r\}$. Since C' is convex and $C'(v)=c$, we must have that for $i\neq j$, $D'_i\cap D'_j=\emptyset$. By Lemma 2, it follows that $D'_i\subseteq C(T_i(v))\setminus\{c\}$ for every i. Thus, $\bigcup_{i=0}^{r}D'_i\subseteq\bigcup_{i=0}^{r}C(T_i(v))\setminus\{c\}$. If there exists a color d such that $d\notin\bigcup_{i=0}^{r}D'_i$ and $d\in\bigcup_{i=0}^{r}C(T_i(v))\setminus\{c\}$ we add it to an arbitrary set D'_i satisfying $d\in C(T_i(v))$. It follows that $(D,D'_1,\ldots,D'_r)\in\text{GOOD}(v,c)$. This means that the good partition $(D,D'_1,\ldots,D'_r)$ is considered by Rule 2. Since $\text{OPT}(v_i,C'(v_i),D'_i)=R(v_i,C'(v_i),D'_i)$ by the inductive hypothesis, it follows that $R(v,c,D)\leq\text{OPT}(v,c,D)$.

Next we show that $\text{OPT}(v,c,D)\leq R(v,c,D)$. Let $(D,D_1,\ldots,D_r)$ be a good partition which minimizes the right hand side of Rule 2. Let C'_i be the recoloring of $T_i(v)$ whose weight is $R'(v_i,c,\mathcal{C}\setminus D_i)$ for $i\in\{1,\ldots,r\}$. We obtain a recoloring of $T(v)$ as follows: $C'(v)=c$ and $C'(u)=C'_i(u)$ for every $u\in T_i(v)$ and every $i\in\{1,\ldots,r\}$. By its construction C' is a good convex recoloring of $T(v)$ that does not use colors from D and such that $C'(v)=c$. Hence, there exists a good convex recoloring C' of $T(v)$ whose weight is $R(v,c,D)$. Therefore, $\text{OPT}(v,c,D)\leq R(v,c,D)$.

It remains to show that $R(v,D)=\text{OPT}(v,D)$. This follows from

$$\text{OPT}(v,D)=\min_{c\in G(v)\setminus D}\text{OPT}(v,c,D)=\min_{c\in G(v)\setminus D}R(v,c,D)=R(v,D)$$

and we are done. □

The number of entries computed by the dynamic programming algorithm is: $O(\sum_{v\in V}|G(v)|\cdot|\text{FORB}(v)|)$. An additional $O(nm)$ space is needed for storing $G(v)$ for every vertex v. Hence, the space complexity of the algorithm is $O(nm+\sum_{v\in V}\sigma_v\cdot 2^{\sigma_v})$.

The running time that is required to compute an entry of the form $R(v,D)$ is $O(|G(v)|)$. For each entry of the form $R(v,c,D)$ we need to go through all possibilities of good partitions of the form $(D,D_1,\ldots,D_r)$, and for each such good partition we perform $O(\deg(v))$ operations. Observe that for every v and c we actually go through every good partition in $\text{GOOD}(v,c)$ for computing the values of

$R(v,c,D)$ for all $D \in \textsc{forb}(v)$. Since every good partition is visited exactly once, we invest $O(|\textsc{good}(v,c)| \cdot \deg(v))$ operations on every pair of vertex v and color c. Hence, by Observation 4, this brings us to $O(\sum_{v\in V} \sigma_v \cdot \pi_v \cdot \deg(v))$. An additional $O(n^2)$ is needed to compute $G(v)$ for every v. Hence, the total running time is $O(n^2 + \sum_{v\in V} \sigma_v \cdot \pi_v \cdot \deg(v))$.

Note that the computation of $R(v,c,D)$ can be modified also to output a corresponding solution. This can be done by keeping track of which option was taken in both rules. Afterwards we can reconstruct the optimal recoloring in a top down manner.

Next, we explain how to improve the running time of the algorithm. Let v be a vertex with r children, and let $c \in G(v)$ be a color. Also, let $\mathcal{D} = (D_0,\dots,D_r), \mathcal{D}' = (D'_0,\dots,D'_r) \in \textsc{good}(v,c)$. We define $\Delta(\mathcal{D},\mathcal{D}') \stackrel{\triangle}{=} \{i : D_i \neq D'_i\}$ and $R'(v,c,\mathcal{D}) \stackrel{\triangle}{=} \sum_{i=1}^{r} R'(v_i, c, \mathcal{C} \setminus D_i)$. Observe that

$$R'(v,c,\mathcal{D}') - R'(v,c,\mathcal{D}) = \sum_{i\in\Delta(\mathcal{D},\mathcal{D}')} R'(v_i,c,\mathcal{C}\setminus D'_i) - R'(v_i,c,\mathcal{C}\setminus D_i).$$

Hence, if $\Delta(\mathcal{D},\mathcal{D}') = \{i,j\}$ and we know the value of $R'(v,c,\mathcal{D})$, then we can compute $R'(v,c,\mathcal{D}')$ by using the equation

$$\begin{aligned} R'(v,c,\mathcal{D}') = R'(v,c,\mathcal{D}) &+ [R'(v_i,c,\mathcal{C}\setminus D'_i) - R'(v_i,c,\mathcal{C}\setminus D_i)] \\ &+ [R'(v_j,c,\mathcal{C}\setminus D'_j) - R'(v_j,c,\mathcal{C}\setminus D_j)] \end{aligned}$$

in $O(1)$ time. If $\Delta(\mathcal{D},\mathcal{D}') = \{0,j\}$ then we can compute $R(v,c,\mathcal{D}')$ by using the equation

$$R'(v,c,\mathcal{D}') = R'(v,c,\mathcal{D}) + [R'(v_j,c,\mathcal{C}\setminus D'_j) - R'(v_j,c,\mathcal{C}\setminus D_j)]$$

also in $O(1)$ time.

For a vertex $v \in V$ and a color $c \in G(v)$ we say that an ordering $\mathcal{D}_1,\dots,\mathcal{D}_{|\textsc{good}(v,c)|}$ of the set $\textsc{good}(v,c)$ is a *close order* if $\Delta(\mathcal{D}_i,\mathcal{D}_{i+1}) = 2$ for every i. We now describe how to construct a close order of $\textsc{good}(v,c)$ for any $v \in V$ with r children and a color c. Let $\mathcal{C} = \{c_1,\dots,c_m\}$. A good partition $(D_0,\dots,D_r) \in \textsc{good}(v,c)$ can be described by a word $\sigma_1 \cdots \sigma_m$ such that $\sigma_i = j$ if $c_i \in D_j$. (Notice that $\sigma_i = j$ is possible only if $c_i \in C(T_j(v))$.) An order of this set of words in which the hamming distance between every two consecutive words is 1 defines a close order on $\textsc{good}(v,c)$. Such a close order can be obtained using a generalization of Gray code for non-binary alphabets (see, e.g., in [7]).

We examine the time complexity for computing the value of $R(v,c,D)$ for every vertex v, $D \in \textsc{forb}(v)$, and $c \in G(v) \setminus D$ using Rule 2. Computing the value corresponding to the first member in the close order of $\textsc{good}(v,c)$ takes $O(\deg(v))$ time. For any other member the computation takes $O(1)$. (Notice that the identification of D can also be done in $O(1)$.) Therefore, the total time complexity for computing all entries of the form $R(v,c,D)$ for a vertex v is

$$O\biggl(\sum_{c\in G(v)} (\deg(v) + |\textsc{good}(v,c)|)\biggr) = O\biggl(\sum_{c\in G(v)} (\deg(v) + \pi_v)\biggr).$$

It follows that the total running time is:

$$O\left(n^2 + \sum_{v \in V} \sigma_v \cdot (\deg(v) + \pi_v)\right).$$

Let n^* be the number of colors that violate convexity in the input tree. There are σ_v colors that are separated by a vertex v. At least $\sigma_v - 1$ of them must be colors that violate convexity in the input tree. Hence, $\sigma_v - 1 \leq n^*$ for every v. It follows that the running time of the algorithm can be bounded by $O(n^2 + n \cdot n^* \cdot \Delta^{n^*+1})$, where Δ is the maximum degree of vertices in the tree.

4 Fixed Parameter Tractability

As in [12] we consider the parameterized complexity of the unweighted convex recoloring problem. Given a colored tree (T, C) we denote by k the recoloring distance of an optimal convex recoloring. A fixed-parameter algorithm for the convex recoloring problem is an algorithm that computes an optimal convex recoloring and whose running time is $n^{O(1)} \cdot f(k)$, where f is a function that depends on k (and not on n). In this section we show that our dynamic programming algorithm is in fact an $O(n^2 + n \cdot k \cdot 2^k)$ time fixed parameter algorithm.

As we have seen, the total running time of the dynamic programming algorithm is $O(n^2 + \sum_{v \in V} \sigma_v \cdot (\deg(v) + \pi_v))$. In the sequel we show that $\sigma_v \leq k + 1$ and $\pi_v \leq \deg(v) \cdot 2^k$ for every vertex v. Hence, the running time is bounded by

$$O\left(n^2 + \sum_{v \in V} (k+1) \cdot (\deg(v) + \deg(v) \cdot 2^k)\right) = O(n^2 + n \cdot k \cdot 2^k).$$

Consider a colored tree (T, C). We show that $\sigma_v \leq k + 1$ for every vertex v. Recall that $S(v)$ contains the σ_v colors that are separated by v. That is, $S(v)$ contains the colors that appear in more than one subtree in $T \setminus v$. Let C' be a convex recoloring such that $C'(v) = c_0$. In this case, C' must recolor at least one vertex colored by c in at least one subtree of $T \setminus v$, for every $c \in S(v) \setminus \{c_0\}$. Hence, $\sigma_v - 1 \leq k$.

Next, we show that $\pi_v \leq \deg(v) \cdot 2^k$ for every vertex v. Using the notation of the previous paragraph, C' must recolor the vertices colored by c in every subtree of $T \setminus v$, but maybe one. Hence, c causes at least $\text{SEP}_v(c) - 1$ recolorings. This argument holds for every color $c \in S(v) \setminus \{c_0\}$. Therefore,

$$\sum_{c \in S(v) \setminus \{c_0\}} (\text{SEP}_v(c) - 1) \leq k.$$

By the following observation we get that $\prod_{c \in S(v) \setminus \{c_0\}} \text{SEP}_v(c) \leq 2^k$.

Observation 5 *Let $\alpha_1, \ldots, \alpha_m \in \mathbb{N}$ and $\sum_i \alpha_i \leq k$. Then, $\prod_i (\alpha_i + 1) \leq 2^k$.*

It follows that

$$\pi_v = \prod_{c\in S(v)} \textsc{sep}_v(c) \le \textsc{sep}_v(c_0)\cdot \prod_{c\in S(v)\setminus\{c_0\}} \textsc{sep}_v(c) \le \deg(v)\cdot 2^k$$

as required.

5 Simple Trees

In this section we define the notion of a *k-simple* tree. Then, we show that the running time of the dynamic programming algorithm from Sect. 3 amounts to $O(n^2+n\cdot k^2 2^k)$ in the special case of k-simple trees.

Let (T,C) be a colored tree and $u\in V$ be a vertex. We say that u is a *(t,d)-separator* if there are t different colors $c_1,\dots,c_t$ such that for $1\le i\le t$, $\textsc{sep}_u(c_i)\ge d$. Observe that in this case $\pi_v \ge d^t$. Also, notice that if v is a (t,d)-separator with at least d neighbors, then for every $c_i\in\{c_1,\dots,c_t\}$ there are d vertices $u_1^i,\dots,u_d^i$ on d different components of $T\setminus v$ such that $C(u_j^i)=c_i$ and $w(u_j^i)>0$ for every $1\le j\le d$. We refer to such set of $t\cdot d$ vertices as a *(t,d)-separating witness of v*.

Definition 5 Let (T,C) be a colored tree, and define SEP $\triangleq \{(2,k),(3,4),(4,3),(k,2)\}$, where $k\ge 2$. We say that the colored tree is *k-simple* if v is not a (t,d)-separator for every $v\in V$ and $(t,d)\in$ SEP.

Observation 6 *Let (T,C) be k-simple for $k\ge 2$. Then, $\sigma_v<k$ for every $v\in V$.*

Consider a vertex v in a k-simple tree. Since $\sigma_v<k$, v separates at most $k-1$ colors, and therefore $|G(v)|\le k+1$.

Lemma 4 *Let (T,C) be k-simple for $k\ge 2$. Then, $\pi_v=O(\deg(v)\cdot k\cdot 2^k)$ for every $v\in V$.*

Proof Let $\mathcal{C}=\{c_1,\dots,c_m\}$, and consider a vertex $v\in V$. Without loss of generality we assume that $\textsc{sep}_v(c_i)\ge \textsc{sep}_v(c_{i+1})$ for every i. We show that the following conditions hold: (1) $\textsc{sep}_v(c_1)\le\deg(v)$, (2) $\textsc{sep}_v(c_2)\le k-1$, (3) $\textsc{sep}_v(c_3)\le 3$, (4) $\textsc{sep}_v(c_4)\le 2$, and (5) $\textsc{sep}_v(c_i)\le 1$ for every $i\ge k$. Hence, $\pi_v\le\deg(v)\cdot(k-1)\cdot 3\cdot 2^{\sigma_v-3}=O(\deg(v)\cdot k\cdot 2^k)$.

First, $\textsc{sep}_v(c_1)\le|T\setminus v|=\deg(v)$. Also, if $\textsc{sep}_v(c_2)\ge k$ then v is a $(2,k)$-separator; if $\textsc{sep}_v(c_3)\ge 4$ then v is a $(3,4)$-separator; if $\textsc{sep}_v(c_4)\ge 3$ then v is a $(4,3)$-separator; and if $\textsc{sep}_v(c_k)\ge 2$ then v is a $(k,2)$-separator. All in contradiction to the fact that (T,C) is k-simple. □

Now we analyze the running time of the dynamic programming algorithm for the special case of k-simple trees. Since $\sigma_v<k$ and $\pi_v=O(\deg(v)\cdot k2^k)$ for every v,

the total running time is

$$O\left(n^2 + \sum_{v \in V} k \cdot \deg(v) \cdot k2^k\right) = O(n^2 + n \cdot k^2 2^k).$$

The number of triplets of the form (v, c, D) computed by the dynamic programming algorithm is $O(nm + \sum_{v \in V} k \cdot 2^k) = O(nm + n \cdot k2^k)$.

6 $(2 + \varepsilon)$-approximation Algorithm

In this section we develop an algorithm that given a colored tree and an accuracy parameter k, computes a $(2 + \frac{2}{k-1})$-approximate convex partial recoloring. The running time of the algorithm is $O(n^2 + n \cdot k^2 \cdot 2^k)$.

The algorithm consists of two phases. In the first phase we use the local ratio technique. We manipulate the weights such that the original weighted colored tree is transformed into a k-simple tree. We show that this transformation has the property that any $(2 + \frac{2}{k-1})$-approximation constructed by the second phase for the k-simple tree is also a $(2 + \frac{2}{k-1})$-approximation for the original instance. The running time of this phase is $O(n^2)$. In the second phase of the algorithm we use our dynamic programming algorithm to compute an optimal convex partial recoloring for the k-simple tree. The running time of this phase is $O(n^2 + nk^2 2^k)$. It follows that the approximation ratio of the our algorithm is $2 + \frac{2}{k-1}$ and its running time is $O(n^2 + nk^2 2^k)$. We note that if we set $k = \frac{\log n}{2} + 1$ the approximation guarantee is $(2 + \frac{4}{\log n})$ and the time complexity is $O(n^2)$.

The local ratio technique [1–4] is based on the Local Ratio Theorem, which applies to optimization problems of the following type. The input is a non-negative weight vector $w \in \mathbb{R}^n$ and a set of feasibility constraints $\mathcal{F}$. The problem is to find a solution vector $x \in \mathbb{R}^n$ that minimizes (or maximizes) the inner product $w \cdot x$ subject to the constraints $\mathcal{F}$.

Theorem 2 (Local Ratio [3]) *Let $\mathcal{F}$ be a set of constraints and let w, w_1, and w_2 be weight vectors such that $w = w_1 + w_2$. Then, if x is r-approximate for both $(\mathcal{F}, w_1)$ and $(\mathcal{F}, w_2)$, for some r, then x is also r-approximate for $(\mathcal{F}, w)$.*

Algorithm **CR-LR** is our local ratio approximation algorithm. It uses our dynamic programming algorithm which is referred to as Algorithm **CR-DP**. We assume that Algorithm **CR-DP** returns a cover. Apart from a weighted colored tree, the input to our algorithm includes an accuracy parameter k. As we shall see this algorithm computes $(2 + \frac{2}{k-1})$-approximate solutions, and its running time is polynomial in n and exponential in k.

Algorithm 1: CR-LR(T, C, w, k)

```
if $(T, C)$ is $k$-simple then
    Return CR-DP($T, w$)
else
    Find $v \in V$ and $(t, d) \in \mathrm{SEP}$ such that $v$ is a $(t, d)$-separator
    Find a $(t, d)$-separating witness $U$ of $v$
    Let $\varepsilon = \min_{u \in U} w(u)$
    Define $w_1(u) = \begin{cases} \varepsilon & u \in U, \\ 0 & \text{otherwise} \end{cases}$
    $X \leftarrow$ CR-LR($T, C, w - w_1, k$)
    $Z \leftarrow \{u : (w - w_1)(u) = 0\}$
    Return $X \cup Z$
end if
```

We first analyze the time complexity of the algorithm. Observe that given a vertex v, checking whether v is a (t, d)-separator, where $(t, d) \in \mathrm{SEP}$, can be done in linear time. Also, observe that if a vertex v is not a (t, d)-separator at some iteration of the algorithm, then v cannot be a (t, d)-separator at any subsequent iteration. Since in each weight subtraction the weight of at least one vertex becomes zero, after no more than n subtractions there are no (t, d)-separators left in the given tree. Hence, the local ratio phase of the algorithm can be implemented to run in $O(n^2)$ time. Moreover, since the input to the dynamic programming algorithm is a k-simple tree, the total running time is $O(n^2 + n \cdot k^2 \cdot 2^k)$.

It remains to show that the solution returned is $(2 + \frac{2}{k-1})$-approximate. We prove this by induction on the recursion. Our proof replies on the fact that a zero-weight vertex can be treated as an uncolored vertex. (Recall that $C(v) = \varnothing$ is equivalent to $w(v) = 0$.) At the recursive base the solution returned is optimal, since it is computed by the dynamic programming algorithm. For the inductive step, consider the set $X \cup Z$. By the inductive hypothesis it follows that X is $(2 + \frac{2}{k-1})$-approximate with respect to $(T, C, w - w_1)$. Hence, this is also true for $X \cup Z$, since adding zero-weight vertices is for free. Moreover, by adding Z to X we ensure that $X \cup Z$ is also feasible with respect to (T, C, w). This is because every vertex u that is uncolored with respect to $w - w_1$ and is colored with respect to w belongs to Z. $X \cup Z$ is also feasible with respect to (T, C, w_1) since $w_1 \leq w$. Next, we show that every solution is $(2 + \frac{2}{k-1})$-approximate with respect to w_1. In follows, by the Local Ratio Theorem, that the computed cover is $(2 + \frac{2}{k-1})$-approximate with respect to (T, C, w) as well.

Lemma 5 *Every convex partial recoloring is* $(2 + \frac{2}{k-1})$*-approximate with respect to* w_1.

Proof Obviously, there are four possible types of w_1 that correspond to the four members of SEP. Let v be a (t, d)-separator, where $(t, d) \in \mathrm{SEP}$, and let U be the (t, d)-separating witness. Consider a cover Y that corresponds to a partial convex recoloring C'. It is not hard to see that $w_1(Y) \leq \varepsilon \cdot td$.

On the other hand, in a convex partial recoloring, for every $v \in V$ there is at most one color $c \in \mathcal{C}$ such that v separates c. Therefore, at least $(t-1)(d-1)$ vertices in U must be recolored. Thus, $w(Y) \geq \varepsilon \cdot (t-1)(d-1)$. It follows that the weight of every cover Y is within a factor of $\frac{td}{(t-1)(d-1)}$ from the optimum with respect to w_1. The lemma follows, since for $(t,d) = (2,k)$ and for $(t,d) = (k,2)$ we get $\frac{td}{(t-1)(d-1)} = \frac{2k}{k-1} = 2 + \frac{2}{k-1}$ and for $(t,d) = (3,4)$ and for $(t,d) = (4,3)$ we get $\frac{td}{(t-1)(d-1)} = 2$. □

Acknowledgements We thank the anonymous referees for their helpful comments and suggestions. We also thank Benny Chor for asking about the parameterized complexity of the dynamic programming algorithm.

References

1. Bafna, V., Berman, P., Fujito, T.: A 2-approximation algorithm for the undirected feedback vertex set problem. SIAM J. Discrete Math. **12**(3), 289–297 (1999)
2. Bar-Noy, A., Bar-Yehuda, R., Freund, A., Naor, J., Schieber, B.: A unified approach to approximating resource allocation and scheduling. J. ACM **48**(5), 1069–1090 (2001)
3. Bar-Yehuda, R.: One for the price of two: a unified approach for approximating covering problems. Algorithmica **27**(2), 131–144 (2000)
4. Bar-Yehuda, R., Even, S.: A local-ratio theorem for approximating the weighted vertex cover problem. Ann. Discrete Math. **25**, 27–46 (1985)
5. Bar-Yehuda, R., Bendel, K., Freund, A., Rawitz, D.: Local ratio: a unified framework for approximation algorithms. ACM Comput. Surv. **36**(4), 422–463 (2004)
6. Bodlaender, H.L., Fellows, M.R., Warnow, T.J.: Two strikes against perfect phylogeny. In: 19th International Colloquium on Automata, Languages, and Programming. LNCS, vol. 623, pp. 273–283. Springer, Berlin (1992)
7. Er, M.: On generating the n-ary reflected gray codes. IEEE Trans. Comput. **33**, 739–741 (1984)
8. Goldberg, L.A., Goldberg, P.W., Phillips, C.A., Sweedyk, E., Warnow, T.: Minimizing phylogenetic number to find good evolutionary trees. Discrete Appl. Math. **71**(1–3), 111–136 (1996)
9. Gusfield, D.: Efficient algorithms for inferring evolutionary trees. Networks **21**(1), 19–28 (1991)
10. Kannan, S., Warnow, T.: Inferring evolutionary history from DNA sequences. SIAM J. Comput. **23**(4), 713–737 (1994)
11. Kannan, S., Warnow, T.: A fast algorithm for the computation and enumeration of perfect phylogenies. SIAM J. Comput. **26**(6), 1749–1763 (1997)
12. Moran, S., Snir, S.: Convex recoloring of trees and strings: definitions, hardness results, and algorithms. In: 9th Workshop on Algorithms and Data Structures. LNCS, vol. 3608, pp. 218–232. Springer, Berlin (2005)
13. Moran, S., Snir, S.: Efficient approximation of convex recoloring. In: 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems. LNCS, vol. 3624, pp. 192–208. Springer, Berlin (2005)
14. Sankoff, D.: Minimal mutation trees of sequences. SIAM J. Appl. Math. **28**, 35–42 (1975)
15. Semple, C., Steel, M.: Phylogenetics. Mathematics and its Applications Series, vol. 22. Oxford University Press, London (2003)