

Convex Recolorings of Strings and Trees

Shlomo Moran*

Sagi Snir†

November 2003

Abstract

A coloring of a tree is convex if the vertices that pertain to any color induce a connected subtree; a partial coloring (which assigns colors to some of the vertices) is convex if it can be completed to a convex (total) coloring. Convex coloring of trees arises in areas such as phylogenetics, linguistics, etc. eg, a perfect phylogenetic tree is one in which the states of each character induce a convex coloring of the tree. Research on perfect phylogeny is usually focused on finding a tree so that few predetermined partial colorings of its vertices are convex.

When a coloring of a tree is not convex, it is desirable to know "how far" it is from a convex one. One common measure for this is based on the parsimony score, which is the number of edges whose endpoints have different colors. In this paper we study another natural measure for this distance: the minimal number of color changes at the vertices needed to make the coloring convex. This can be viewed as minimizing the number of "exceptional vertices" w.r.t. to a closest convex coloring. We also study a similar measure which aims at minimizing the number of "exceptional edges" w.r.t. a closest convex coloring. We show that finding each of these distances is NP-hard even for strings. We then focus on the first measure and generalize it to weighted trees, and then to non-uniform coloring costs. In the positive side we present few algorithms for convex recoloring of strings of trees: First we present algorithms for optimal convex recolorings of strings and trees with non-uniform coloring costs, which for any fixed number of colors are linear in the input size. Then we present fixed parameter tractable algorithms and approximation algorithms for convex recolorings of weighted strings and trees.

*Computer Science dept., Technion, Haifa 32000, Israel. moran@cs.technion.ac.il

†Computer Science dept., Technion, Haifa 32000, Israel. ssagi@cs.technion.ac.il

1 Introduction

Given a set of taxa (a group of related biological species), the goal of phylogenetic reconstruction is to build a tree which best represents the course of evolution for this set over time. The leaves of the tree are labeled with the given, extant taxa. Internal vertices correspond to hypothesized, extinct taxa. Because events of taxon divergence are assumed to be rare, the sought after tree is bifurcating (or binary), with internal vertices of degree three. (In case of ambiguous data one might have to resort to multifurcating trees, which are less informative.) In early days, morphologic features were mostly used to study evolution. Today, molecular data are the primary basis for phylogenetic analysis of evolution, but other sources of information (for example paleontology, anatomy, and morphology) are also in use. Phylogenetic reconstruction was one of the first algorithmic challenges posed by biology.

Phylogeny reconstruction methods are broadly divided into *character-based* and *distance-based* methods. Distance based methods start by computing “evolutionary distances” between pairs of taxa. Then a tree with weighted edges whose pairwise tree distances approximate the evolutionary distances is sought. In contrast, character based methods work directly on character data. In biology, characters describe attributes of the species under consideration and are the data that biologists use to reconstruct phylogenetic trees. Characters can be morphological (for example, wings versus no-wings), biochemical, physiological, behavioral, embryological, or genetic (for example, the nucleotide at a particular DNA sequence position, or the order of certain genes on a chromosome). A rooted phylogenetic tree describes the evolutionary history of a set of extant species, evolved from some ancestral species at the root of the tree. In this way we can see the character states along a path from the root to some species as “evolving” to that species’ state.

Mathematically, if X is the set of species under consideration, a (qualitative or discrete) character on X is a function C from X into a set \mathcal{C} of character states. A natural biological constraint is that the reconstructed phylogeny have the property that each of the characters could have evolved without reverse or convergent transitions: In a reverse transition some species regains a character state of some old ancestor whilst its direct ancestor has lost this state. A convergent transition occurs if two species possess the same character state, while their least common ancestor possesses a different state. The concept behind this constraint is of “innovation”. That is, each time the character state changes, it acquires a new state. The innovation assumption exclude reverse and convergent transitions, and is denoted alternatively - *homoplasy free* character [18]. In nature, homoplasy does occur, however these events are considered relatively rare. The acquisition of teeth with the birds ancestor, the Archaeopteryx, and their subsequent loss is an example of reverse transition, and the convergence of evolution in placental and Australian mammals is an example of convergent transition.

In graph theoretic terms, a character in a phylogenetic tree is homoplasy free if it is convex, that is: for each state of this character, all species (extant and/or extinct) possessing that state induce a subtree. Thus, the above discussion implies that in a phylogenetic tree, each character is likely to be convex or “almost convex”. This makes convexity a fundamental property in the context of phylogenetic trees.

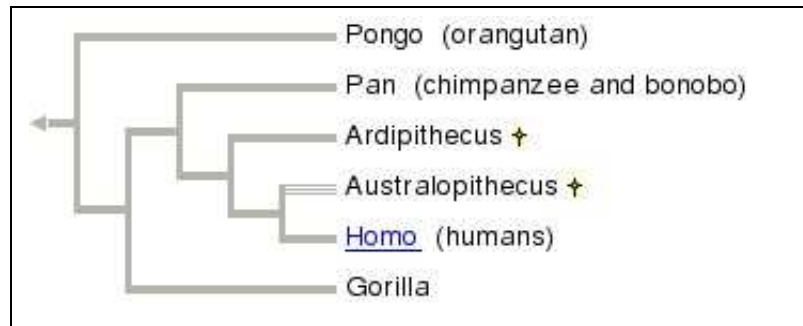


Figure 1: The primates tree, taken from “The Tree of Life” project.

One of the best known and most widely used character-based methods for constructing phylogenetic trees is *maximum parsimony* (MP) [9]. This is a combinatorial method, which attempts to construct trees which minimize the number of violations of the convexity property. Specifically, Fitch algorithm for “small parsimony” [9] finds, for a given tree with character states assigned to its leaves, an assignments of characters states to internal vertices which minimizes the number of violations of the convexity property.

Perfect phylogeny is a special case of maximum parsimony. The input is a set of characters while the tree sought is homoplasy free for each of the characters. The general perfect phylogeny problem was shown to be NP-Complete by Bodlaender *et. al.* [5] and independently by Steel [19]. Nevertheless, when restricting the number of possible states for each character, r , or considering r as a parameter, there are some positive results: For binary character, i.e. each state is either 0 or 1, Gusfield [11] gave an $O(nk)$ time algorithm where n is the number of species and k is the number of characters. Dress and Steel [7] devised an $O(nk^2)$ for $r \leq 3$ and Kannan and Warnow [13] gave a $O(n^2k)$ algorithm for $r \leq 4$. when considering r as a parameter, Agarwala and Fernandez-Baca [1] showed a fixed parameter tractable algorithm that runs in time $O(2^{3r}(nk^3 + k^4))$. This work was later improved by Kannan and Warnow [14] to $O(2rnk^2)$ time.

In this work we deal with following aspect of perfect phylogeny. For some set of species, the evolutionary tree is already known (e.g. the primates tree shown in Figure 1). Given a character relating to a subset of all the species (extant and extinct) in the tree, we want to know how much this character “agrees” with the tree. That is, how far this character is from perfect phylogeny on the given tree. There are few natural measures for this distance. In this paper we focus on the minimal number of species whose states should be changed to make the given character convex. Another measure that we discuss is the minimal number of state changes that should be removed for making the character convex on the tree, or the minimum number of changes of both types which are needed to achieve this purpose. Indeed, the evolution of a character which can be made convex by removing a small number of exceptions, can be explained by searching biological reasoning for few exceptional phenomena, as was done for the two above mentioned cases (see e.g. [15]). If however a very large number of state changes is needed to make the character convex, a biological explanation becomes less probable, and the reliability of the given phylogeny as a correct

description of the evolution of this character diminishes accordingly.

We note that our problem for partially colored trees bears some similarity to the small parsimony problem mentioned above. In both problems a tree with a (partial) assignments of states to the vertices is given. The small parsimony problem is equivalent to find the minimum number of violations to the perfect phylogeny property, and the convex recoloring problem finds the minimum number of color changes needed to achieve perfect phylogeny. It is therefore a bit surprising that while the weighted version of the maximum parsimony problem has an efficient optimal solution [17], the unweighted version of minimum convex recoloring is NP-Hard, as we show here.

We study also two more general cost functions, which enable more flexibility in measuring the distance of a given coloring from a convex one: The first allows weights on the vertices, and the most general one allows also non uniform cost function for color changes (details are in Section 2).

Our negative results show that the (unweighted) versions of the above problems are hard even for a simple tree - the string, and for the case where character states are given only at the leaves (so that changes on extant species are not counted); we also prove that finding the minimum number of mutation removals needed to obtain convexity, in a sense to be defined, is NP-hard. On the positive side, we present dynamic programming algorithms that for each fixed number of colors solve the non-uniform versions of the problem in polynomial time. Then we show that for strings and trees of bounded degree, the (unweighted version of the) problem can be solved by a fixed parameter tractable algorithm. The proof of this result is based on identifying *signatures* of recolorings, so that the number of possible signatures is bounded for a fixed parameter, and then presenting a polynomial time algorithm for a minimal convex recoloring with a given signature. Finally, we present polynomial time algorithms for 2-approximation for strings and 3-approximation for trees, for the weighted versions of the problem. The 2-approximation for strings is based on a short lower bound argument for optimal convex recoloring of strings. The 3-approximation for trees is based on reducing the given weighted colored tree to one of smaller support (set of vertices of positive weight); depending on the specific instance, the reduction can be either of “local ratio” type, which keeps the topology but decreases the weight function [3], or of “combinatorial” type, which changes the topology of the input in a local manner.

The rest of the paper is organized as follows. The next section presents the notations used and define the unweighted, weighted and non-uniform versions of the problem. In Section 3 we present our NP-Hardness results. In Section 4 we present dynamic programming algorithms for the problem. In Section 5 we present fixed parameter tractable algorithms, and in Section 6 we present approximation algorithms for it.

2 Preliminaries

A colored tree is a pair (T, C) where $T = (V, E)$ is a tree with vertex set $V = \{v_1, \dots, v_n\}$, and C is a *coloring* of T , i.e. - a function from V onto a set of colors \mathcal{C} . A *block* in a colored tree is a maximal set of vertices which induces a monochromatic subtree. A *d-block* is a block of color d . The number of d -blocks is denoted by $n_b(C, d)$, or $n_b(d)$ when C is clear from the context. A coloring C is said to be *convex* if $n_b(C, d) = 1$ for every color $d \in \mathcal{C}$. The number of *d-violations* in

the coloring C is $n_b(C, d) - 1$, and the total number of *violations* of C is $\sum_{c \in \mathcal{C}} (n_b(C, d) - 1)$. Thus a coloring C is convex iff the total number of violations of C is zero. In fact, some authors use the above sum, taken over all characters, as a measure of the distance of a given phylogenetic tree from perfect phylogeny [8].

The definition of convex coloring is extended to *partially colored* trees, in which the input coloring C assigns colors to some subset of the vertices. A partial coloring is said to be convex if it can be completed to a (total) convex coloring (see [18]). Let (T, C) be a (totally or partially) colored tree. For each color d , the d -*container*, $T_{d,C}$ denotes the minimal subtree of T which contains $C^{-1}(d)$. We say that C has the *disjointness property* if for each pair of colors $\{d, d'\}$ it holds that $T_{d,C} \cap T_{d',C} = \emptyset$. It is easy to see that a total coloring C is convex iff it satisfies the disjointness property. Moreover, a partial coloring C' of T can be completed to a (total) convex coloring C of T iff it satisfies the disjointness property (in [7] convexity is actually defined by the disjointness property).

When some (partial or total) input coloring (C, T) is given, any other coloring C' of T is viewed as a *recoloring* of the input coloring C . We say that a recoloring C' of C *retains* (the color of) a vertex v if $C(v) = C'(v)$, otherwise C' *overwrites* v . When C is a total coloring, C' retains (overwrites) a block B if it retains (overwrites resp.) every vertex in B . For a recoloring C' of an input coloring C , $\mathcal{X}_C(C')$ (or just $\mathcal{X}(C')$) is the set of vertices overwritten by C' , i.e. $\mathcal{X}_C(C') = \{v \in V : C(v) \text{ is defined and } C(v) \neq C'(v)\}$.

With each recoloring C' of C we associate a *cost*, denoted as $cost_C(C')$ (or $cost(C')$ when C is understood), which is the number of vertices overwritten by C' , i.e. $cost_C(C') = |\mathcal{X}_C(C')|$. A coloring C^* is an *optimal convex recoloring* of C , or in short an *optimal recoloring* of C , and $cost_C(C^*)$ is denoted by $OPT(T, C)$, if C^* is a convex coloring of T for which $cost_C(C^*) = \min_{C'} \{cost_C(C')\}$, where the minimum is taken over all convex recolorings of C .

Finding optimal (convex) recoloring can be defined as a minimum cost covering problem, as follows: Let $(T = (V, E), C)$ be a colored tree. For a subset U of V , the restriction of C to U , $C|_U$, has the obvious meaning. We say that a set of vertices X is a *cover* for (T, C) if the (partial) coloring $(T, C|_{[V \setminus X]})$ is convex (i.e., C can be transformed to a convex coloring by overwriting the vertices in X). Observe that if C' is a convex recoloring of (T, C) Then $\mathcal{X}_C(C')$ is a cover for (T, C) , and vice versa: if X is a cover for (T, C) , then there is a convex recoloring C' of C such that $\mathcal{X}(C') = X$. Thus, finding an optimal convex recoloring is equivalent to finding a a cover of minimum possible cardinality for C .

The cost function of a coloring can be generalized in few ways. The first one is the *weighted* version, which corresponds to a weighted cover problem. The input is a triplet (T, C, w) , where $w : V \rightarrow \mathbb{R}^+ \cup \{0\}$ is a nonnegative weight function which assigns to each vertex v a nonnegative weight $w(v)$. Let the *support* of w be the set $support(w) = \{v \in V : w(v) > 0\}$. We will assume that C assigns colors only to vertices in $support(w)$ (since if $w(v) = 0$ then v can be overwritten by any color at no cost). For a set of vertices X , $w(X) = \sum_{v \in X} w(v)$. Thus the weight of a recoloring C' of C is $W_C(C') = w(\mathcal{X}(C'))$.

A yet further generalization allows *non-uniform* cost functions. This version, motivated by weighted maximum parsimony [17], assumes that the cost of replacing one given color by a another could

depend on the specific colors involved. Thus for each pair of colors there is a nonnegative integer $g(d, d')$, which denotes the cost of changing the color d to d' (hence $g(d, d) = 0$ for all d). The cost of assigning color d to vertex v in this model is given by $cost_C(v, d) = g(C(v), d) \cdot w(v)$. and the cost of a recoloring C' is the sum $\sum_{v \in X_C(C')} cost_C(v, C'(v))$. The non-uniform model appears to be more subtle than the two previous ones. Unless otherwise stated, our results assume the weighted and unweighted models.

We complete this section with a definition and a simple observation which will be useful in the sequel. Let (T, C) be a colored tree. A coloring C^* is an *expanding* recoloring of C if in each block of C^* at least one vertex v is retained (i.e., $C(v) = C^*(v)$).

Observation 2.1 *let (T, C) be a colored tree. Then there exists an expanding optimal convex recoloring of C .*

Proof. Let C' be an optimal recoloring of C which uses a minimum number of colors (i.e. $|C'(V)|$ is minimized). We shall prove that C' is an expanding recoloring of C .

If C' uses just one color c , then by the optimality of C' , there must be a vertex v such that $C(v) = c$ and the claim is proved. Assume for contradiction that C' uses at least two colors, and there is a color c used by C' for which there is no vertex v s.t. $C(v) = C'(v) = c$. There must be an edge (u, v) such that $C'(u) = c$ but $C'(v) = d \neq c$. Then the coloring C'' which is identical to C' except that all vertices colored c are now colored by d is an optimal recoloring of C which uses a smaller number of colors - a contradiction. ■

3 NP-Hardness results

The main result of this section is that unweighted minimum convex recoloring of strings is NP-Hard. Then we use reductions from this problem to prove that the unweighted versions of minimal convex recoloring of leaves and minimum number of mutations needed to transform a coloring of a tree to a convex coloring are NP-Hard as well.

3.1 Minimal Convex Recoloring of Strings is NP-Hard

Strings are simple trees in which $E = \{(v_i, v_{i+1}) | i = 1, \dots, n - 1\}$. In a colored string (S, C) , a d -block is simply a maximal sequence of consecutive vertices colored by d . A nice property of optimal convex recolorings of strings, which is valid also for the non-uniform model, is given below:

Claim 3.1 *Let (S, C) be a colored string, and let C^* be an optimal recoloring of C . Then each block of C is either completely retained or completely overwritten by C^* .*

Proof. Suppose, for contradiction, that B' is a d -block in C that is partially overwritten by C^* . Let C' be a recoloring identical to C^* except that C' retains the block B' . Then C' is convex and $cost(C') < cost(C^*)$ - a contradiction. ■



Figure 2: A Schematic view of the colored string corresponding to F . Informative segments appear white (in the figure) where junk segments are longer and have distinct colors.

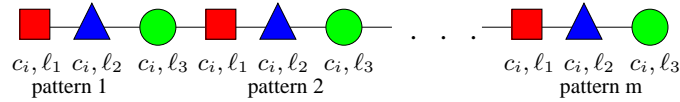


Figure 3: A clause segment. The literals are ℓ_1 , ℓ_2 and ℓ_3 , and the clause of size $3A$ consists of A repetitions of the corresponding triplet. Each block is a single vertex.

We prove that the problem is NP-Hard by reducing the 3 satisfiability problem to the following decision version of minimal convex recoloring:

Minimal Convex Recoloring of Strings:

Input: A colored string (S, C) and an integer k

Question: Is there a convex recoloring C^* of C such that $cost_C(C^*) \leq k$.

Let formula F be an input to the 3 satisfiability problem, $F = D_1 \wedge \dots \wedge D_m$, where $D_i = (l_{i1} \vee l_{i2} \vee l_{i3})$ is a clause of three literals, each of which is either a variable x_j or its negation $\neg x_j$, $1 \leq j \leq n$. We describe below a polynomial time reduction of F to a colored string (S, C) and an integer k , such that there is a convex coloring C^* of C with $cost_C(C^*) \leq k$ iff F is satisfiable.

In the reduction we define block sizes using parameters A and B , where A and B are integers satisfying $A > m - 2$ and $B > 2mA$. k is set to $n(2m + 1)B + 2mA$ (e.g., possible values are $A = 3m$, $B = 9m^2$, and $k = 3m^2(6mn + 3n + 2)$).

We describe the coloring C of S as a sequence of *segments*, where each segment consists of one or more consecutive blocks. There will be $2n + m$ *informative* segments: one for each clause and one for each literal, and $2n + m - 1$ *junk* segments separating the informative segments (see Figure 2). Each junk segment consists of a unique block of $k + 1$ vertices colored by a distinct color, thus $2n + m - 1$ colors are used for the junk segments. The informative segments will use additional n *variable colors* d_1, \dots, d_n and $2nm$ *literal colors* $\{c_{i,x_j}, c_{i,\neg x_j} \mid i = 1, \dots, m; j = 1, \dots, n\}$.

For each clause $D_i = (l_1 \vee l_2 \vee l_3)$ there is a *clause segment* S_{D_i} of size $3A$, obtained by A repetitions of the pattern $c_{i,\ell_1}, c_{i,\ell_2}, c_{i,\ell_3}$ (see Figure 3).

for each non-negated literal x_j there is a *literal segment* S_{x_j} , which consists of $2m + 1$ consecutive blocks of the same size B . All the $m + 1$ odd numbered blocks are d_j -blocks, called *variable blocks*. The m even numbered blocks are *literal blocks*, colored by $c_{i,x_j}, i = 1, \dots, m$, see Figure 4. Similarly, for each negated literal $\neg x_j$ we have a literal segments $S_{\neg x_j}$, which is similar to S_{x_j} except that the colors of the literal blocks are $c_{i,\neg x_j}, i = 1, \dots, m$ (note that each of the literal segments S_{x_j} and $S_{\neg x_j}$ contain $m + 1$ d_j -blocks).

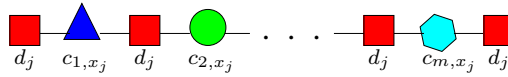


Figure 4: S_{x_j} , the segment of the literal x_j . $m + 1$ d_j -blocks are interleaved by the m blocks c_{i,x_j} , $i = 1, \dots, m$.

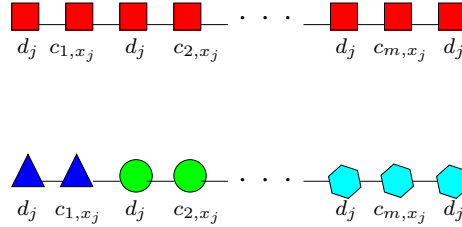


Figure 5: A recoloring of segments S_{-x_j} and S_{x_j} corresponding to a satisfying assignment.

Theorem 3.2 *Let (S, C) be the colored string defined by the above reduction. Then $OPT(S, C) \leq k$ iff F is satisfiable.*

Proof. \Leftarrow we need to prove that if the formula F is satisfiable, then there is a convex recoloring C^* of C such that $cost_C(C^*) \leq k$.

Let f be a satisfying assignment of F . The coloring C^* is defined for literal segments as follows: For each variable x_j s.t. $f(x_j) = 1$, C^* overwrites each of the d_j -blocks in segment $S_{\neg x_j}$ (there are $m + 1$ such blocks); in the segment S_{x_j} , C^* overwrites all the c_{i,x_j} blocks, for $i = 1, \dots, m$ (see Figure 5). The coloring when $f(x_j) = 0$ is obtained by interchanging the roles of S_{x_j} and $S_{\neg x_j}$. This requires recoloring of $(2m + 1)B$ vertices for each variable, so the total cost for all literal segments is $n(2m + 1)B$.

We now define C^* on clause segments. Since f is a satisfying assignment, in each clause there is a literal which is set by f to 1. Assume without loss of generality that $x_j \in D_i$ and $f(x_j) = 1$. By the written above, C^* does not color any vertex in the literal segments by c_{i,x_j} . Thus we can transform segment D_i to a c_{i,x_j} -block by recoloring $2A$ vertices (since A vertices are originally colored by c_{i,x_j}). Thus the total cost of coloring all the m clause segments is $2mA$.

\Rightarrow Now we have to prove that if $OPT(S, C) \leq k$, then F is satisfiable. Let C^* be an expanding optimal recoloring of C (see Observation 2.1). Clearly, $cost_C(C^*) \leq k$. The proof proceeds through the following claims.

Claim 3.3 C^* retains all the junk segments.

Proof. A junk segment, J , consists of a single block of $k + 1$ vertices. By Claim 3.1 C^* either completely overwrites J or completely retains it. Since C^* overwrites at most k vertices altogether, the latter possibility must hold. ■

Claim 3.4 *The coloring C^* satisfies the following for each pair of literal segments $\{S_{x_j}, S_{\neg x_j}\}$, $j \in \{1, \dots, n\}$:*

1. *In exactly one of these segments, C^* overwrites all the d_j -blocks, and retains all the literal blocks.*
2. *In the other segment, C^* overwrites exactly m blocks.*

In particular, C^ overwrites exactly $2m + 1$ blocks in these two segments.*

Proof.

consider the substring containing segments S_{x_j} and $S_{\neg x_j}$. Then it contains exactly $2m + 1$ d_j -violation, since each of these segments contains $m + 1$ d_j -blocks. For C^* to be convex, it must remove all these violations. Since by claim 3.3 all junk blocks retain their colors, C^* must overwrite all the d_j -blocks in one of the above segments, and leave at most one d_j -block in the other. The former case clearly requires overwriting each of the $m + 1$ d_j -blocks in the relevant segment, which leaves $m + 1$ d_j -blocks and (hence) m d_j -violations in the other segment, which must be removed. Since overwriting any single block of C can reduce the number of d_j -violations by at most one, at least m such blocks must be overwritten.

So far we have shown that C^* must overwrite at least $m + 1$ blocks in one segment and at least m blocks in the other, a total of $2m + 1$ blocks in each such pair of segments. To complete the proof it suffices to show that C^* does not overwrite any other block in the literal segments. To this end we observe that if for some j at least $2m + 2$ blocks are overwritten in the variable segments $S_{x_j}, S_{\neg x_j}$, then C^* overwrites at least $n(2m + 1) + 1$ blocks in the literal segments, and since each such block has B vertices, the total number of overwritten vertices is at least $n(2m + 1)B + B > n(2m + 1)B + 2mA = k$ (since $B > 2mA$), contradicting the assumption on C^* .

■

Using Claim 3.4 above, we can now define a truth assignment f which satisfies F , as follows: for $j = 1, \dots, n$, $f(x_j) = 1$ iff C^* overwrites exactly m blocks in S_{x_j} (and hence exactly $m + 1$ blocks in $S_{\neg x_j}$). To simplify notations, we assume in the rest of the proof that for all j , exactly m blocks are overwritten in S_{x_j} , and hence $f(x_j) = 1, j = 1, \dots, n$. We complete the proof by showing that f indeed satisfies F .

Claim 3.5 *C^* overwrites at least $2A - 2$ vertices at every clause segment.*

Proof. Consider a clause segment, D , whose three literal colors are c_1, c_2 and c_3 . The claim trivially holds if all the $3A$ vertices in D are overwritten, so assume that this is not the case. Since all junk segments are retained by C^* , we may assume, using argument similar to the one in the proof of Observation 2.1, that $D \subseteq C^{*-1}(\{c_1, c_2, c_3\})$, and thus $C^*(D)$ consists of at most 3 blocks of these colors. Let the lengths of the c_i -block be l_i ($l_i \geq 0, l_1 + l_2 + l_3 = 3A$). Observe that out of any 3 consecutive vertices within each such block, C^* must overwrite exactly 2 vertices. Hence, for each i the following holds: if $l_i = 0 \pmod{3}$ then C^* overwrites exactly $\frac{2}{3}l_i$ vertices in the c_i -block; if $l_i = 1 \pmod{3}$ then at least $\frac{2}{3}(l_i - 1)$ vertices are overwritten in that block, and if $l_i = 2 \pmod{3}$

then at least $\frac{2}{3}(l_i - 2) + 1 = \frac{2}{3}(l_i + 1)$ vertices are overwritten. Thus, for $i = 1, 2, 3$, at least $\frac{2}{3}(l_i - 1)$ vertices must be overwritten in the c_i -block. Altogether at least $\frac{2}{3}(l_1 + l_2 + l_3 - 3) = \frac{2}{3}(3A - 3) = 2A - 2$ vertices must be overwritten in D . ■

Claim 3.6 *At every clause segment, at least one vertex is retained.*

Proof. Seeking for contradiction, assume all the $3A$ vertices in some clause segment S_{D_i} are overwritten. Then by Claim 3.5, C^* overwrites at least $(m - 1)(2A - 2) + 3A = 2mA + A - 2m + 2 > 2mA$ vertices in all clauses' segments (the last inequality holds since $A > 2m - 2$ by definition). Adding this to the $n(2m + 1)B$ vertices overwritten in the variable segments, we get that C^* overwrites more than $n(2m + 1)B + 2mA = k$ vertices - a contradiction. ■

The proof of Theorem 3.2 is now completed by the following claim:

Claim 3.7 *The function f (as defined before Claim 3.5) satisfies F .*

Proof. Since $f(x_j) = 1$ for $j = 1 \dots, n$, we need to show that each clause D_i in F contains an unnegated variable.

By Claim 3.6, at least one vertex is retained in S_{D_i} . The color of this vertex can be either $c_{i,-x_j}$ or c_{i,x_j} for some j . By Claim 3.4.1 C^* retains all the $c_{i,-x_j}$ -blocks in the literal segments, and hence (by convexity) it cannot retain another such block in any clause segment. Thus the color of the retained vertex must be of the form c_{i,x_j} , meaning that the non negated literal x_j is in clause D_i . ■

3.2 NP Hardness of Minimal Convex Recoloring of Leaves

A *leaf colored tree* is a partial colored tree (T, C) in which the coloring C assigns colors only to leaves of T . Such trees are common in phylogenetics, where the leaves present existing species, and internal vertices present extinct ones. Now, given a certain character states on the existing species, we wish to know what is the minimum number of color changes at colored vertices (leaves) needed for transforming the input coloring to a convex coloring. The NP hardness result of the previous section does not apply directly to this problem, and we show in this section that the corresponding decision problem for the unweighted version of this problem is NP complete.

Minimal Unweighted Convex Recoloring of Leaves

Input: A leaf colored tree (T, C) and an integer k

Question: Is there a convex recoloring C' of C s.t. $|\mathcal{X}_C(C')| \leq k$

Theorem 3.8 *Minimal unweighted convex recoloring of leaves is NP-Complete.*

Proof. We reduce the minimal convex string recoloring problem to a minimal convex leaves recoloring problem. Given a colored string (S, C) , we reduce it to a leaf colored tree as follows.

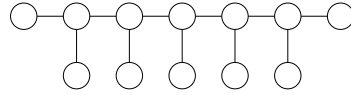


Figure 6: A caterpillar of length 5.

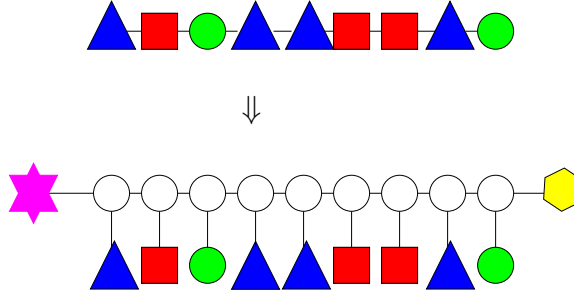


Figure 7: A reduction from a fully colored string to a leaf colored caterpillar. Two leaves in the two ends are colored with two new colors. All other leaves are colored with the same color as the corresponding vertex in the string.

For a colored string (S, C) of length n and an integer l , $dup_l(S, C) = (S', C')$ is a colored string of length ln defined as follows: Let $V(S) = \{v_1, \dots, v_n\}$; then $V(S') = \{v_i^j : 1 \leq i \leq n, 1 \leq j \leq l\}$ and $E(S') = \{(v_i^{j-1}, v_i^j) : 1 \leq i \leq n, 1 < j \leq l\} \cup \{(v_{i-1}^n, v_i^1) : 1 \leq i < n\}$. $C'(v_i^j) = C(v_i)$, $i = 1, \dots, n, j = 1, \dots, l$. Informally, $dup_l(S, C)$ is a duplication of every vertex v in (S, C) l times, obtaining an ln long colored string. The proof of the following observation follows easily from Claim 3.1.

Observation 3.9 $OPT(dup_\ell(S, C)) = \ell \cdot OPT(S, C)$.

We now define a type of an unrooted binary tree. A *caterpillar* is a binary tree having at most two vertices which are each adjacent to two leaves. A caterpillar is of length n if it has (a string of) n internal vertices (see Figure 6). Given a (totally) colored string (S, C) of length n we construct a *leaf colored caterpillar* of length n , $cat(S, C) = (T, C')$ as follows: The internal vertices of T form a string isomorphic to S , numbered 1 to n from left to right. The leftmost leaf (connected to internal vertex 1) is colored with a distinct new color, as well as rightmost leaf (connected to internal vertex n). Each other leaf connected to an internal vertex i inherits its color from vertex i in the colored string (S, C) (see Figure 7).

Claim 3.10 *Let (S, C) be a colored string, where C uses n_c colors, and let $(T, C_T) = cat(dup_{n_c}(S, C))$. Then,*

$$(OPT(S, C) = k) \iff (n_c k - n_c < OPT((T, C_T)) \leq n_c k).$$

Proof. We assume first that $OPT(S, C) = k$ and prove the two inequalities at the right hand side.

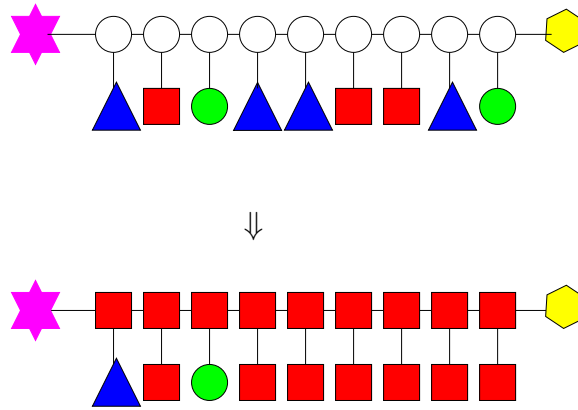


Figure 8: A convex recoloring of the input caterpillar. All blue (triangle) and green (circles) blocks were recolored, so a single vertex of each color can be retained without violating convexity.

Let $(S', C') = \text{dup}_{n_c}(S, C)$. By Observation 3.9, (S', C') has a recoloring C^* with cost $n_c k$. We transform C^* to a total convex coloring C_T^* of (T, C_T) as follows: C_T^* duplicates C^* on the internal vertices of T , and it colors the leaves of T with the color of their neighbors. C_T^* is convex, and $\text{cost}(C_T^*) = \text{cost}(C^*) = n_c k$. This proves the right inequality.

To prove the other (strict) inequality, let C_T^* be an optimal expanding convex recoloring of (T, C_T) . First observe that C_T^* on the internal vertices of T induces a convex recoloring on S' , which we will call C^* .

Since C_T^* uses at most n_c colors, it has at most $n_c - 1$ blocks of size one, hence the number of leaves whose color under C_T^* is different than the color of their neighboring internal vertices is at most $n_c - 1$. Hence $\text{cost}(C^*) < \text{cost}(C_T^*) + n_c$. Thus we have

$$n_c k \leq \text{cost}(C^*) < \text{cost}(C_T^*) + n_c = \text{OPT}((T, C_T)) + n_c,$$

which implies the left inequality.

The proof of the other direction is similar, and omitted. ■

By Claim 3.10 above a polynomial time solution for minimal convex recoloring of leaves will imply such a solution for the minimal convex recoloring of strings, which completes the proof of the theorem. ■

3.3 NP Hardness of Minimum Block-Recolorings

A *block-recoloring* corresponds to changing the colors of all the vertices in a block to a unique different color. Such an operation seems a reasonable modeling of removing a mutation from a phylogenetic tree. Indeed, mutation is an edge (u, v) such that $C(u) \neq C(v)$, and the removal of a mutation implies changing the color of a block at one end of the edge to the color of the block at the other end. Note that a block-recoloring which corresponds in this way to the removal

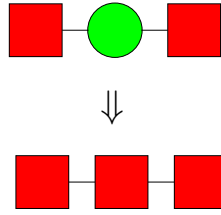


Figure 9: Removing the mutation at the left edge implies the removal of the one at the right

of one mutation can imply the elimination of other mutations, as depicted in Figure 9. Also, as in Observation 2.1 we can show that allowing block-recolorings by arbitrary colors (i.e., not only by colors of adjacent blocks) cannot reduce the minimum number of block-recolorings needed to transform a given coloring to a convex one. Therefore we can model the problem of minimizing the number of mutation removals as minimizing the number of block-recolorings needed to transform the input coloring to a convex one.

The weighted version of this problem is a special case of the weighted version of the minimum convex recoloring problem, since a colored weighted tree can be transformed to an equivalent "blocks tree", where each block in the original tree is collapsed to a single vertex whose weight is the weight of the original block. Applying this constructions on colored strings can be used to show that the *weighted* version of the problem is NP-Hard for colored strings. In the rest of this section we show that the *unweighted* version of this problem for colored string is NP-Hard as well. We actually prove the following stronger result: Let a *Zebra string* be a colored string (S, C) in which for every edge $(u, v) \in E$ it holds that $C(u) \neq C(v)$ (i.e., every block is a single vertex).

Theorem 3.11 *The unweighted minimum convex recoloring of Zebra strings is NP-Hard.*

Proof. The proof is by reduction from the minimum convex recoloring of strings. Let (S, C) be a colored string of n vertices. We reduce it to a Zebra string (S_z, C_z) of length $16n$ such that (S, C) has a recoloring C' with $cost_C(S, C') = k$ iff (S_z, C_z) has a recoloring C'_z with $cost_{C_z}(S_z, C'_z) = 5n + k - 1$. The Zebra string (S_z, C_z) consists of three neighboring segments: informative segment, junk segment and a counter-weight segment, in this order. The segments are constructed as follows:

- **Informative segment:** A $2n - 1$ long segment comprised of the input string in which a spacer vertex, colored with a new color d_s , is inserted between any neighboring vertices u and v (See Figure 10).
- **Junk segment** A $6n$ long segment in which the vertices are colored by $6n$ new distinct colors, used to separate between the informative segment and the counter-weight segment.
- **Counter-weight segment** A $8n + 1$ long segment comprised of $2n$ consecutive quartets $[d_s, d_1, d_s, d_2]$ appended with a d_s -vertex, where d_s is the spacer color used in the informative segment and d_1 and d_2 are new additional colors (See Figure 11).

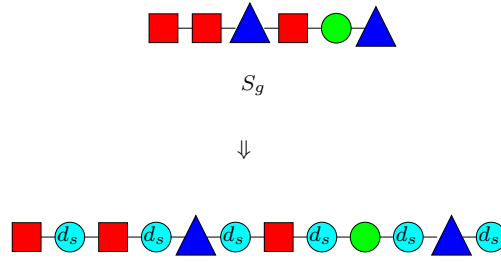


Figure 10: The input string (S, C) and the corresponding informative segment in (S_z, C_z) .

we now show that (S, C) has a convex recoloring C' of cost k if and only if (S_z, C_z) has a convex recoloring C'_z of cost $m = 5n + k - 1$.

\implies Assume that (S, C) has a convex recoloring C' of cost k . The corresponding convex recoloring C'_z of S_z is defined as follows:

In the informative segment, the n vertices corresponding to the input string (S, C) are colored as defined by C' , and then the $n - 1$ remaining d_s -vertices are overwritten by expanding the coloring of their neighbors. Thus the cost of C'_z in the informative segment is $n + k - 1$. In addition, the $4n$ vertices colored by d_1 and d_2 in the counter-weight segment are colored by d_s .

The total cost of C'_z is m , as required. It is easy to verify that C'_z is a convex coloring of S_z .

\Leftarrow Assume now that C'_z is a convex recoloring of (S_z, C_z) of cost m . W.l.o.g. we may assume that C'_z is an expanding recoloring of C_z . We construct a recoloring C' of (S, C) of cost k , using the following observations.

Observation 3.12 *If C'_z retains a d_s -vertex in the counter-weight segment, then it overwrites all the d_s -vertices in the informative segment.*

Proof. If C'_z retains d_s -vertices in both the informative and counter-weight segments, then it must overwrite (by d_s) all the $6n$ vertices in the junk segment, but $6n > m$. ■

Observation 3.13 *C'_z retains a d_s -vertex in the counter-weight segment.*

Proof. Any convex recoloring of the counter-weight segment must overwrite either a d_1 -vertex or a d_2 -vertex in $2n - 1$ out of the $2n$ quartets in this segment. This sums to at least $2n - 1$ vertices. If C'_z overwrites also all the $4n + 1$ d_s -vertices in the counter-weight segment, then it overwrites (in this segment) $6n > m$ vertices, a contradiction. ■

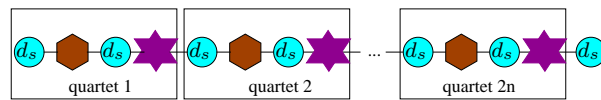


Figure 11: The counter-weight segment in S_s .

Observation 3.14 C'_z overwrites at least $4n$ vertices in the counter-weight segment.

Proof. It is straightforward to show that the only optimal convex coloring of the counter-weight segment is the one which transform it to a d_s -block, and this coloring overwrites exactly $4n$ vertices. ■

Observation 3.14 implies that C'_z overwrites at most $m - 4n = n + k - 1$ vertices in the informative segment, and observations 3.12 and 3.13 imply that $n - 1$ of them must be d_s -vertices. The remaining k vertices in the informative segments which are overwritten by C'_z belong to the copy of (S, C) , and define a convex recoloring of (S, C) of cost k . ■

Note: In a Zebra string, overwriting a single vertex is also a block recoloring. Thus Theorem 3.11 also implies that the problem of minimizing the total number of vertex recolorings *and* block recolorings needed to transform a colored string to convex one is NP-Hard.

4 Exact Algorithms

4.1 Non-uniform Convex String Recoloring

We describe here a dynamic programming algorithm that finds a minimum convex recoloring of a string. The algorithm is valid for the non-uniform model, in which $cost_C(v, d)$, the cost of recoloring a vertex v by color d equals $w(v) \cdot g(C(v), d)$, where $w(v)$ is the weight of v , and $g(C(v), d)$ is the cost of changing the color of v from its input coloring $C(v)$ to d . By Claim 3.1, an optimal convex recoloring of a colored string (S, C) either completely retains or completely overwrites every block of (S, C) . It is not hard to show that if a block B is completely overwritten by an optimal recoloring C^* , then we may assume that C^* overwrites B by a single color. The cost of coloring B with color d is denoted by $cost(B, d) = \sum_{v \in B} cost(v, d)$.

Let (S, C) be a colored, n -long input string with n_b blocks, where B_i is the i th block starting from left. The algorithm scans the string, block by block, from left to right. After processing block B_i , it keeps for each subset of colors $\mathcal{D} \subseteq \mathcal{C}$, and for each color $d \in \mathcal{D}$, the cost of the optimal coloring of blocks B_1, \dots, B_i which uses only colors from \mathcal{D} , and the rightmost block B_i is colored by d ; this cost is denoted $opt_{\mathcal{D}, d}(i)$, and $opt_{\mathcal{D}}(i)$ denotes $\min_{d \in \mathcal{D}} opt_{\mathcal{D}, d}(i)$. Thus, the cost of an optimal coloring of the input string is $opt_{\mathcal{C}}(n_b)$. Let n_c be the cardinality of \mathcal{C} . Then the number of scores which needs to be saved is at most $\sum_{i=1}^{n_c} i \binom{n_c}{i} = n_c 2^{n_c - 1}$.

Claim 4.1

$$opt_{\mathcal{D}, d}(i) = cost(B_i, d) + \min(opt_{\mathcal{D}, d}(i-1), opt_{\mathcal{D} \setminus \{d\}}(i-1))$$

where for all $\mathcal{D} \subseteq \mathcal{C}$ and $d \in \mathcal{D}$, $opt_{\mathcal{D}, d}(0) = 0$

Proof. It is easy to verify correctness of the claim for $i = 1$. Assume for contradiction that i is the minimal integer for which $opt_{\mathcal{D}, d}(i)$ is smaller than $cost(B_i, d) + \min(opt_{\mathcal{D}, d}(i-1), opt_{\mathcal{D} \setminus \{d\}}(i-1))$.

Then if $C^*(b_{i-1}) = d$ it contradicts the optimality of $opt_{\mathcal{D},d}(i-1)$. Otherwise, it contradicts optimality of $opt_{\mathcal{D}\setminus\{d\}}(i-1)$. ■

Claim 4.1 yields the following dynamic programming algorithm for the minimal convex string recoloring:

Non-Uniform Optimal Convex String Recoloring

1. for every $\mathcal{D} \subseteq \mathcal{C}$ and for every $d \in \mathcal{D}$, $opt_{\mathcal{D},d}(0) \leftarrow 0$
2. for $i = 1$ to n_b
 - (a) for every $\mathcal{D} \subseteq \mathcal{C}$
 - i. for every $d \in \mathcal{D}$, $opt_{\mathcal{D},d}(i) \leftarrow cost(B_i, d) + \min(opt_{\mathcal{D},d}(i-1), opt_{\mathcal{D}\setminus\{d\}}(i-1))$
 - ii. $opt_{\mathcal{D}}(i) \leftarrow \min_{d \in \mathcal{D}} opt_{\mathcal{D},d}(i)$.
3. return $opt_{\mathcal{C}}(n_b)$

Computing $cost(B_i, d)$ for all $d \in \mathcal{C}$ and $i = 1, \dots, n_b$ can be done in $O(n_c \cdot n)$ time. Each of the n_b iterations of the algorithms requires $O(n_c \cdot 2^{n_c})$ time. So the running time of the above algorithm is $O(n_c(n_b 2^{n_c} + n))$.

4.2 Enhanced algorithm

The running time of the above algorithm does not improve even when the input string is already convex. However, for the weighted model, we can modify the algorithm so that its running time on convex or nearly convex strings is substantially smaller. For this we need the following definitions for a coloring C of a tree or a string. A color d is a *good color* (for the coloring C) if $n_b(C, d) = 1$, and the corresponding unique d -block is a *good block*. If $n_b(C, d) > 1$ then d is a *bad color* and all the d -blocks are *bad blocks*. The proof of the following claim is similar to that of Claim 3.1, and is omitted.

Claim 4.2 *Let (S, C) be a colored string, and let C^* be an optimal convex recoloring of C . Then each sequence of consecutive good blocks of C is either completely retained or completely overwritten by C^* ; in the latter case it is completely overwritten by a single bad color.*

Claim 4.2 above gives rise to the following improvement: coalesce each maximal sequence of consecutive good blocks into a single good block with a special color \hat{d} . Run a similar dynamic programming algorithm, modified to allow the output optimal convex coloring to have any number of \hat{d} -blocks.

Let n_c^* be the number of bad colors and n_b^* the number of bad blocks. Since there are at most $n_b^* + 1$ \hat{d} -blocks, the improved algorithm has running time of $O(n_c^*(n_b^* 2^{n_c^*} + n))$. In particular, for each fixed value of n_c^* the running time is polynomial in the input size.

4.3 Non-uniform Optimal Convex Recoloring of Binary Trees

We present below an algorithm for optimal convex recoloring of binary trees. The algorithm advances along the tree similarly to Sankoff's algorithm for small maximum weighted parsimony [17].

First, we root the tree at some vertex r (which can be a new vertex, obtained by splitting an edge in the original tree). Let $T(v)$ be the subtree rooted at v . A convex recoloring of $T(v)$ denotes a convex recoloring of the colored subtree $(T(v), C|_{V(T(v))})$. Starting from the leaves and advancing upwards towards the root, the algorithm computes for each vertex v , each set of colors $\mathcal{D} \subseteq \mathcal{C}$ and each color $d \in \mathcal{D}$, the value $opt_{\mathcal{D},d}(v)$, which is the minimal possible cost of a (convex) coloring of $T(v)$ which (i) uses only colors from \mathcal{D} and (ii) colors vertex v by color d . Let $opt_{\mathcal{D}}(v)$ be $\min_{d \in \mathcal{D}} opt_{\mathcal{D},d}(v)$. Then $opt_{\mathcal{C}}(r)$ is the cost of an optimal coloring of T . The recursive computation of $opt_{\mathcal{D},d}(v)$ is based on the following equality:

Claim 4.3 *For an internal vertex v , let v_l and v_r be v 's left and right children respectively. Then, the following recurrence relation holds:*

$$opt_{\mathcal{D},d}(v) = cost(v, d) + \min_{\mathcal{D}' \subseteq \mathcal{D} \setminus \{d\}} (\min(opt_{\mathcal{D}'}(v_r), opt_{\mathcal{D}' \cup \{d\},d}(v_r)) + \min(opt_{\mathcal{D} \setminus (\mathcal{D}' \cup \{d\})}(v_l), opt_{\mathcal{D} \setminus \mathcal{D}',d}(v_l))) \quad (1)$$

Proof. Let \hat{C} be a convex coloring of $T(v)$ s.t. (a) \hat{C} uses only colors from \mathcal{D} , (b) $\hat{C}(v) = d$, and (c) \hat{C} is of minimum possible cost among all colorings satisfying (a) and (b).

A direct construction shows that there is a convex recoloring of $T(v)$ which satisfies (a)-(b) above and its cost equals the right hand side of Equation (1). Hence $cost(\hat{C})$ cannot be larger than the right hand side of Equation (1). It remains to prove that $cost(\hat{C})$ is also not smaller than this value.

Let \mathcal{C}_l and \mathcal{C}_r be the color sets used by \hat{C} on $T(v_l)$ and $T(v_r)$ respectively. For \hat{C} to be convex, we must have that if $d \in \mathcal{C}_l$ (\mathcal{C}_r resp.) then $\hat{C}(v_l)$ ($\hat{C}(v_r)$ resp) must be d . Let $\mathcal{D}_l = \mathcal{C}_l \setminus \{d\}$ and $\mathcal{D}_r = \mathcal{C}_r \setminus \{d\}$. Then, by definition of the cost function

$$cost(\hat{C}) = cost(v, d) + cost(\hat{C}|_{V(T(v_r))}) + cost(\hat{C}|_{V(T(v_l))}) \geq cost(v, d) + (\min(opt_{\mathcal{D}_r}(v_r), opt_{\mathcal{D}_r \cup \{d\},d}(v_r)) + \min(opt_{\mathcal{D}_l}(v_l), opt_{\mathcal{D}_l \cup \{d\},d}(v_l)))$$

The proof is completed by substituting \mathcal{D}_r for \mathcal{D}' in equation (1). ■

The above claim leads to the straightforward dynamic programming algorithm. In order to compute $opt_{\mathcal{D},d}(v)$ we only need the required values at v_l and v_r to be defined. This can be achieved by post order visit in a DFS traversal in T' , starting at r . At a vertex v we need to find $opt_{\mathcal{D},d}(v)$ for every $\mathcal{D} \subseteq \mathcal{C}$ and $d \in \mathcal{D}$; for this we need to take the minimum over all \mathcal{D}' as in Equation (1). This yields $O(n_c 3^{n_c})$ operation per vertex; in total $O(n \cdot n_c 3^{n_c})$.

Note: The algorithm can be generalized to trees of bounded degree $\Delta > 3$, with the higher complexity $O(nc\Delta^{n_c})$.

We conclude this section by presenting a simpler linear time algorithm for optimal recoloring of a tree by two colors d_1, d_2 . For this, we compute for $i = 1, 2$ the minimal cost convex recoloring

C_i which sets the color of the root to d_i (i.e. $C_i(r) = d_i$). The required optimal convex recoloring is either C_1 or C_2 . The computation of C_1 can be done as follows:

Compute for every edge $e = (u \rightarrow v)$ a cost defined by

$$\text{cost}(e) = \sum_{v' \in T(v)} \text{cost}(v', d_2) + \sum_{v' \notin T(v)} \text{cost}(v', d_1)$$

This can be done by one post order traversal of the tree. Then, select the edge $e^* = (u_0 \rightarrow v_0)$ which minimizes this cost, and set $C_1(w) = d_2$ for each $w \in T(v_0)$, and $C_1(w) = d_1$ otherwise.

5 Fixed Parameter Tractable Recoloring Algorithms

A fixed parameter tractable (FPT) algorithm for optimal convex recoloring is an algorithm which receives as an input a colored tree on n vertices, (T, C) , and an integer k , and decides whether $\text{OPT}(T, C) \leq k$ in $p(n)f(k)$ time, for some polynomial p and arbitrary function f (see [6]). It is easy to see that an FPT recoloring algorithm exists iff there is an algorithm which finds an optimal recoloring of the input coloring C in $p(n) \cdot f(k)$ time for p and f as above, where k is the value of the optimal recoloring of C . In this section we first present an $O(k^2 \cdot 2^{2k+4} + 2kn)$ convex recoloring algorithm for strings, and an $O(2^{3k\Delta} \cdot n^2)$ convex recoloring of trees with bounded degree Δ , where in both cases k denotes the cost of the optimal recoloring. In particular, we prove that for each fixed Δ there is an FPT recoloring algorithm for the set of trees with bounded degree Δ . The complexity analyses of both algorithms are based on Claim 5.1 below.

As in Section 4.2, we denote by n_c^* and n_b^* the numbers of bad colors and bad blocks of the given input coloring C .

Claim 5.1 *Let (T, C) be a colored tree, where T is of bounded degree Δ . Then for every convex recoloring C' of C it holds that*

$$\text{cost}_C(C') \geq \frac{n_b^* - n_c^*}{\Delta} \geq \frac{n_b^*}{2\Delta}.$$

Proof. By definitions, the number of violations of the coloring C is $\sum_{d \in \mathcal{C}} (n_b(d) - 1) = n_b^* - n_c^*$. To prove the left inequality, we first observe that overwriting a single vertex v of degree at most Δ reduces the number of violations by at most Δ . [The maximum is obtained when v has Δ neighbors u_1, \dots, u_Δ , $C(v) = d$ for some bad color d , and $C(u_i) = d', i = 1, \dots, \Delta$. In this case setting $C'(v) = d'$ reduces the number of d' -violations by $\Delta - 1$, and the number of d violations by one - a total of Δ - see Figure 12]. Since C' must remove all the violations of C , we have that $\text{cost}_C(C') \geq \frac{n_b^* - n_c^*}{\Delta}$. The second inequality follows since $n_b^* \geq 2n_c^*$ (each bad color has at least two blocks), and hence $n_b^* - n_c^* \geq \frac{n_b^*}{2}$. ■

By substituting $\Delta = 2$ in Claim 5.1 above, we conclude that the improved algorithm of Section 4.2 provides an FPT algorithm for string recoloring: Since, by that claim, $n_b^* \leq 4k$, and hence also $n_c^* \leq 2k$, the improved algorithm finds a minimal cost recoloring in $O(n_c^*(n_b^* 2^{n_c^*} + n)) \leq O(k^2 \cdot 2^{2k+4} + 2kn)$ time. However, this FPT string recoloring algorithm is not generalizable to

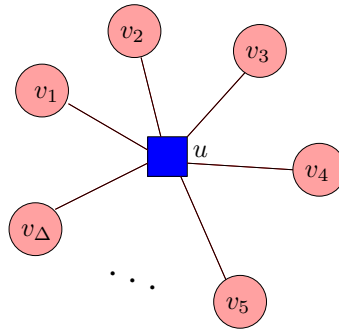


Figure 12: Changing u 's color to red (circle) reduces the number of \circ violations by $\Delta - 1$ and the number of \square violations by 1.

trees, since claims 3.1 and 4.2 are not valid for trees: as shown in Figure 13, an optimal recoloring of a tree may be forced to overwrite only a part of a block. As a result, the idea of treating all

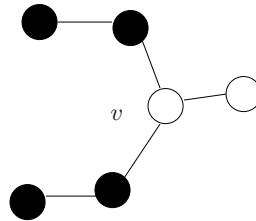


Figure 13: The white good block is only partially overwritten in the optimal coloring.

good colors as one special color, as done in Section 4.2 for strings, cannot be applied to trees.

In the next section we present an algorithm which finds an optimal convex recoloring of a tree in $O(2^{2k\Delta}n^2)$ time, where Δ is the maximal degree of a vertex in the input tree, and k is the value of an optimal recoloring. Specifically, this is an FPT recoloring algorithm for trees of bounded degree.

5.1 FPT Recoloring Algorithm for Bounded Degree Trees

For presenting the algorithm we need few more definitions. Let C' be a recoloring of C and let B be a block of C . We say that B is *partially retained* by C' if there is at least one vertex $v \in B$ for which $C(v) = C'(v)$. Let $\mathcal{B}^* = \{b_1, \dots, b_{n_b^*}\}$ be the set of bad blocks of C . *Signature*(C'), the *signature* of a recoloring C' of C , is defined by:

$$\text{Signature}(C') = \{B \in \mathcal{B}^* : B \text{ is partially retained by } C'\}$$

i.e., *Signature*(C') includes all the bad blocks which are partially retained by C' . Note that there are at most $2^{|\mathcal{B}^*|} = 2^{n_b^*}$ distinct signature. Our FPT recoloring algorithm is based on an $O(n^2)$ algorithm *OPT_SIGNATURE* which, given the input coloring and a subset $\mathcal{B} \subseteq \mathcal{B}^*$, decides

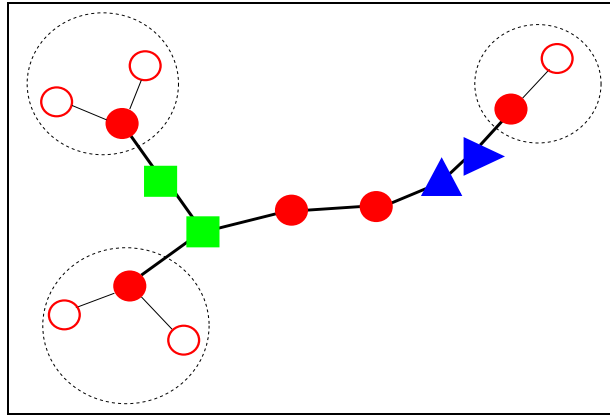


Figure 14: The red (circles) core defined by the three leaf blocks in the core. Filled shapes (vertices) are in the red core.

whether a convex recoloring whose signature is \mathcal{B} exists, and in this case finds a minimal cost recoloring, $C_{\mathcal{B}}$, having this signature. Since by Claim 5.1 $n_b^* \leq 2k\Delta$, executing *OPT_SIGNATURE* for all subsets \mathcal{B} of \mathcal{B}^* gives an $O(2^{2k\Delta}n^2)$ optimal recoloring algorithm, where k and Δ are as above.

In the rest of this section we describe the algorithm *OPT_SIGNATURE* for some fixed set $\mathcal{B} \subseteq \mathcal{B}^*$ of bad blocks. A color d is *bad for* \mathcal{B} if \mathcal{B} contains at least two d -blocks, and in this case each d -block in \mathcal{B} is a bad block for \mathcal{B} . For each color d which is bad for \mathcal{B} , the d -core of \mathcal{B} is the minimal subtree $core(d, \mathcal{B})$ of the input tree T which intersects each of the d -blocks in \mathcal{B} (see Figure ??). Observe that if C' is a convex coloring whose signature is \mathcal{B} , then $C'(v) = d$ for every vertex v in $core(d, \mathcal{B})$.

In its first stage, algorithm *OPT_SIGNATURE* constructs $core(d, \mathcal{B})$ for each color d which is bad for \mathcal{B} . If $core(d, \mathcal{B}) \cap core(d', \mathcal{B}) \neq \emptyset$ for some colors d, d' which are bad for \mathcal{B} , then no convex coloring whose signature is \mathcal{B} exists, and *OPT_SIGNATURE* stops; else, it attempts to construct the desired coloring $C_{\mathcal{B}}$ as follows.



Figure 15: The three right hand vertices are in the signature. The signature is invalid since the right red (second from right) is totally overwritten by the black core.

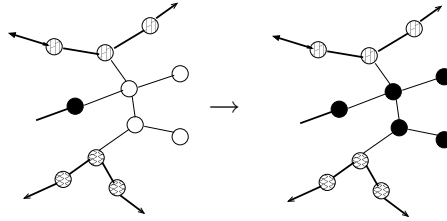


Figure 16: A block in the signature (left side) that is partially overwritten by two other cores (partially filled vertices), is maximally expanded (right side).

1. [Fix the colors of cores] For each color d which is bad for \mathcal{B} , set $C_{\mathcal{B}}(v) = d$ for each $v \in core(d, \mathcal{B})$.
 If at the end of this stage some block $B \in \mathcal{B}$ is completely overwritten,^a or some vertex in a block $B \notin \mathcal{B}$ is retained, then stop - there is no convex recoloring of C whose signature is \mathcal{B} (see Figure 15). Else, complete the setting of the recoloring $C_{\mathcal{B}}$ in the following steps.
 2. [Expand cores by retained vertices] $C_{\mathcal{B}}$ expands each d -core to include as many as possible vertices of d -blocks of \mathcal{B} (see Figure 16).
 3. [Expand other blocks in \mathcal{B} by retained vertices] Let F be the union of all d -cores. For each block $B \in \mathcal{B}$ which is not bad for \mathcal{B} , $C_{\mathcal{B}}$ retains the vertices of a maximal subset of B which is included in a connected component in $T \setminus F$ (see Figure 17).
 4. [Overwrite the remaining vertices] $C_{\mathcal{B}}$ overwrites the remaining vertices in an arbitrary manner that preserves convexity (e.g. by expanding neighboring blocks).
- ^aNote that this can happen only if B is not bad for \mathcal{B} .

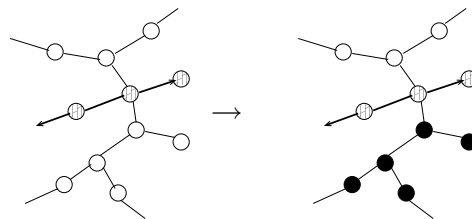


Figure 17: The maximal sub block in a good block for a signature, is expanded.

Theorem 5.2 *On an input coloring C of a tree of n vertices, the algorithm above finds an optimal recoloring of C in $O(2^{2k\Delta}n^2)$ time, where Δ is the maximal degree of the input tree and k is the cost of an optimal convex recoloring of C .*

Proof. (an outline)

Since $n_b \leq 2k\Delta$, it suffices to show that the complexity of the procedure *OPT_SIGNATURE* above is $O(n^2)$, and that it finds an optimal recoloring whose signature is \mathcal{B} , or informs that no such recoloring exists. The complexity bound is immediate, so we prove correctness. During all the stages of the algorithm, the constructed coloring $C_{\mathcal{B}}$ has at most one block of each color - hence $C_{\mathcal{B}}$ is convex. Stages 1 and 4 guarantee that each block in \mathcal{B} is partially retained and each block not in \mathcal{B} is completely overwritten, hence $Signature(C_{\mathcal{B}}) = \mathcal{B}$. Stages 2-3 guarantee that $C_{\mathcal{B}}$ retains a maximal possible number of vertices, hence is optimal. ■

6 Approximation Algorithms for Convex Recoloring.

Let AL be an algorithm which receives as an input a colored tree (T, C) and outputs a convex recoloring of (T, C) , and let $AL(T, C)$ be the cost of the convex recoloring output by AL on input (T, C) . We say that AL is an r -approximation algorithm for the convex tree recoloring problem if for all inputs (T, C) it holds that $AL(T, C)/OPT(T, C) \leq r$ [10, 12]. An optimization problem is *fully p -approximable* [16], or has a *fully polynomial time approximation scheme* [10], if for each ε there is an algorithm which provides an ε -approximation to it in time which is polynomial in n and $\frac{1}{\varepsilon}$. We first note that our NP completeness proof implies that if $P \neq NP$ then the problem is not fully p -approximable: This follows by the fact that the value of the optimal solution is bounded by the input size, using an observation in [16].

In this section we present polynomial time algorithms which provide approximation by constant factors (2 for strings, 3 for trees), for the unweighted and weighted versions of the problem.

6.1 Lower Bounds via Penalties

Let (T, C) be a colored tree. For a color d and $U \subseteq V(T)$ let:

$$penalty_{C,d}(U) = |\{v : v \in U \text{ and } C(v) \neq d\}| + |\{v : v \notin U \text{ and } C(v) = d\}|,$$

When the vertices in U induce a subtree, the penalty associated with d and U is the number of vertices which must be overwritten when U is the d -block in some convex coloring, either for being in U and having color different from d , or for not being in U and having color d .

We now associate a penalty with a convex recoloring, that sums the penalties of every colored block: Let C' be a convex recoloring of C . Then:

$$penalty_C(C') = \sum_{d \in \mathcal{C}} penalty_{C,d}(C'^{-1}(d))$$

Figure 18 depicts the calculation of a penalty associated with a convex recoloring C' of C .

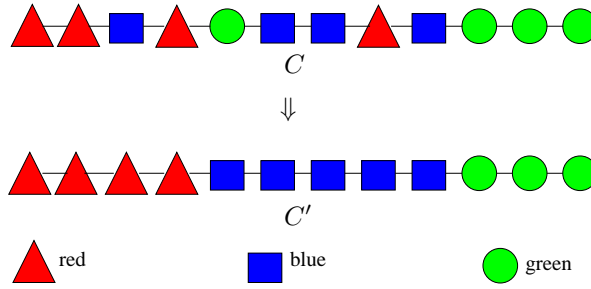


Figure 18: C' is a convex recoloring for C which defines the following penalties: $p_{green}(C') = 1$, $p_{red}(C') = 2$, $p_{blue}(C') = 3$

In the sequel we assume that the input colored tree (T, C) is fixed, and omit it from the notations.

Claim 6.1 $penalty(C') = 2cost(C')$

Proof. From the definitions we have

$$\begin{aligned}
 penalty(C') &= \sum_{d \in \mathcal{C}} (|\{v \in V : C'(v) = d \text{ and } C(v) \neq d\}| + |\{v \in V : C'(v) \neq d \text{ and } C(v) = d\}|) \\
 &= 2 \sum_{d \in \mathcal{C}} |\{v \in V : C'(v) \neq d \text{ and } C(v) = d\}| = 2cost(C')
 \end{aligned}$$

■

As can be seen in Figure 18, $penalty(C') = 6$ while $cost(C') = 3$.

For each color d , p_d^* is the penalty of a block which minimizes the penalty for d :

$$p_d^* = \min\{penalty_d(V(T')) : T' \text{ is a subtree of } T\}$$

Corollary 6.2 For any recoloring C^* of C ,

$$\sum_{d \in \mathcal{C}} p_d^* \leq \sum_{d \in \mathcal{C}} penalty_d(C') = 2cost(C').$$

Proof. The inequality follows immediately from the definition of p_d^* , and the equality from Claim 6.1. ■

Corollary 6.2 above provides a lower bound on the cost of convex recoloring of trees. It can be shown that this lower bound can be quite poor for trees, that is: $OPT(T, C)$ can be considerably larger than the sum $\sum_{d \in \mathcal{C}} p_d^*$. However in the next section we show that this bound can be used to obtain a polynomial time 2-approximation for convex recoloring of strings.

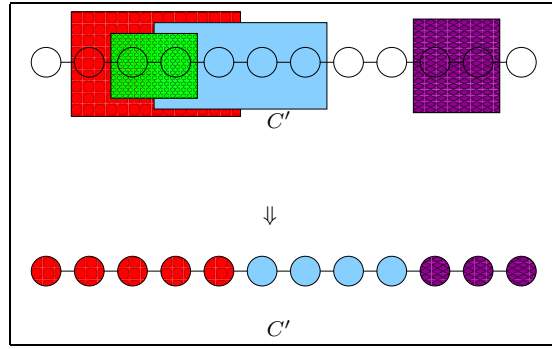


Figure 19: The upper part of the figure shows the optimal blocks on the string and the lower part shows the coloring returned by the algorithm.

6.2 A 2-Approximation Algorithm for a String

Let an input colored string (S, C) , where $S = (v_1, \dots, v_n)$, be given. For $1 \leq i \leq j \leq n$, $S[i, j]$ is the substring $(v_i, v_{i+1}, \dots, v_j)$ of S . The algorithm starts by finding for each d a subsequence $B_d = S[i_d, j_d]$ for which $penalty_d(S[i_d, j_d]) = p_d^*$. It is not hard to verify that B_d consists of a subsequence of consecutive vertices in which the difference between the number of d -vertices and number of other vertices (i.e. $|B_d \cap C^{-1}(d)| - |B_d \setminus C^{-1}(d)|$) is maximized, and thus B_d can be found in linear time. We say that a vertex v_i is *covered* (by the set of blocks $\{B_d : d \in \mathcal{C}\}$) if $v_i \in B_d$ for at least one color d ; if v_i is not covered then it is *free*.

We describe below a linear time algorithm which, given the blocks B_d , defines a convex coloring \hat{C} so that $cost(\hat{C}) \leq \sum_d p_d^*$, which by Corollary 6.2 is a 2-approximation to a minimal convex recoloring of C .

\hat{C} is constructed by performing one scan of S from left to right. The scan consists of at most c stages, where stage j defines the j -th block of \hat{C} , as follows.

$\hat{C}(v_1) = d_1$, where d_1 is one of the colors of the leftmost covered vertex. Thus d_1 is the color of the first (leftmost) block of \hat{C} , to be denoted \hat{B}_{d_1} . Then the scan of S from left to right begins; when vertex v_i is scanned, it is added to \hat{B}_{d_1} if it belongs to B_{d_1} or it is free - otherwise v_i is taken to be the first vertex in the second block \hat{B}_{d_2} , where d_2 is any of the colors that cover v_{i_2} .

In general, block $\hat{B}_{d_{j+1}}$ starts at the leftmost covered vertex $v_{i_{j+1}}$ which does not belong to B_{d_j} , and its color d_{j+1} is one of the colors that cover $v_{i_{j+1}}$; if there is no such vertex $v_{i_{j+1}}$, \hat{B}_j terminates at v_n , the rightmost vertex of S , and the construction terminates.

Observe that during the construction the following invariant is kept for all k :

$$\bigcup_{j=1}^k B_{d_j} \subseteq \bigcup_{j=1}^k \hat{B}_{d_j},$$

which guarantees that $d_j \neq d_{j'}$ for $j \neq j'$, hence \hat{C} is a convex coloring of S . Thus it remains to prove

Lemma 1 $cost(\hat{C}) \leq \sum_{d \in \mathcal{C}} p_d^*$.

Proof. We show that each vertex which is overwritten by \hat{C} contributes one to the above sum. Let v_i be a such a vertex. Then $C(v_i) = d$ and $\hat{C}(v_i) = d'$ for some $d' \neq d$. Then if $i \in B_{d'}$ then it was counted for in $p_{d'}^*$. Otherwise v_i is free, and hence $v_i \notin B_d$, and hence v_i is counted for by p_d^* . ■

6.3 3-approximation algorithm for a tree

In this section we present a polynomial time algorithm which approximates the minimal convex coloring of a tree by factor three. The algorithm is defined for the weighted version of the problem. Recall that in this version the input is a triplet (T, C, w) , where w is a nonnegative weight function and C is a (possibly partial) coloring whose domain is the set $support(w) = \{v \in V : w(v) > 0\}$.

Our approximation algorithm makes use of the local ratio technique, which is useful for approximating optimization covering problems such as vertex cover, dominating set, minimum spanning tree, feedback vertex set and more [4, 2, 3]. We hereafter describe it briefly:

The input to the problem is a triplet $(V, f : 2^V \rightarrow \{0, 1\}, w : V \rightarrow \mathbb{R}^+)$, and the goal is to find a subset $X \subseteq V$ such that $f(X) = 1$ and $w(X)$ is minimized, i.e. $w(X) = OPT(V, f, w) = \min_{Y \subseteq V \& f(Y)=1} w(Y)$ (in our context V is the set of vertices, and $f(Y) = 1$ if Y is a cover). The local ratio principle is based on the following observation (see e.g. [3]):

Observation 6.3 For every two weight functions w_1, w_2 :

$$OPT(V, f, w_1) + OPT(V, f, w_2) \leq OPT(V, f, w_1 + w_2)$$

Now, given our initial weight function w , we select w_1, w_2 s.t. $w_1 + w_2 = w$ and $|support(w_1)| < |support(w)|$. We first apply the algorithm to find an r -approximation to (V, f, w_1) (in particular, if $V \setminus support(w_1)$ is a cover, then it is an optimal cover to (V, f, w_1)). Let X be the solution returned for (V, f, w_1) , and assume that $w_1(X) \leq r \cdot OPT(V, f, w_1)$. If we could also guarantee that $w_2(X) \leq r \cdot OPT(V, f, w_2)$ then by Observation 6.3 we are guaranteed that X is also an r -approximation for $(V, f, w_1 + w_2 = w)$. The original property, introduced in [4], which was used to guarantee that $w_2(X) \leq r \cdot OPT(V, f, w_2)$ is that w_2 is r -effective, that is: for every X s.t. $f(X) = 1$ it holds that $w_2(X) \leq r \cdot OPT(V, f, w_2)$ (note that if $f(V) = 1$, the above is equivalent to requiring that $w_2(V) \leq r \cdot OPT(V, f, w_2)$).

Theorem 6.4 [4] Given a cover X s.t. $w_1(X) \leq r \cdot OPT(V, f, w_1)$. If w_2 is r -effective, then $w(X) = w_1(X) + w_2(X) \leq r \cdot OPT(V, f, w)$.

The technique of Theorem 6.4 was extended in several ways (see e.g. [2, 3]). We now show two applications of the original local ratio technique to obtain a 3-approximation algorithm for the convex weighted string problem and a 4-approximation algorithm for the convex weighted tree problem.

3-string-APPROX:

Given an instance to convex weighted string problem (S, C, w) :

1. If $V \setminus \text{support}(w)$ is a cover then $X \leftarrow V \setminus \text{support}(w)$. Else:
2. Find 3 vertices $x, y, z \in \text{support}(w)$ s.t. $C(x) = C(z) \neq C(y)$ and y lies between x and z .
 - (a) $\varepsilon \leftarrow \min\{w(x), w(y), w(z)\}$
 - (b) $w_2(v) = \begin{cases} \varepsilon & \text{if } v \in \{x, y, z\} \\ 0 & \text{otherwise.} \end{cases}$
 - (c) $w_1 \leftarrow w - w_2$
 - (d) $X \leftarrow \text{3-string-APPROX}(S, C|_{\text{support}(w_1)}, w_1)$

Note that if a (partial) coloring of a string is not convex then the condition in 2 must hold. It is also easy to see that w_2 is 3-effective, since any cover Y must contain at least one vertex from any triplet described in condition 2, hence $w_2(Y) \geq \varepsilon$ while $w_2(V) = 3\varepsilon$.

The above algorithm cannot serve for approximating convex tree coloring since in a tree the condition in 2 might not hold and yet the coloring is not convex. In the following algorithm we generalize this condition to one which must hold in any non-convex coloring of a tree, in the price of increasing the approximation ratio from 3 to 4.

4-tree-APPROX:

Given an instance to convex weighted tree problem (T, C, w) :

1. If $V \setminus \text{support}(w)$ is a cover then $X \leftarrow V \setminus \text{support}(w)$. Else:
2. Find two pairs of (not necessarily distinct) vertices (x_1, x_2) and (y_1, y_2) in $\text{support}(w)$ s.t. $C(x_1) = C(x_2) \neq C(y_1) = C(y_2)$, and the path connecting x_1 and x_2 intersects the path connecting y_1 and y_2 :
 - (a) $\varepsilon \leftarrow \min\{w(x_i), w(y_i)\}, i = \{1, 2\}$
 - (b) $w_2(v) = \begin{cases} \varepsilon & \text{if } v \in \{x_1, x_2, y_1, y_2\} \\ 0 & \text{otherwise.} \end{cases}$
 - (c) $w_1 \leftarrow w - w_2$
 - (d) $X \leftarrow \text{4-tree-APPROX}(S, C|_{\text{support}(w_1)}, w_1)$

The algorithm is correct since if there are no two pairs as described in step 2, then $V \setminus \text{support}(w)$ is a cover. Also, it is easy to see that w_2 is 4-effective. Hence the above algorithm returns a cover with weight at most $4 \cdot \text{OPT}(T, C, w)$.

We now describe algorithm 3-tree-APPROX. Informally, the algorithm uses an iterative method, in the spirit of the local ration technique, which approximates the solution of the input (T, C, w) by reducing it to (T', C', w_1) where $|\text{support}(w_1)| < |\text{support}(w)|$. Depending on the given input,

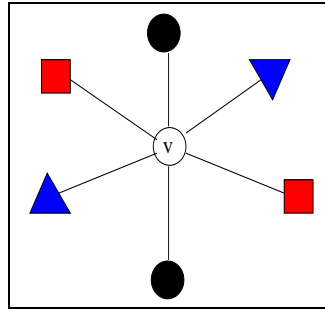


Figure 20: Case 2: a vertex v is contained in 3 different containers.

this reduction is either of the local ratio type (via an appropriate 3-effective weight function) or, the input graph is replaced by a smaller one which preserves the optimal solutions.

3-tree-APPROX(T, C, w)

On input (T, C, w) of a weighted colored tree, do the following:

1. If $V \setminus \text{support}(w)$ is a cover then $X \leftarrow V \setminus \text{support}(w)$. Else:
2. $(T', C', w_1) \leftarrow \text{REDUCE}(T, C, w)$. \The function *REDUCE* guarantees that $|\text{support}(w_1)| < |\text{support}(w)|$
 - (a) $X' \leftarrow \text{3-tree-APPROX}(T', C', w_1)$.
 - (b) $X \leftarrow \text{UPDATE}(X', T)$. \The function *UPDATE* guarantees that if X' is a 3-approximation to (T', C', w_1) , then X is a 3-approximation to (T, C, w) .

Next we describe the functions *REDUCE* and *UPDATE*, by considering few cases. In the first two cases we employ the local ratio technique.

Case 1: $\text{support}(w)$ contains three vertices x, y, z such that y lies on the path from x to z and $C(x) = C(z) \neq C(y)$.

In this case we use the same reduction of 3-string-APPROX: Let $\varepsilon = \min\{w(x), w(y), w(z)\} > 0$. Then $\text{REDUCE}(T, C, w) = (T, C|_{\text{support}(w_1)}, w_1)$, where $w_1(v) = w(v)$ if $v \notin \{x, y, z\}$, else $w_1(v) = w(v) - \varepsilon$. The same arguments which implies the correctness of 3-string-APPROX implies that if X' is a 3-approximation for (T', C', w_1) , then it is also a 3-approximation for (T, C, w) , thus we set $\text{UPDATE}(X', T) = X'$.

Case 2: Not Case 1, and T contains a vertex v such that $v \in T_{d_1, C} \cap T_{d_2, C} \cap T_{d_3, C}$ for three distinct colors d_1, d_2 and d_3 (see Figure 20).

In this case we must have that $w(v) = 0$ (else Case 1 would hold), and there are three *designated pairs* of vertices $\{x_1, x_2\}, \{y_1, y_2\}$ and $\{z_1, z_2\}$ such that $C(x_i) = d_1, C(y_i) = d_2, C(z_i) = d_3 (i = 1, 2)$, and v lies on each of the three paths connecting these three pairs (see Figure 20). We set $\text{REDUCE}(T, C, w) = (T, C|_{\text{support}(w_1)}, w_1)$, where w_1 is defined as follows.

Let $\varepsilon = \min\{w(x_i), w(y_i), w(z_i) : i = 1, 2\}$. Then $w_1(v) = w(v)$ if v is not in one of the designated

pairs, else $w_1(v) = w(v) - \varepsilon$. Finally, any cover for (T, C) must contain at least two vertices from the set $\{x_i, y_i, z_i : i = 1, 2\}$, hence $w - w_1 = w_2$ is 3-effective, and by the local ratio theorem we can set $UPDATE(X', T) = X'$.

Case 3: Not Cases 1 and 2.

Root T at some vertex r and for each color d let r_d be the root of the subtree $T_{d,C}$. Let r_{d_0} (for some color d_0) be a farthest root from r . Let \bar{T} be the subtree of T rooted at r_{d_0} , and let $\hat{T} = T \setminus \bar{T}$ (see Figure 21). By the definition of r_{d_0} , no vertex in \hat{T} is colored by d_0 , and since Case 2 does not hold, there is a unique color d' so that $\{d_0\} \subseteq C(V(\bar{T})) \subseteq \{d_0, d'\}$.

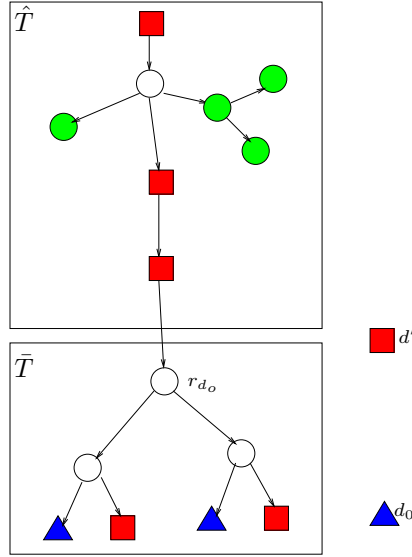


Figure 21: Case 3: Not case 1 nor 2. \bar{T} is the subtree rooted at r_{d_0} and $\hat{T} = T \setminus \bar{T}$.

Sub case 3a: $C(V(\bar{T})) = \{d_0\}$ (see Figure 22).

In this case, $T_{d_0,C} \cap T_{d,C} = \emptyset$ for each color $d \neq d_0$, and for each optimal solution X it holds that $X \cap V(\bar{T}) = \emptyset$. We set $REDUCE(T, C, w) \leftarrow (\hat{T}, C|_{V(\hat{T})}, w|_{V(\hat{T})})$. The 3-approximation X' to (T', C', w_1) is also a 3-approximation to (X, C, w) , thus $UPDATE(X', T) = X'$.

We are left with the last case.

Sub case 3b: $r_{d_0} \in T_{d_0,C} \cap T_{d',C}$. See Figure 23.

Observe that in this case we have $w(r_{d_0}) = 0$ and $|support(w) \cap V(\bar{T})| \geq 3$, since $V(\bar{T})$ must contain at least two vertices colored d_0 and at least one vertex colored d' . Figure 23 illustrates this case.

Similarly to the proof of Observation 2.1, we obtain:

Observation 6.5 *There is an optimal convex coloring C' which satisfies the following: $C'(v) \neq d_0$ for any $v \in V(\hat{T})$, and $C'(v) \in \{d_0, d'\}$ for any $v \in V(\bar{T})$.*

The function $REDUCE$ in sub case 3b is based on the following observation: Let C' be any optimal recoloring of T satisfying Observation 6.5, and let s be the parent of r_{d_0} in T . Then $C'|_{V(\bar{T})}$,

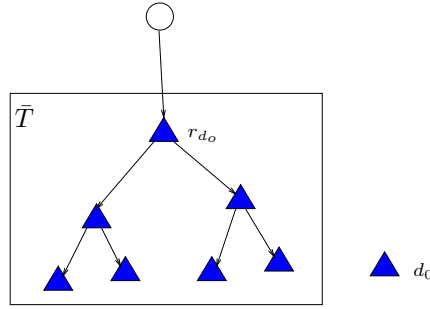


Figure 22: Case 3a: No vertices of \hat{T} are colored by d' .

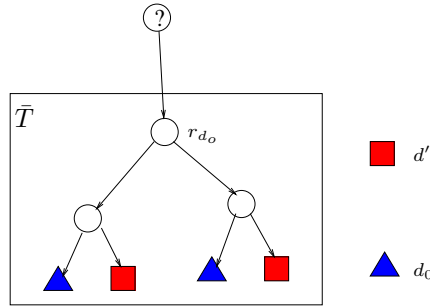


Figure 23: Case 3b: $r_{d_0} \in T_{d_0} \cap T_{d'}$

the restriction of the coloring C' to the vertices of \bar{T} , depends only on whether $C'|_{V(\hat{T})}$ contains a d' -block, and on the location of this d' -block. Specifically, $C'|_{V(\bar{T})}$ must be one of the three colorings of $V(\bar{T})$, C_{high} , C_{medium} and C_{min} , according to whether \hat{T} has a d' -block, and in this case whether this block is disjoint from s . The details follow:

1. $C'|_{\hat{T}}$ contains a d' -block which is disjoint from s . Then it must be the case that C' colors all the vertices in $V(\bar{T})$ by d_0 . This coloring of \bar{T} is denoted as C_{high} .
2. $C'|_{\hat{T}}$ contains a d' -block which includes s . Then $C'|_{\bar{T}}$ is a coloring of minimal possible cost of \bar{T} which either equals C_{high} (i.e. colors all vertices by d_0), or otherwise sets $C_{medium}(r_{d_0}) = d'$. This coloring of \bar{T} is called C_{medium} .
3. $C'|_{\hat{T}}$ does not use the color d' . Then $C'|_{\bar{T}}$ must be an optimal convex recoloring of \bar{T} by the two colors d_0, d' .

The function *REDUCE* in Sub case 3b modifies the tree T by replacing \bar{T} by a subtree \bar{T}_0 with only 2 vertices, r_{d_0} and v_0 , which encodes the three colorings $C_{high}, C_{medium}, C_{min}$. Specifically, $REDUCE(T, C, w) = (T', C', w_1)$ where (see Figure 24):

- T' is obtained from T by replacing the subtree \bar{T} by the subtree \bar{T}_0 which contains two vertices: a root r_{d_0} with a single descendant v_0 .

- $C'(v) = C(v)$ for each $v \in V(\hat{T})$; $C'(r_{d_0}) = d_0$ and $C'(v_0) = d'$.
- $w_1(v) = w(v)$ for each $v \in V(\hat{T})$. For r_{d_0} and v_0 w_1 is defined as follows: $w_1(r_{d_0}) = \text{cost}(C_{\text{medium}}) - \text{cost}(C_{\text{min}})$ and $w_1(v_0) = \text{cost}(C_{\text{high}}) - \text{cost}(C_{\text{min}})$

Figure 24 illustrates *REDUCE* for case 3b. In the figure, C_{high} requires overwriting all d' vertices and therefore costs 3, C_{medium} requires overwriting one d_0 vertex and costs 2 and C_{min} is the optimal coloring for \bar{T} with cost 1. The new subtree \bar{T}_0 reflects these weight with $w(r_{d_0}) = C_{\text{high}} - C_{\text{min}} = 2$ and $w(v_0) = C_{\text{medium}} - C_{\text{min}} = 1$.

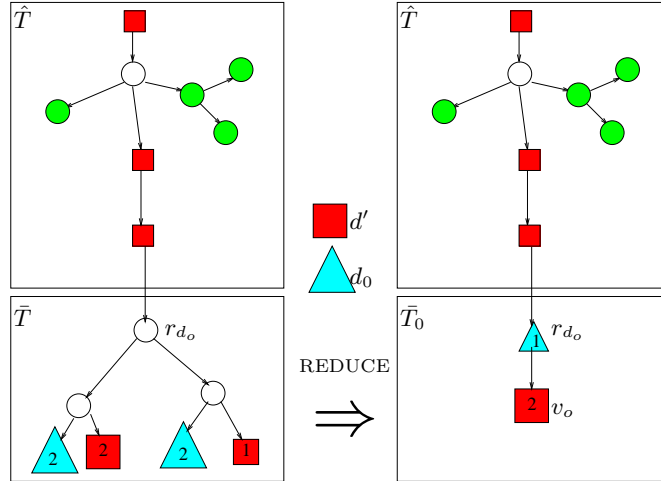


Figure 24: *REDUCE* of case 3b: \bar{T} is replaced with \bar{T}_0 where $w(r_{d_0}) = C_{\text{high}} - C_{\text{min}} = 2$ and $w(v_0) = C_{\text{medium}} - C_{\text{min}} = 1$.

Claim 6.6 $OPT(T', C', w_1) = OPT(T, C, w) - \text{cost}(C_{\text{min}})$.

Proof. We first show that $OPT(T', C', w_1) \leq OPT(T, C, w) - \text{cost}(C_{\text{min}})$. Let C^* be an optimal recoloring of C satisfying Observation 6.5, and let $X^* = \mathcal{X}(C^*)$. By the discussion above, we may assume that $C^*|_{V(\bar{T})}$ has one of the forms C_{high} , C_{medium} or C_{min} . Thus, $X^* \cap V(\bar{T})$ is either $\mathcal{X}(C_{\text{high}})$, $\mathcal{X}(C_{\text{medium}})$ or $\mathcal{X}(C_{\text{min}})$. We map C^* to a coloring C' of T' as follows: for $v \in V(\hat{T})$, $C'(v) = C^*(v)$. C' on r_{d_0} and v_0 is defined as follows:

- If $C^*|_{V(\bar{T})} = C_{\text{high}}$ then $C'(r_{d_0}) = C'(v_0) = d_0$, and $\text{cost}(C') = w_1(v_0)$;
- If $C^*|_{V(\bar{T})} = C_{\text{medium}}$ then $C'(r_{d_0}) = C'(v_0) = d'$, and $\text{cost}(C') = w_1(r_{d_0})$;
- If $C^*|_{V(\bar{T})} = C_{\text{min}}$ then $C'(r_{d_0}) = d'$, $C'(v_0) = d_0$, and $\text{cost}(C') = 0$.

Note that in all three cases, $\text{cost}(C') = \text{cost}(C^*) - \text{cost}(C_{\text{min}})$.

The proof of the opposite inequality $OPT(T, C) - \text{cost}(C_{\text{min}}) \leq OPT(T', C')$ is similar. ■

Corollary 6.7 C^* is optimal recoloring of (T, C, w) iff C' is an optimal recoloring of (T', C', w_1) .

We now can define the *UPDATE* function for Sub case 3b: Let $X' = 3\text{-tree-APPROX}(T', C', w_1)$. Then X' is a disjoint union of the sets $\hat{X}' = X' \cap V(\hat{T})$ and $\bar{X}'_0 = X' \cap V(\bar{T}_0)$. Moreover, $\bar{X}'_0 \in \{\{r_{d_0}\}, \{v_0\}, \emptyset\}$. Then $X \leftarrow \text{UPDATE}(X') = \hat{X}' \cup \bar{X}'$, where \bar{X}' is $\mathcal{X}(C_{high})$ if $\bar{X}'_0 = \{r_{d_0}\}$, is $\mathcal{X}(C_{medium})$ if $\bar{X}'_0 = \{v_0\}$, and is $\mathcal{X}(C_{min})$ if $\bar{X}'_0 = \emptyset$. Note that $w(X) = w(X') + \text{cost}(C_{min})$.

We have the following inequalities:

$$\begin{aligned} w(X) &= w_1(X') + \text{cost}(C_{min}) \leq 3OPT(T', C') + \text{cost}(C_{min}) \\ &\leq 3(OPT(T', C') + \text{cost}(C_{min})) = 3OPT(T, C) \end{aligned}$$

6.3.1 Correctness and complexity

We now summarize the discussion of the previous section to show that the algorithm terminates and return a cover X which is a 3-approximation for (T, C, w) .

Let $(T = (V, E), C, w)$ be an input to 3-tree-APPROX. if $V \setminus \text{support}(w)$ is a cover then the returned solution is optimal. Else, in each of the cases *REDUCE* (T, C, w) reduces the input to (T', C', w_1) such that $|\text{support}(w_1)| < |\text{support}(w)|$, hence the algorithm terminates within at most $n = |V|$ iterations. Also, as detailed in the previous section, the function *UPDATE* guarantees that that if X' is a 3-approximation for (T', C', w_1) then X is a 3-approximation to (T, C, w) . Thus after at most n iterations the algorithm provides a 3-approximation to the original input. It remains to show that each iteration requires a polynomial time - which is obvious for all cases except, maybe, for Sub case 3b.

In Sub case 3b we need to compute C_{high} , C_{medium} and C_{min} . For this we need to compute an optimal recoloring of \bar{T} by the two colors d_0 and d' , and a minimal cost convex recoloring of \bar{T} by these two colors, given that r_{d_0} must be colored by d' . This can be done in linear time by the optimal recoloring algorithm for the case of two colors, presented at the end of Section 4.3.

The complexity of each iteration is $O(cn)$, and since there are at most n iterations, the overall complexity is $O(cn^2)$. Thus we have

Theorem 6.8 *Algorithm 3-tree-APPROX is a polynomial time 3-approximation algorithm for the minimum convex recoloring problem.*

7 Discussion and Future Work

In this work we studied the complexity of computing the distance from a given coloring of a tree or string to a convex coloring, motivated by the scenario of introducing a new character to an existing phylogenetic tree. We considered few natural definitions for that distance, and proved that the problem is NP-Hard in each of them. We then presented exact and approximate algorithms to solve this problem.

Few interesting research directions which suggest themselves are:

- Similarly to the generalization of the small parsimony question to the general one: When the number of colors is fixed, is there an efficient algorithm which computes a phylogenetic tree of minimum distance from a perfect phylogeny, where the distance is taken as the number of color changes needed to achieve perfect phylogeny? Note that, as in maximum parsimony, this problem is trivial for one character.
- Similarly to the above, but rather than bounding the number of colors, the bound now is on the number of color changes. This corresponds to a fixed parameter tractable algorithm for constructing an optimal tree.
- Are there FPT and/or constant ratio polynomial time approximation algorithms for the non-uniform version of this problem.
- Can our approximation ratios for strings or trees be improved.
- This is a more focused variant of the previous item. A problem has a *polynomial approximation scheme* [10, 12], or is *fully approximable* [16], if for each ε it can be ε -approximated in $p_\varepsilon(n)$ time for some polynomial p_ε . Are the weighted or unweighted version of the problem fully approximable, (or equivalently have a polynomial approximation scheme)?

8 Acknowledgments

Rami, Reuven, Mike Steel

References

- [1] R. Agrawala and D. Fernandez-Baca. Simple algorithms for perfect phylogeny and triangulating colored graphs. *International Journal of Foundations of Computer Science*, 7(1):11–21, 1996.
- [2] V. Bafna, P. Berman, and T. Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM J. on Discrete Mathematics*, 12:289–297, 1999.
- [3] R. Bar-Yehuda. One for the price of two: A unified approach for approximating covering problems. *Algorithmica*, 27:131–144, 2000.
- [4] R. Bar-Yehuda and S. Even. A local-ratio theorem for approximating the weighted vertex cover problem. *Annals of Discrete Mathematics*, 25:27–46, 1985.
- [5] H. Bodlaender, M. Fellows, and T. Warnow. Two strikes against perfect phylogeny. In *Proceedings of the 19th International Colloquium on Automata, Languages, and Programming*, Lecture Notes in Computer Science, pages 273–283. Springer Verlag, 1992.
- [6] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.

- [7] A. Dress and M.A. Steel. Convex tree realizations of partitions. *Applied Mathematics Letters*, 5(3):3–6, 1992.
- [8] D. Fernandez-Baca and J. Lagergren. A polynomial-time algorithm for near-perfect phylogeny. *SIAM Journal on Computing*, 32(5):1115–1127, 2003.
- [9] W. M. Fitch. A non-sequential method for constructing trees and hierarchical classifications. *Journal of Molecular Evolution*, 18(1):30–37, 1981.
- [10] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.
- [11] D. Gusfield. Efficient algorithms for inferring evolutionary history. *Networks*, 21:19–28, 1991.
- [12] D. S. Hochbaum, editor. *Approximation Algorithms for NP-Hard Problem*. PWS Publishing Company, 1997.
- [13] S. Kannan and T. Warnow. Inferring evolutionary history from DNA sequences (extended abstract). *SIAM J. Computing*, 23(3):713–737, 1994.
- [14] S. Kannan and T. Warnow. A fast algorithm for the computation and enumeration of perfect phylogenies when the number of character states is fixed. *SIAM J. Computing*, 26(6):1749–1763, 1997.
- [15] E. J. Kollar and C. Fisher. Tooth induction in chick epithelium: Expression of quiescent genes for enamel synthesis. *Science*, 207:993–995, 1980.
- [16] A. Paz and S. Moran. Non deterministic polynomial optimization problems and their approximability. *Theoretical Computer Science*, 15:251–277, 1981. Abridged version: Proc. of the 4th ICALP conference, 1977.
- [17] D. Sankoff. Minimal mutation trees of sequences. *SIAM Journal on Applied Mathematics*, 28:35–42, 1975.
- [18] C. Semple and M.A. Steel. *Phylogenetics*. Oxford University Press, 2003.
- [19] M.A. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification*, 9(1):91–116, 1992.