Convex Recolorings of Strings and Trees: Definitions, Hardness Results and Algorithms *

Shlomo Moran† Sagi Snir‡

December 9, 2007

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 colorings of trees arise in areas such as phylogenetics, linguistics, etc. e.g., a perfect phylogenetic tree is one in which the states of each character induce a convex coloring of the tree.

When a coloring of a tree is not convex, it is desirable to know "how far" it is from a convex one, and what are the convex colorings which are "closest" to it. In this paper we study a natural definition of this distance - the *recoloring distance*, which is the minimal number of color changes at the vertices needed to make the coloring convex. We show that finding this distance is NP-hard even for a colored string (a path), and for some other interesting variants of the problem. In the positive side, we present algorithms for computing the recoloring distance under some natural generalizations of this concept: the first generalization is the *uniform weighted* model, where each vertex has a weight which is the cost of changing its color. The other is the *non-uniform* model, in which the cost of coloring a vertex v by a color d is an arbitrary nonnegative number $cost(v, d)$. Our first algorithms find optimal convex recolorings of strings and bounded degree trees under the non-uniform model in time which, for any fixed number of colors, is linear in the input size. Next we improve these algorithm for the uniform model to run in time which is linear in the input size for a fixed number of *bad* colors, which are colors which violate convexity in some natural sense. Finally, we generalize the above result to hold for trees of unbounded degree.

*A preliminary version of some of the results in this paper appeared in [17].

†Computer Science dept., Technion, Haifa 32000, Israel. moran@cs.technion.ac.il. This reserach was supported by the Technion VPR-fund and by the Bernard Elkin Chair in Computer Science.

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

1 Introduction

A phylogenetic tree is a tree which represents the course of evolution for a given set of species. The leaves of the tree are labelled with the given species. Internal vertices correspond to hypothesized, extinct species. A *character* is a biological attribute shared among all the species under consideration, although every species may exhibit a different *character state*. Mathematically, if X is the set of species under consideration, a character on X is a function C from X into a set $\mathcal{C}$ of character states. A character on a set of species can be viewed as a *coloring* of the species, where each color represents one of the character's 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.

In graph theoretic terms, the lack of reverse and convergent transitions means that the character is *convex* on the tree: for each state of this character, all species (extant and extinct) possessing that state induce a single *block*, which is a maximal monochromatic 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 to which a lot of research has been dedicated throughout the years. The *Perfect Phylogeny* (PP) problem, whose complexity was extensively studied (e.g. [12, 14, 1, 15, 5, 20]), receives a set of characters on a set of species and seeks for a phylogenetic tree on these species, that is simultaneously convex on each of the characters. *Maximum parsimony* (MP) [10, 18] is a very popular tree reconstruction method that seeks for a tree which minimizes the parsimony score defined as the number of mutated edges summed over all characters (therefore, PP is a special case of MP). [11] introduce another criterion to estimate the distance of a phylogeny from convexity. They define the *phylogenetic number* as the maximum number of connected components a single state induces on the given phylogeny (obviously, phylogenetic number one corresponds to a perfect phylogeny). However, both the parsimony score and the phylogenetic number of a tree do not specify a distance to some *concrete* convex coloring of the given tree: there are colored trees with large phylogenetic numbers (and large parsimony scores) that can be transformed to convex coloring by changing the color of a single vertex, while other trees with smaller phylogenetic numbers can be transformed to convex colorings only by changing the colors of many vertices.

Convexity is a desired property in other areas of classification, beside phylogenetics. For instance, in [4, 3] a method called *TNoM* is used to classify genes, based on data from gene expression extracted from two types of tumor tissues. The method finds a separator on a binary vector, which minimizes the number of "1" in one side and "0" in the other, and thus defines a convex vector of minimum Hamming distance to the given binary vector. Algorithms which finds this distance for vectors with any number of letters, in order to handle more types of tumor tissues, are given by the optimal string recoloring algorithms in this paper. In [13], distance from convexity is used (although not explicitly) to show strong connection between strains of Tuberculosis and their human carriers.

In this work we define and study a natural distance from a colored tree to a convex one: the *recoloring distance*. In the simplest, unweighted model, this distance is the minimum number of color changes at the vertices needed to make the given coloring convex (for strings this reduces to Hamming distance from a closest convex coloring). This measure is naturally motivated by the scenario of introducing a new character to an existing phylogenetic tree: the new character should not affect the structure of the tree, and we wish to find the minimum number of state changes needed to make the new charcter convex. We note that this problem has a natural generalization to the "big convex recoloring" problem, where one is given a set of characters (colorings) and the goal is to construct a phylogenetic tree which minimizes the recoloring distance from a perfect phylogeny. A somewhat a restricted version of this "big convex recoloring" problem, where characters are restricted to two states only, is studied in [8]. In [2] a similar problem, which relaxes the notion of compatibility into similarity, is studied. For a given a set of binary characters, a tree that maximizes the similarity to each of the characters is sought. The problem is shown to be NP-hard and efficient approximation algorithms for it are presented.

The "recoloring distance" measure generalizes to a weighted model, where changing the color of vertex v costs a nonnegative weight $w(v)$. These weighted and unweighted models are *uniform*, in the sense that the cost of changing the color of a vertex is independent of the colors involved. The most general model we study is the *non-uniform* model, where the cost of coloring vertex v by a color d is an arbitrary nonnegative number $cost(v, d)$.

We show that finding the recoloring distance in the unweighted model is NP-hard even for a string (a tree with two leaves), and also for the case where character states are given only at the leaves (so that changes on extinct species are not counted); we also address a variant of the problem, in which a block-recoloring is considered as an atomic operation. This operation changes the color of all the vertices in a given input block. We show that finding the minimum number of block-recolorings needed to obtain convexity is NP-Hard as well.

On the positive side, we present few algorithms for minimal convex recoloring of strings and trees. The first algorithms solve the problem in the non-uniform model. The running time of these algorithms for bounded degree trees is exponential in the number of colors, but for each fixed number of colors is linear in the input size. Then we improve these algorithms for the uniform model, so that the running time is exponential only in the number of *bad* colors, which are colors that violate convexity (to be defined precisely). These algorithms are noted to be fixed parameter tractable algorithms ([6]) for bounded degree trees, where the parameter is taken to be the recoloring distance. Finally, we eliminate the dependence on the degree of the tree in both the non-uniform and the uniform versions of the algorithms.

The rest of the paper is organized as follows. In the next section we present the notations used and define the unweighted, weighted and non-uniform versions of the problem. In Section 3 we show our NP-Hardness results and in Section 4 we present the algorithms. We conclude and point out future research directions in Section 5.

2 Preliminaries

A colored tree is a pair (T, C) where $T = (V, E)$ is a tree with vertex set $V = \{v_1, \ldots, v_n\}$, and C is a *coloring* of T, i.e. - a function from V onto a set of colors $\mathcal{C}$. For a set $U \subseteq V$, $C|_U$ denotes the restriction of C to the vertices of U, and $C(U)$ denotes the set $\{C(u) : u \in U\}$. 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 $\sum_{d \in \mathcal{C}}(n_b(C, d) - 1)$. Thus a coloring C is convex iff the total number of violations of C is zero (in [9] the above sum, taken over all characters, is used as a measure of the distance of a given phylogenetic tree from perfect phylogeny).

The definition of convex coloring is extended to *partially colored* trees, in which the coloring C assigns colors to some subset of vertices $U \subseteq V$, which is denoted by $Domain(C)$. A partial coloring is said to be convex if it can be extended to a total convex coloring (see [19]). Convexity of partial and total coloring have simple characterization by the concept of *carriers*: For a subset U of V, $carrier(U)$ is the minimal subtree that contains U. for a colored tree (T, C) and a color $d \in C$, $carrier_T(C, d)$ (or $carrier(C, d)$ when T is clear) is the carrier of $C^{-1}(d)$. We say that C has the *disjointness property* if for each pair of colors $\{d, d'\}$ it holds that $carrier(C, d) \cap carrier(C, d') = \emptyset$. It is easy to see that a total or partial coloring C is convex iff it satisfies the disjointness property (in [7] convexity is actually defined by the disjointness property).

When some (total or partial) 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. Specifically, a recoloring C' of C overwrites a vertex v either by changing the color of v, or just by *uncoloring* v. We say that C' retains (overwrites) a set of verices U if it retains (overwrites resp.) every vertex in U. For a recoloring C' of an input coloring C, $\mathcal{X}_C(C')$ (or just $\mathcal{X}(C')$) is the set of the vertices overwritten by C', i.e.

$$\mathcal{X}_C(C') = \{v \in V : [v \in Domain(C)] \bigwedge [(v \notin Domain(C')\ ) \vee (C(v) \neq C'(v)\ )]\}.$$

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, and $cost_C(C^*) \leq cost_C(C')$ for any other convex coloring C' of C.

The above cost function naturally generalizes to the *weighted* version: the input is a triplet (T, C, w), where $w : V \to \mathbb{R}^+ \cup \{0\}$ is a weight function which assigns to each vertex v a nonnegative weight $w(v)$. For a set of vertices X, $w(X) = \sum_{v \in X} w(v)$. The cost of a convex recoloring C' of C is $cost_C(C') = w(\mathcal{X}(C'))$, and C' is an optimal convex recoloring if it minimizes this cost.
The above unweighted and weighted cost models are *uniform*, in the sense that the cost of a recoloring is determined by the set of overwritten vertices, regardless the specific colors involved. A yet further generalization allows *non-uniform* cost functions. This version, motivated by weighted

maximum parsimony [18], assumes that the cost of assigning color d to vertex v is given by an arbitrary nonnegative number $cost(v, d)$ (note that, formally, no initial coloring C is assumed in this cost model). In this model $cost(C')$ is defined only for a total recoloring C', and is given by the sum $\sum_{v \in V} cost(v, C'(v))$. The non-uniform cost model appears to be more subtle than the uniform ones. Unless otherwise stated, our results assume the uniform, 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 d, then by the optimality of C', there must be a vertex v such that $C(v) = d$ and the claim is proved. Assume for contradiction that C' uses at least two colors, and that for some color d used by C', there is no vertex v s.t. $C(v) = C'(v) = d$. Then there must be an edge (u, v) such that $C'(u) = d$ but $C'(v) = d' \neq d$. Note that, in the uniform cost model, each vertex v s.t. $C'(v) = d$ has already been overwritten and contributed its weight to the total cost. Therefore, the coloring C'' which is identical to C' except that all vertices colored d 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 a natural variant of the problem called minimal convex *block recoloring*, in which an atomic operation changes the color of a complete block, are NP-Hard as well.

3.1 Minimal Convex Recoloring of Strings is NP-Hard

A string $S = (v_1, \ldots, v_n)$ is a simple tree with $V = \{v_1, \ldots, v_n\}$ and $E = \{(v_i, v_{i+1}) | i = 1, \ldots, n-1\}$. 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 recoloring of strings 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 1: 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.

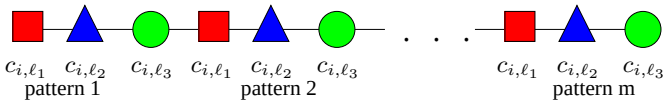

Figure 2: A clause segment. The literals are ℓ_1, ℓ_2 and ℓ_3, and the clause of size $3A$ consists of A repetitions of the corresponding triplet. Each block is a single vertex.

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, $F = D_1 \bigwedge ... \bigwedge D_m$, 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 1). 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* $d_1, \ldots, d_n$ and $2nm$ *literal colors* $\{c_{i,x_j}, c_{i,\neg x_j} | i = 1, \ldots, m; j = 1, \ldots n\}$.

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 2).

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 $c_{i,x_j}, i = 1, \ldots, m$, see Figure 3. Similarly, for each negated literal $\neg x_j$ we have a literal segment $S_{\neg x_J}$, which is similar to S_{x_j} except that the colors of the literal blocks are $c_{i,\neg x_j}, i = 1, \ldots, m$ (note that each of the literal segments S_{x_j} and $S_{\neg x_j}$ contain $m + 1$ d_j-blocks).

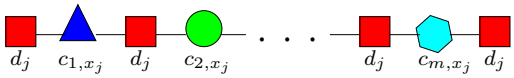

Figure 3: S_{x_j}, the segment of the literal x_j. $m+1$ d_j-blocks are interleaved by the m blocks c_{i,x_j}, $i = 1, \ldots, m$.

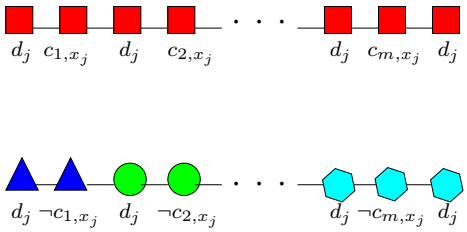

Figure 4: A recoloring of segments $S_{\neg x_j}$ and S_{x_j} corresponding to the assignment $f(x_j) = 1$. The segment S_{x_j} is overwritten by d_j, while in the segment $S_{\neg x_j}$, the d_j blocks are overwritten by the literal colors.

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. $\Longleftarrow$ 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 $i = 1, \ldots, m$ (see Figure 4). 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 overwriting $2A$ out of the $3A$ vertices in this block (since A vertices are originally colored by c_{i,x_j}). Thus the total cost of coloring all the m clause segments is $2mA$.

$\Longrightarrow$ 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 $\{S_{x_j}, S_{\neg x_J}\}$, $j \in \{1, \ldots, n\}$:*

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 $j = 1, \ldots, n$, $f(x_j) = 1$ 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 $f(x_j) = 1, j = 1, \ldots, n$. 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 $l_i = 0 (\text{mod } 3)$ then C^* overwrites exactly $\frac{2}{3} l_i$ vertices in the c_i-block; if $l_i = 1 (\text{mod } 3)$ then at least $\frac{2}{3}(l_i - 1)$ vertices are overwritten in that block, and if $l_i = 2 (\text{mod } 3)$

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 $j = 1\ldots, n$, 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 $c_{i,\neg x_j}$ or c_{i,x_j} for some j. By Claim 3.4.1 C^* retains all the $c_{i,\neg x_j}$-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 partially 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.

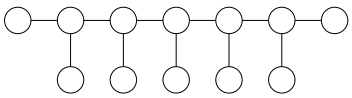

Figure 5: A caterpillar of length 5.

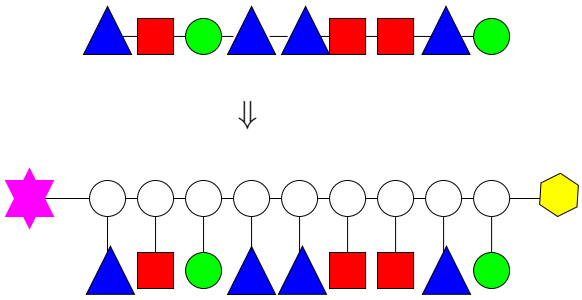

Figure 6: A reduction from a fully colored string to a leaf colored caterpillar. The two *external* leaves are colored with two new colors. All the *internal* leaves are colored with the same color as the corresponding vertex in the string.

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 $V(S) = \{v_1, \ldots, v_n\}$; then $V(S') = \{v_i^j : 1 \leq i \leq n, 1 \leq j \leq \ell\}$ and $E(S') = \{(v_i^{j-1}, v_i^j) : 1 \leq i \leq n, 1 < j \leq \ell\} \cup \{(v_{i-1}^{\ell}, v_i^1) : 1 < i \leq n\}$. $C'(v_i^j) = C(v_i), i = 1, \ldots, n, j = 1, \ldots, \ell$. 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 5). 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 6).

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,*

$$[OPT((S, C)) = k] \iff [n_c(k-1) < OPT((T, C_T)) \leq n_c k] .$$

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

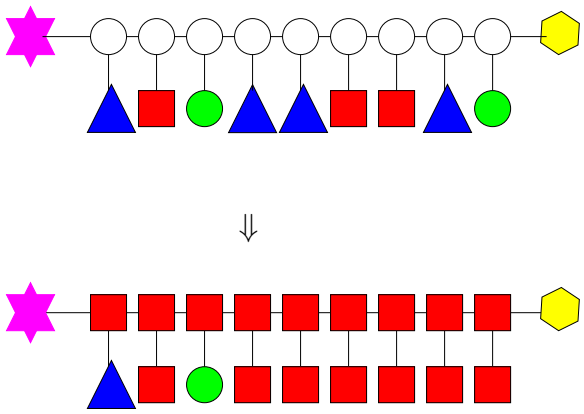

Figure 7: A convex recoloring of the caterpillar of Figure 6. Two *internal* leaves have colors different from their neighbors.

Let $(S', C') = dup_{n_c}(S, C)$. By Observation 3.9, (S', C') has a recoloring C^* with $cost_{C'}(C^*) = n_c k$. 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 $cost_{C_T}(C^*_T) = cost_{C'}(C^*) = n_c k$. 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 $n_c - 1$ (see Figure 7). Hence $cost_{C'}(C^*) < cost_{C_T}(C^*_T) + n_c$. Thus we have

$$n_c k \leq cost_{C'}(C^*) < cost_{C_T}(C^*_T) + n_c = OPT((T, C_T)) + n_c,$$

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-Recoloring

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 modelling 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 of

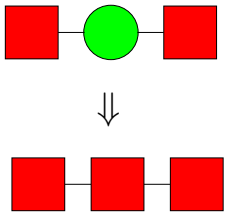

Figure 8: Removing the mutation at the left edge implies the removal of the one at the right

a given mutation can imply the elimination of other mutations, as depicted in Figure 8. Also, as in Observation 2.1 we can show that allowing block-recoloring by arbitrary colors (i.e., not only by colors of adjacent blocks) cannot reduce the minimum number of block-recoloring 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-recoloring needed to transform the input coloring to a convex one.

By Claim 3.1, convex recoloring of unweighted strings can be reduced to the problem of convex block recoloring of weighted strings, by collapsing each block B in the input string to a single vertex whose weight is the number of vertices in B. Hence, by Theorem 3.2, convex block recoloring of *weighted* strings is NP-Hard. In the rest of this section we show that the *unweighted* version of this problem 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 *Minimal unweighted 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: an informative segment, a 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 9).
- **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 10).

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 $m = 5n + k - 1$.

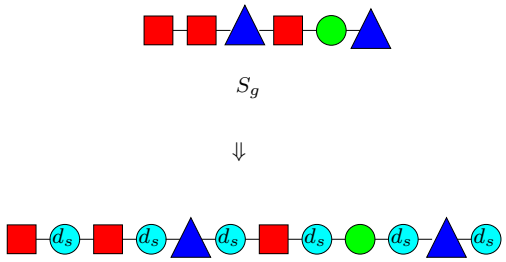

Figure 9: The input string (S, C) and the corresponding informative segment in (S_z, C_z).

$\Longrightarrow$ 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.

$\Longleftarrow$ 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. ∎

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

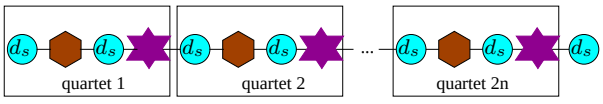

Figure 10: The counter-weight segment in S_s.

Proof. It is straightforward to show that any optimal convex coloring of the counter-weight segment must overwrite two vertices in each quartet (eg, the coloring which transforms it to a d_s-block), and such a 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 recoloring *and* block recoloring needed to transform a colored string to a convex one is NP-Hard.

4 Optimal Convex Recoloring Algorithms

In this section we present dynamic programming algorithms for optimal convex recoloring of totally colored strings or trees. The input is either a totally colored string (S, C) or a totally colored tree (T, C), which will be clear from the context. The algorithms are formulated so that they return the cost of an optimal convex recoloring, but are easily modified to return actual optimal convex recolorings, which will be either total or partial, as will be detailed.

The basic ingredient in all the algorithms is coloring with forbidden colors: A convex recoloring of the whole tree is constructed by extending convex recolorings of smaller subtrees, and in order to maintain convexity of the coloring, in each subtree certain colors cannot be used.

The computational costs of the algorithms depend either on n_c, the total number of colors used, or on n_c^*, the number of colors which violate convexity in the input tree, defined as follows: A color d is a *good color* for a totally colored tree (T, C) if (T, C) contains a unique d-block. Else d is a *bad color*. n_c^* denotes the number of bad colors in the input. In the sequel, a good (bad) color refers to a color that is good (bad) for some input coloring C, which will be obvious from the context.

We start with basic algorithms which are valid for the general non-uniform cost model, and their time complexity in bounded degree trees is $Poly(n)Exp(n_c)$. We then modify these algorithms to run in time $Poly(n)Exp(n_c^*)$ in the uniform weighted model. Finally, we remove the degree bound and modify the algorithms to run in $Poly(n)Exp(n_c^*)$ time for arbitrary trees.

4.1 Basic Algorithms for the Non-Uniform Cost Model

Our first algorithms find optimal convex recoloring of strings and trees in the non-uniform model, where for each vertex v and each color $d \in \mathcal{C}$, the cost of coloring v by d is an arbitrary nonnegative number $cost(v, d)$. The running times of both algorithms are governed by 2^{n_c}, the number of subsets of the set of colors $\mathcal{C}$. First we present an algorithm for colored strings, and then extend it to colored trees.

4.1.1 Non-Uniform Optimal Convex Recoloring of Strings

Throughout this section (S, C) is a fixed, n-long input colored string, where $S = (v_1, \ldots, v_n)$. The algorithm scans the string from left to right. After processing vertex v_i, it keeps for each subset of colors $\mathcal{D} \subseteq \mathcal{C}$, and for each color $d \notin \mathcal{D}$, the cost of the optimal coloring of the i leftmost vertices $v_1, \ldots, v_i$ which *does not* use colors from $\mathcal{D}$, and the rightmost vertex v_i is colored by d. We define this more formally now:

Definition 4.1 *Let $\mathcal{D} \subseteq \mathcal{C}$ be a set of colors and $i \in \{1, \ldots, n\}$. A coloring C' is a $(\mathcal{D}, i)$-coloring (of the string $S = (v_1, \ldots, v_n)$) if it is a convex coloring of $(v_1, \ldots, v_i)$, the i leftmost vertices of S, such that $C'(\{v_1, \ldots, v_i\}) \cap \mathcal{D} = \emptyset$. $opt(\mathcal{D}, i)$ is the cost of an optimal $(\mathcal{D}, i)$-recoloring of (S, C).*

[The reason for defining $\mathcal{D}$ as the set of colors which are not used by the coloring, rather then defining it as the set of *permitted* colors, which appears more natural, is that this definition fits better to the presentation of the main algorithm, in Section 4.2.2.]
It is easy to see that by the above definition, $opt(\emptyset, n)$ is the cost of an optimal convex recoloring of (S, C).

Definition 4.2 *For a set of colors $\mathcal{D}$, a color d, and $i \in \{1, \ldots, n\}$, a coloring C' is a $(\mathcal{D}, d, i)$-coloring if it is a $(\mathcal{D}, i)$-coloring and $C'(v_i) = d$. $opt(\mathcal{D}, d, i)$ is the cost of an optimal $(\mathcal{D}, d, i)$-coloring. $opt(\mathcal{D}, d, i) = \infty$ when no $(\mathcal{D}, d, i)$-coloring exists (eg when $d \in \mathcal{D}$).*

Observation 4.1 $opt(\mathcal{D}, i) = \min_{d \in \mathcal{C}} opt(\mathcal{D}, d, i)$.

For the recursive calculation of $opt(\mathcal{D}, d, i)$ we use the following function R, defined for a color set $\mathcal{D} \subseteq \mathcal{C}$, a color $d \in \mathcal{C}$ and $i \in \{1, \ldots, n\}$:

$$R(\mathcal{D}, d, i) = \min\{opt(\mathcal{D} \cup \{d\}, i), opt(\mathcal{D} \setminus \{d\}, d, i)\}$$

That is, $R(\mathcal{D}, d, i)$ is the minimal cost of a convex recoloring of the leftmost i vertices, which does not use colors from $\mathcal{D} \setminus \{d\}$, and may use the color d only as the color of the last (rightmost) block in $(v_1, \ldots, v_i)$. By convention, $opt(\mathcal{D}, d, 0) = 0$ for all $\mathcal{D} \subseteq \mathcal{C}$ and $d \notin \mathcal{D}$. Note that $R(\mathcal{D}, d, i) = R(\mathcal{D} \cup \{d\}, d, i) = R(\mathcal{D} \setminus \{d\}, d, i)$; we will usually use this function when $d \notin \mathcal{D}$.

Theorem 4.2 *For a color set $\mathcal{D}$, a color $d \notin \mathcal{D}$ and $i \in \{1, \ldots, n\}$:*

$$opt(\mathcal{D}, d, i) = cost(v_i, d) + R(\mathcal{D}, d, i - 1)$$

Proof. Let C' be an optimal $(\mathcal{D}, d, i)$-coloring. Then, since C' is convex and $C'(v_i) = d$, the restriction of C' to $(v_1, \ldots, v_{i-1})$ is either a $(\mathcal{D}, d, i-1)$-coloring or a $(\mathcal{D} \cup \{d\}, i-1)$-coloring. Hence the cost of this restriction is at least $R(\mathcal{D}, d, i-1)$. This proves that $opt(\mathcal{D}, d, i)$ is at least the righthand side of the equation. Conversely, let C' be a coloring of $(v_1, \ldots, v_{i-1})$ of cost $R(\mathcal{D}, d, i-1)$ which does not use colors from $\mathcal{D}$, and uses color d only if $C(v_i) = d$. Then by setting $C'(v_i)$ to d we get a $(\mathcal{D}, d, i)$-coloring whose cost is the righthand side of the equation. Therefore this cost is at least the cost of an optimal $(\mathcal{D}, d, i)$-coloring. ∎

Theorem 4.2 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 $d \notin \mathcal{D}$, $opt(\mathcal{D}, d, 0) \leftarrow 0$
2. for $i = 1$ to n

   for every $\mathcal{D} \subseteq \mathcal{C}$

   (a) for every $d \notin \mathcal{D}$, $opt(\mathcal{D}, d, i) \leftarrow cost(v_i, d) + R(\mathcal{D}, d, i-1)$

   (b) $opt(\mathcal{D}, i) \leftarrow \min_d opt(\mathcal{D}, d, i)$.
3. return $opt(\emptyset, n)$

Each of the n iterations of the algorithms requires $O(n_c \cdot 2^{n_c})$ time. So the running time of the above algorithm is $O\left(n \cdot n_c 2^{n_c}\right)$.

4.1.2 Non-uniform Optimal Convex Recoloring of Trees

We extend the algorithm of the previous section for optimal convex recoloring of trees. First, we root the tree at some vertex r. For each vertex $v \in V$, T_v is the subtree rooted at v. A convex recoloring of T_v denotes a convex recoloring of the colored subtree $(T_v, C|_{V(T_v)})$. We extend the definitions of the previous section to handle trees:

Definition 4.3 *Let $\mathcal{D} \subseteq \mathcal{C}$ be a set of colors and $v \in V$. Then a coloring C' is a $(\mathcal{D}, T_v)$-coloring if it is a recoloring of T_v s.t. $C'(V(T_v)) \cap \mathcal{D} = \emptyset$. $opt(\mathcal{D}, T_v)$ is the cost of an optimal $(\mathcal{D}, T_v)$-coloring.*

Again, a $(\mathcal{D}, T_v)$-coloring is a (convex) coloring on T_v that does *not* use any color of $\mathcal{D}$. Thus $opt(\emptyset, T_r)$ is the cost of an optimal coloring of $T = T_r$.

Definition 4.4 *For a set of colors $\mathcal{D} \subseteq \mathcal{C}$, a color d and $v \in V$, a coloring C' is a $(\mathcal{D}, d, T_v)$-coloring if it is a $(\mathcal{D}, T_v)$-coloring such that $C'(v) = d$. $opt(\mathcal{D}, d, T_v)$ is the cost of an optimal $(\mathcal{D}, d, T_v)$-coloring; in particular, if $d \in \mathcal{D}$ then $opt(\mathcal{D}, d, T_v) = \infty$.*

If v is a leaf and $d \notin \mathcal{D}$, then $opt(\mathcal{D}, d, T_v) = cost(v, d)$. For the recursive calculation of $opt(\mathcal{D}, d, T_v)$ at internal vertices we need the following generalization of the function R used for the string algorithm:

$$R(\mathcal{D}, d, T_v) = \min\{opt(\mathcal{D} \cup \{d\}, T_v), opt(\mathcal{D} \setminus \{d\}, d, T_v)\}$$

That is, $R(\mathcal{D}, d, T_v)$ is the minimal cost of a convex recoloring of T_v, which uses no colors from $\mathcal{D} \setminus \{d\}$ and does not include a d-block which is disjoint from the root v.

The calculation of $opt(\mathcal{D}, d, T_v)$ at an internal vertex with k children $v_1, \ldots, v_k$ uses the notion of *k-ordered partition* of a set S, which is a k-tuple $(S_1, \ldots, S_k)$, where each S_i is a (possibly empty)

subset of S, s.t. $S_i \cap S_j = \emptyset$ for $i \neq j$ and $\cup_{i=1}^{k} S_i = S$. The set of $k^{|S|}$ k-ordered partitions of a set S is denoted by $\mathcal{PART}_k(S)$.

Theorem 4.3 *Let v be an internal vertex with children $v_1, \ldots, v_k$. Then, for a color set $\mathcal{D}$ and a color $d \notin \mathcal{D}$:*

$$opt(\mathcal{D}, d, T_v) \quad = \quad cost(v, d) + \min_{(\mathcal{E}_1,\ldots,\mathcal{E}_k) \in \mathcal{PART}_k(\mathcal{C} \setminus (\mathcal{D} \cup \{d\}))} \sum_{i=1}^{k} R(\mathcal{C} \setminus \mathcal{E}_i, d, T_{v_i})$$

Proof.
$\geq$: Let C' be an optimal $(\mathcal{D}, d, T_v)$-coloring. Then $cost(C') = opt(\mathcal{D}, d, T_v)$. For $i = 1, \ldots, k$, let $\mathcal{E}'_i = C'(T_{v_i}) \setminus \{d\}$, that is: $\mathcal{E}'_i$ is the set of colors different from d which C' uses in coloring T_{v_i}. Since C' is convex and $C'(v) = d$, we must have that, for $i \neq j$, $\mathcal{E}'_i \cap \mathcal{E}'_j = \emptyset$. Since $\mathcal{E}'_i$ cannot contain a color from $\mathcal{D} \cup \{d\}$, we have that $\bigcup_{i=1}^{k} \mathcal{E}'_i \subseteq \mathcal{C} \setminus (\mathcal{D} \cup \{d\})$. If $\bigcup_{i=1}^{k} \mathcal{E}'_i$ is strictly included in $\mathcal{C} \setminus (\mathcal{D} \cup \{d\})$, then replace $\mathcal{E}'_1$ by a larger set which includes all the missing colors from $\mathcal{C} \setminus (\mathcal{D} \cup \{d\})$. With this modification, $(\mathcal{E}'_1, \ldots, \mathcal{E}'_k)$ is an ordered partition of $\mathcal{C} \setminus (\mathcal{D} \cup \{d\})$, and for each i, $C'|_{T_{v_i}}$ is a convex recoloring of T_{v_i} which uses only colors from $\mathcal{E}'_i \cup \{d\}$, and if it uses d then $C'(v_i) = d$. Therefore, for every $1 \leq i \leq k$, $cost(C'|_{T_{v_i}}) \geq R(\mathcal{C} \setminus \mathcal{E}_i, d, T_{v_i})$. Hence $cost(C')$ is at least the righthand side of the equation.
$\leq$: Let $(\mathcal{E}_1, \ldots, \mathcal{E}_k)$ be an ordered partition which minimizes the righthand side of the equation, and let C'_i be the coloring of T_{v_i} attaining the cost $R(\mathcal{C} \setminus \mathcal{E}_i, d, T_{v_i})$ $(i = 1, \ldots, k)$. Let C' be the coloring of T_v defined by $C'|_{T_{v_i}} = C'_i$ and $C'(v) = d$. Then $cost(C')$ equals the righthand side of the equation. Also, by the construction, C' is a convex recoloring of T_v which does not use colors from $\mathcal{D}$ and $C'(v) = d$. Hence $cost(C')$ is at least $opt(\mathcal{D}, d, T_v)$. ∎

Theorem 4.3 above leads to a straightforward dynamic programming algorithm. In order to compute $opt(\mathcal{D}, d, T_v)$ for each $\mathcal{D} \subseteq \mathcal{C}$ and $d \notin \mathcal{D}$, we only need the corresponding values at v's children. This can be achieved by a post order visit of the vertices, starting at r. To evaluate the complexity of the algorithm, we first note that each subset of colors $\mathcal{D}$ and a k-ordered partition $(\mathcal{E}_1, \ldots, \mathcal{E}_k)$ of $\mathcal{C} \setminus (\mathcal{D} \cup \{d\})$ corresponds to the $(k+1)$-ordered partition $(\mathcal{D}, \mathcal{E}_1, \ldots, \mathcal{E}_k)$ of $\mathcal{C} \setminus \{d\}$. For each such ordered partition, $O(k)$ computation step are needed. As there are n_c colors, the total time for the computation at vertex v with k children is $O(kn_c(k+1)^{n_c-1})$. Since $k \leq \Delta - 1$, the time complexity of the algorithm for trees with bounded degree Δ is $O(n \cdot n_c \cdot \Delta^{n_c})$.

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 each vertex $v \neq r$ a cost defined by

$$cost(v) = \sum_{v' \in T_v} cost(v', d_2) + \sum_{v' \notin T_v} cost(v', d_1))$$

This can be done by one post order traversal of the tree. Then, select the vertex 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.

4.2 Enhanced Algorithms for the Uniform Cost Model

The running times of the algorithms in Section 4.1 do not improve even when the input coloring is convex. However, for the uniform cost model, we can modify these algorithms so that their running time on convex or nearly convex input (string or tree) is substantially smaller. The new algorithms, instead of returning a total coloring, return a convex partial coloring, in which some of the new colors assigned to the vertices are unspecified. For the presentation of the algorithms we need the notion of convex cover which we define next.

A set of vertices X is a *convex cover* (or just a cover) for a colored tree (T, C) if the (partial) coloring $C_X = C|_{[V \setminus X]}$ is convex (i.e., C can be transformed to a convex coloring by overwriting the vertices in X). Thus, if C' is a convex recoloring of (T, C), then $\mathcal{X}_C(C')$, the set of vertices overwritten by C', is a cover for (T, C). Moreover, deciding whether a subset $X \subseteq V$ is a cover for (T, C), and constructing a total convex recoloring C' of C such that $\mathcal{X}(C') \subseteq X$ in case it is, can be done in $O(n \cdot n_c)$ time. Also, in the uniform cost model, the cost of a recoloring C' is $w(\mathcal{X}(C'))$. Therefore, in this model, finding an optimal convex total recoloring of C is polynomially equivalent to finding an optimal cover X, or equivalently a partial convex recoloring C' of C so that $w(\mathcal{X}(C')) = w(X)$ is minimized.

4.2.1 Optimal String Recoloring via Relaxed Convex Recoloring

The enhanced algorithm for the string, makes use of the fact that partially colored strings can be characterized by the following property of "local convexity":

Definition 4.5 *A color d is* locally convex *for a partially colored tree (T, C) iff $C(carrier(C, d)) = \{d\}$, that is $carrier(C, d)$ does not contain a vertex of color different from d.*

Observation 4.4 *A partially colored string (S, C) is convex iff it is locally convex for each color $d \in \mathcal{C}$.*

Note that Observation 4.4 does not hold for partially colored trees, since every leaf-colored tree is locally convex for each of its colors.

Given a colored string (S, C) and a color d, (S, C) is a *d-relaxed convex coloring* if it can be completed to total coloring such that for every color $d' \neq d$ there is a unique d'-block.

Observation 4.5 *C is a d-relaxed convex coloring of a string S if and only if each color $d' \neq d$ is locally convex for (S, C).*

Given a colored string (S, C), we transform C to a coloring $\hat{C}$ as follows:
For every vertex $v \in V(S)$:

$$\hat{C}(v) = \begin{cases} \hat{d} & \text{if } C(v) \text{ is a good color} \\ C(v) & \text{otherwsise.} \end{cases}$$

where $\hat{d}$ is a new color. Figure 11 illustrates such a transformation.

A set of vertices $X \subseteq V$ is a *d-relaxed cover* of (S, C) if the partial coloring $C|_{V \setminus X}$, denoted C_X, is a d-relaxed convex coloring of (S, C).

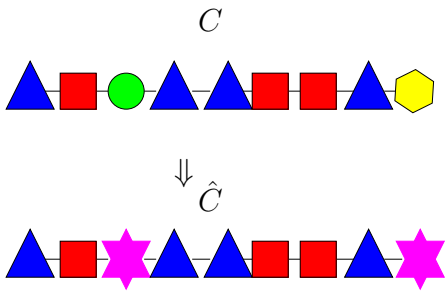

Figure 11: All good colors are replaced by a the new color $\hat{d}$, represented by ★.

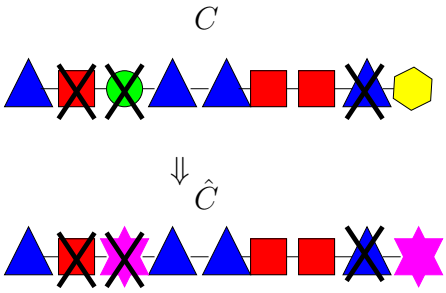

Figure 12: A cover of C implies a relaxed cover of $\hat{C}$ (color $\hat{d}$ is represented by ★).

Theorem 4.6 *Let (S, C) and $\hat{C}$ be as above. Then $X \subseteq V$ is a cover for (S, C) if and only if X is a $\hat{d}$-relaxed cover for $(S, \hat{C})$.*

Proof. Assume that X is a cover for (S, C). Then clearly all colors are locally convex for C_X, which implies that every color $d' \neq \hat{d}$ is locally convex for $\hat{C}_X$. Hence, by Observation 4.5, $\hat{C}_X$ is a $\hat{d}$-relaxed convex cover. The converse is also true: If each color $d' \neq \hat{d}$ is locally convex for $\hat{C}_X$, then each bad color (for C) is locally convex for $\hat{C}_X$, and hence also for C_X. Each good color for C is trivially locally convex for C_X. Thus by observation 4.4, C_X is convex. The theorem follows. ■

Figures 12, 13 depict Theorem 4.6 above.

Theorem 4.6 implies that an optimal convex cover (and hence an optimal convex recoloring) of (S, C) can be obtained as follows: transform C to $\hat{C}$, and then compute an optimal $\hat{d}$-relaxed convex recoloring, C', for $(S, \hat{C})$. The $\hat{d}$-relaxed cover defined by C' is an optimal cover of (S, C). An optimal convex recoloring of $(S, \hat{C})$ can be obtained by replacing step 2(a)of the non-uniform string recoloring algorithm of Section 4.1.1 by:

$$opt(\mathcal{D}, d, i) \leftarrow w(v)\delta_{C(v_i),d} + \begin{cases} opt(\mathcal{D}, i-1) & \text{if } d = \hat{d} \\ R(\mathcal{D}, d, i-1) & \text{otherwise.} \end{cases}$$

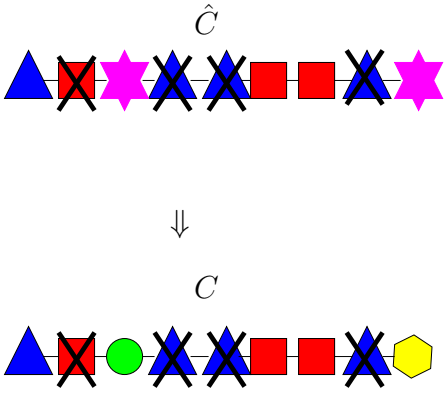

Figure 13: A relaxed cover of $\hat{C}$ implies a cover of C

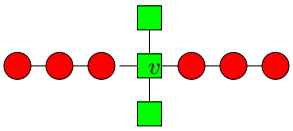

Figure 14: $\{v\}$ is not a cover, but as ■ is a good color, $\{v\}$ is a relaxed cover.

where R is defined in Section 4.1.1, and where $\delta_{d,d'}$ is the complement of Kronecker delta:

$$\delta_{d,d'} = \begin{cases} 1 & \text{if } d \neq d' \\ 0 & \text{otherwise} \end{cases}$$

The improved algorithm has running time of $O\left(n_c^* n 2^{n_c^*}\right)$. In particular, for each fixed value of n_c^* the running time is polynomial in the input size.

4.2.2 Extension for Trees

The technique of getting convex recoloring by treating all good colors as a special color $\hat{d}$ and then finding a $\hat{d}$-relaxed cover does not apply to trees, as can be seen in Figure 14: In this example there is a unique good color d, thus $d = \hat{d}$ and $C = \hat{C}$; $\{v\}$ is a d-relaxed cover for $(T, \hat{C})$, but it is not a cover for (T, C).

Let $(T = (V, E), C)$ be a (totally) colored tree, and let $\mathcal{C}^*$ be the set of bad colors. For a vertex $v \in V$, let $\mathcal{C}_v^* = \mathcal{C}^* \cup \{C(v)\}$ (note that if $C(v) \in \mathcal{C}^*$ then $\mathcal{C}_v^* = \mathcal{C}^*$). Assume that the children of v are $v_1, \ldots, v_k$. The crucial observation for our improved algorithm for convex recoloring of trees is that only colors from $\mathcal{C}_v^*$ may appear in more than one subtree T_{v_i} of T_v. This observation enables us to modify the recursive calculation of the algorithm of Section 4.1.2 so that instead of computing $opt(\mathcal{D}, d, T_v)$ for all subsets $\mathcal{D}$ of $\mathcal{C}$ and each $d \notin \mathcal{D}$, it computes similar values only for subsets $\mathcal{D} \subseteq \mathcal{C}_v^*$ and $d \in \mathcal{C}_v^* \setminus \mathcal{D}$, and thus to reduce the exponential factor in the complexity bound from 2^{n_c} to $2^{n_c^*}$.

To enable the bookkeeping needed for the algorithm, it considers only optimal *partial* recolorings of (T, C), which use good colors in a very restricted way: no vertex is overwritten by a good color (ie vertices are either retained, or uncolored, or overwritten by bad colors), and good colors are either retained or overwritten (by bad colors), but are never uncolored. The formal definition is given below.

Definition 4.6 *A partial convex recoloring C' of the input coloring C is* conservative *if it satisfies the following:*

1. *If $C'(v) \neq C(v)$ then $C'(v) \in \mathcal{C}^*$ (a color can be changed only to a bad color).*
2. *If $C(v) \notin \mathcal{C}^*$ then $v \in Domain(C')$ and $C'(v) \in \{C(v)\} \cup \mathcal{C}^*$ (a good color is either retained or overwritten by a bad color, but* not *uncolored).*
3. *For every $d \in \mathcal{C}$, $C'^{-1}(d)$ is connected (if a vertex is left uncolored then it does not belong to any carrier of C').*

The fact that a conservative recoloring of minimum possible cost is an optimal convex recoloring follows from the following lemma, which seems to be of independent interest:

Lemma 4.7 *Let X be a convex cover of a colored tree (T, C). Then there is a convex total recoloring $\hat{C}$ of (T, C) so that $\mathcal{X}(\hat{C}) \subseteq X$ and for each vertex v for which $C(v) \notin \mathcal{C}^*$, $\hat{C}(v) = C(v)$ or $\hat{C}(v) \in \mathcal{C}^*$ (that is, $\hat{C}$ does not overwrite a good color by another good color). In particular, there is an optimal total recoloring $\hat{C}$ of (T, C) which never overwrites a good color by another good color.*

Proof. The proof is by induction on $|X|$. If $|X| = 0$ (i.e. C is convex) then let $\hat{C} = C$. Assume correctness for $k \geq 0$, and let $|X| = k + 1$. If X contains a convex cover X' of cardinality $\leq k$ then by induction there is a convex recoloring $\hat{C}$ which does not overwrite a good color by another good color and $\mathcal{X}(\hat{C}) \subseteq X' \subset X$, and the lemma holds. So assume that no proper subset X' of X is a convex cover (i.e., X is a minimal convex cover). Let $C_X = C|_{V \setminus X}$ be the partial (convex) coloring defined by X. If $C(X) \subseteq \mathcal{C}^*$ then the lemma holds for each convex recoloring $\hat{C}$ with $\mathcal{X}(\hat{C}) = X$, so assume that $C(u) \notin \mathcal{C}^*$ for some $u \in X$. This implies, by the minimality of X, that there is a vertex $v \in X$ such that $C(v) = d$ for some good color $d \notin \mathcal{C}^*$, and v is a leaf in the unique d-block of C. Let $X' = X \setminus \{v\}$. By the minimality of X, X' is not a convex cover. Let $C_{X'} = C|_{V \setminus X'}$ be the (non-convex) partial coloring defined by X'.

By assumption $C_{X'}$ is not convex, and the only color whose carrier under $C_{X'}$ is different from its carrier under C_X is d. Hence, there is a color $d' \neq d$ s.t. $carrier(C_{X'}, d)$ (which is $carrier(C_X^{-1}(d) \cup \{v\})$) intersects with $carrier(C_X, d')$. Since carriers of good colors do not intersect, each such color d' is a bad color. Hence either $v \in carrier(C_X, d')$ for some $d' \in \mathcal{C}^*$, or there is a vertex u which is the first vertex on the path from v to $carrier(C_X, d)$ which belongs to $carrier(C_X, d')$ for some $d' \in \mathcal{C}^*$ (see Figure 15; note that all vertices on the path from v to u must be in X).

Let C' be the total coloring which is identical to C except that $C'(v) = d'$ (see Figure 16). Then C' and C use the same colors, and every color which is good for C is good also for C' (this is

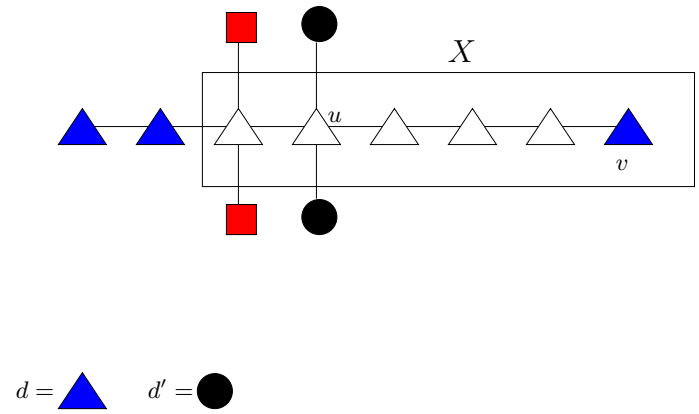

Figure 15: u is the first vertex on the path from v to $carrier(C_X, d)$, which belongs to carrier of another color.

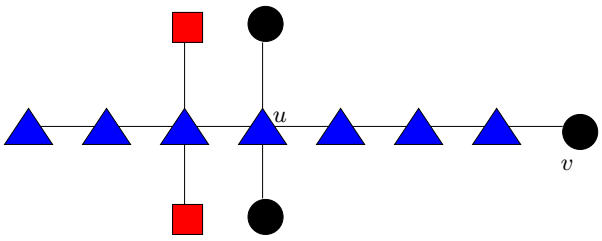

Figure 16: C' is obtained from C by changing the color of v from ▲ to ●.

trivial for good colors other than d, and also for d since v is a leaf in the unique d-block of C). Let $C'_{X'} = C'|_{V \setminus X'}$. Then for any color $d'' \neq d'$, $carrier(C'_{X'}, d'') = carrier(C_X, d'')$, and by the way d' was selected, also $carrier(C'_{X'}, d') \cap carrier(C'_{X'}, d'') = \emptyset$. Hence all color carriers in $C'|_{X'}$ are disjoint, meaning that X' is a convex cover for (T, C') with $|X'| = k$. By applying the induction hypothesis on C' and X', there is a convex recoloring $\hat{C}$ of C' so that $\mathcal{X}_{C'}(\hat{C}) \subseteq X'$ and no good color (of C', and hence also of C) is overwritten by another good color. Consider now $\hat{C}$ as a recoloring of C. Then $\hat{C}$ still satisfies the above, and since $\mathcal{X}_{C'}(C) \subseteq X'$, we have that $\mathcal{X}_C(\hat{C}) \subseteq X$, and the lemma is proved. ■

Let $\hat{C}$ be a convex total recoloring satisfying Lemma 4.7. Then it can be easily verified that the partial coloring obtained from $\hat{C}$ by *uncoloring* all the vertices v for which $\hat{C}(v) \neq C(v)$ and $\hat{C}(v) \notin \mathcal{C}^*$, is a conservative recoloring. Hence a conservative recoloring of minimum possible cost is an optimal convex recoloring.

For our algorithm we need variants of the functions *opt* and R, adapted for conservative recolorings, which we define next. A coloring C' is a $(\mathcal{D}, T_v)$-conservative recoloring if it is a conservative recoloring of T_v which does not use colors from $\mathcal{D}$. If in addition $C'(v) = d$, then C' is a $(\mathcal{D}, d, T_v)$-conservative recoloring; a $(\mathcal{D}, T_v)$-conservative recoloring in which v is uncolored is a

$(\mathcal{D}, *, T_v)$-conservative recoloring. Note that for certain combinations of $\mathcal{D} \subseteq \mathcal{C}, f \in (\mathcal{C} \setminus \mathcal{D}) \cup \{*\}$, and $v \in V$, no $(\mathcal{D}, f, T_v)$-conservative recoloring exists (eg, when $C(v)$ and f are two distinct good colors).

For $f \in \mathcal{C} \cup \{*\}$, a set of colors $\mathcal{D} \subseteq \mathcal{C}$ and $v \in V$, $\widehat{opt}(\mathcal{D}, f, T_v)$ is the cost of an optimal $(\mathcal{D}, f, T_v)$-conservative recoloring ($\widehat{opt}(\mathcal{D}, f, T_v) = \infty$ if no $(\mathcal{D}, f, T_v)$-conservative recoloring exists). $\widehat{opt}(\mathcal{D}, T_v)$, the optimal cost of a conservative recoloring of T_v which does not use colors from $\mathcal{D}$, is given by $\min_f \widehat{opt}(\mathcal{D}, f, T_v)$. By Lemma 4.7, the cost of an optimal recoloring of a colored tree (T, C) is given by $\widehat{opt}(T_r, \emptyset)$, where r is the root of T. The recursive computation of this value uses the function $\widehat{R}$, given by

$$\widehat{R}(\mathcal{D}, d, T_v) = \min\{\widehat{opt}(\mathcal{D} \cup \{d\}, T_v), \widehat{opt}(\mathcal{D} \setminus \{d\}, d, T_v)\}$$

Recall that $\mathcal{C}_v^* = \mathcal{C}^* \cup \{C(v)\}$. Rather than computing the functions $\widehat{opt}$ (and $\widehat{R}$) at each vertex v for all subsets $\mathcal{D}$ of $\mathcal{C}$, our algorithm computes $\widehat{opt}(\mathcal{D}, f, T_v)$ at a vertex v only for subsets of $\mathcal{C}_v^*$. The correctness and complexity of the algorithm follows from following two lemmas.

Lemma 4.8 *For a vertex v with children $v_1, \ldots, v_k$, a set of colors $\mathcal{D} \subseteq \mathcal{C}_v^*$, and a color $d \in \mathcal{C}_v^*$:*

1. *If $d \in \mathcal{D}$ then $\widehat{opt}(\mathcal{D}, d, T_v) = \infty$. If $d \in \mathcal{C}_v^* \setminus \mathcal{D}$ then:*

$$\widehat{opt}(\mathcal{D}, d, T_v) \quad = \quad w(v)\delta_{C(v),d} + \min_{(\mathcal{E}_1,\ldots,\mathcal{E}_k) \in \mathcal{PART}_k(\mathcal{C}_v^* \setminus (\mathcal{D} \cup \{d\}))} \sum_{i=1}^{k} \widehat{R}(\mathcal{C}_v^* \setminus \mathcal{E}_i, d, T_{v_i})$$

2. *If $C(v) \notin \mathcal{C}^*$ then $\widehat{opt}(\mathcal{D}, *, T_v) = \infty$. Else (ie $C(v) \in \mathcal{C}^*$ and $\mathcal{C}_v^* = \mathcal{C}^*$):*

$$\widehat{opt}(\mathcal{D}, *, T_v) \quad = \quad w(v) + \min_{(\mathcal{E}_1,\ldots,\mathcal{E}_k) \in \mathcal{PART}_k(\mathcal{C}_v^* \setminus \mathcal{D})} \sum_{i=1}^{k} \widehat{opt}(\mathcal{C}_v^* \setminus \mathcal{E}_i, T_{v_i})$$

Proof.

1. $\geq$: If $d \in \mathcal{D}$ then there is no $(\mathcal{D}, d, T_v)$-conservative recoloring. Otherwise the proof goes along the same lines of the proof of Theorem 4.3, only that this time we consider only colors from $\mathcal{C}_v^*$. Let C' be an optimal $(\mathcal{D}, d, T_v)$-conservative recoloring. By the same arguments of the proof of Theorem 4.3, C' induces an ordered partition $(\mathcal{E}'_1, \ldots, \mathcal{E}'_k)$ on $\mathcal{C}_v^* \setminus (\mathcal{D} \cup \{d\})$, such that, for $i = 1, \ldots, k$, $C'|_{T_{v_i}}$ overwrites all the colors in $\mathcal{C}_v^* \setminus \mathcal{E}'_i$. Now, since $C'(v) = d$, and C' is convex, for each i either $C'(v_i) = d$ or $d \notin C'(T_{v_i})$. Also, $w(v)$ is added to the cost iff $C(v) \neq d$. Hence, the cost of C' is at least $w(v)\delta_{C(v),d} + \sum_{i=1}^{k} \widehat{R}(\mathcal{C}_v^* \setminus \mathcal{E}'_i, d, T_{v_i})$, which is at least as large as the minimum of this sum over all ordered partitions in $\mathcal{PART}_k(\mathcal{C}_v^* \setminus (\mathcal{D} \cup \{d\}))$.
   $\leq$: Let $(\mathcal{E}_1, \ldots, \mathcal{E}_k)$ be an ordered partition of $\mathcal{C}_v^* \setminus (\mathcal{D} \cup \{d\})$ which minimizes the sum at the righthand side of the equation, and for $i = 1, \ldots, k$ let C'_i be the corresponding conservative recoloring of T_{v_i}, with $cost(C'_i) = \widehat{R}(\mathcal{C}_v^* \setminus \mathcal{E}_i, d, T_{v_i})$. Then, since the colorings C'_i are conservative, for each color $d' \notin \mathcal{C}_v^*$ there is at most one i s.t. $d' \in C'(T_{v_i})$. Hence, for $i \neq j$, $C'_i(T_{v_i}) \cap C'_j(T_{v_j}) \subseteq \{d\}$, and d is used by C'_i only if $C'_i(v_i) = d$. Let C' be the coloring

which equals C'_i on T_{v_i} $(i = 1, \ldots, k)$, and $C'(v) = d$. Then C' is a $(\mathcal{D}, d, T_v)$-conservative recoloring. Hence the cost of an optimal conservative recoloring is at most the cost of C', which is given in the righthand side.

2. If $C(v) \notin \mathcal{C}^*$ then there is no $(\mathcal{D}, *, T_v)$-conservative coloring. The proof for the case that $C(v) \in \mathcal{C}^*$ is similar to that of the previous item but simpler, and is omitted.

■

Lemma 4.8 implies a dynamic programming algorithm similar to the one presented in Section 4.1.2. The algorithm computes for each vertex v, for each subset of colors $\mathcal{D} \subseteq \mathcal{C}^*_v$ and for each $f \in (\mathcal{C}^*_v \setminus \mathcal{D}) \cup \{*\}$, the values of $\widehat{opt}(\mathcal{D}, d, T_v)$. when v is a leaf, this value for each $\mathcal{D} \subseteq \mathcal{C}^*_v$ and each $d \in \mathcal{D}$ is given by $\widehat{opt}(\mathcal{D}, d, T_v) = w(v)\delta_{C(v),d}$, and the value of $\widehat{opt}(\mathcal{D}, *, T_v)$ when $C(v) \in \mathcal{C}^*$ is $w(v)$. So it remains to show that these values can be computed at internal vertices, assuming they were previously computed at their children.

For an internal vertex v with children $v_1, \ldots, v_k$, the algorithm uses Lemma 4.8(1) to compute the values $\widehat{opt}(\mathcal{D}, d, T_v)$ for each $\mathcal{D} \subseteq \mathcal{C}^*_v$ and for each $d \in \mathcal{C}^*_v \setminus \mathcal{D}$. If $C(v) \in \mathcal{C}^*$, then Lemma 4.8(2) is used to compute the value of $\widehat{opt}(\mathcal{D}, *, T_v)$. There is however a subtle point in the realization of this algorithm, which stems from the fact that the sets $\mathcal{C}^*_v$ which define the values computed at each vertex v may vary from vertex to vertex. The following claim guarantees that all the values needed for the calculations at an internal vertex v are calculated by its children $v_1, \ldots, v_k$.

Lemma 4.9 *Let v be an internal vertex with children $v_1, \ldots, v_k$, and assume that v is visited by the algorithm after its children. Then for each subset of colors $\mathcal{D} \subseteq \mathcal{C}^*_v$ and each $f \in \mathcal{C}^*_v \cup \{*\}$, all the values required for computing $\widehat{opt}(\mathcal{D}, f, T_v)$ by Lemma 4.8 (1) and (2) are computed by $v_1, \ldots, v_k$.*

Proof. By our assumption, for each $i = 1, \ldots, k$, for each $\mathcal{D} \subseteq \mathcal{C}^*_{v_i}$ and for each $f \in \mathcal{C}^*_{v_i} \cup \{*\}$, the value of $\widehat{opt}(\mathcal{D}, f, T_{v_i})$ is computed by v_i. We have to prove that these values suffice to compute the formulas at (1) and (2) of Lemma 4.8.

Consider first the formula at (1). To compute the function $\widehat{R}(\mathcal{C}^*_v \setminus \mathcal{E}_i, d, T_{v_i})$, we need to compute $\widehat{opt}(\mathcal{C}^*_v \setminus \mathcal{E}_i, T_{v_i})$ and $\widehat{opt}(\mathcal{C}^*_v \setminus (\mathcal{E}_i \cup \{d\}), d, T_{v_i})$. Since d must be a member of $\mathcal{C}^*_v$, and $\mathcal{E}_i$ a subset of $\mathcal{C}^*_v \setminus \{d\}$, these values can be computed if we can compute $\widehat{opt}(\mathcal{D}', f, T_{v_i})$ for all $\mathcal{D}' \subseteq \mathcal{C}^*_v$ and $f \in \mathcal{C}^*_v \cup \{*\}$.

By our assumption, the values of $\widehat{opt}(\mathcal{D}', f, T_{v_i})$ are computed at v_i whenever $\mathcal{D}' \subseteq \mathcal{C}^*_{v_i}$ and $f \in \mathcal{C}^*_{v_i} \cup \{*\}$, so we only need to consider the cases where $\mathcal{D}' \not\subseteq \mathcal{C}^*_{v_i}$ or $f \notin \mathcal{C}^*_{v_i} \cup \{*\}$. If $f \notin \mathcal{C}^*_{v_i} \cup \{*\}$ then there is no $(\mathcal{D}', f, T_{v_i})$-conservative coloring, and hence $\widehat{opt}(\mathcal{D}', f, T_{v_i}) = \infty$. Thus we are left with the case that $\mathcal{D}' \not\subseteq \mathcal{C}^*_{v_i}$ and $f \in \mathcal{C}^*_{v_i} \cup \{*\}$.

Since $\mathcal{C}^*_v \setminus \mathcal{C}^*_{v_i} \subseteq \{C(v)\}$ and $\mathcal{D}' \subseteq \mathcal{C}^*_v$, in this case we must have that $\mathcal{D}' \setminus \mathcal{C}^*_{v_i} = \{C(v)\}$. That is: $C(v)$ is a good color and $C(v) \notin C(T_{v_i})$. Hence, in this case we have that every $(\mathcal{D}', d, T_{v_i})$-conservative recoloring of T_{v_i} is also a $(\mathcal{D}' \setminus \{C(v)\}, d, T_{v_i})$-conservative recoloring of T_{v_i}, and vice versa. Therefore, $\widehat{opt}(\mathcal{D}', d, T_{v_i}) = \widehat{opt}(\mathcal{D}' \setminus \{C(v)\}, d, T_{v_i})$, and since $\mathcal{D}' \setminus \{C(v)\} \subseteq \mathcal{C}^*_{v_i}$, the value of $\widehat{opt}(\mathcal{D}' \setminus \{C(v)\}, d, T_{v_i})$ is computed at v_i.

Consider now the formula at (2) of Lemma 4.8. The values needed at v here are $\widehat{opt}(\mathcal{D}_i, T_{v_i})$ for all $\mathcal{D}_i \subseteq \mathcal{C}^*_v$. Since in this case $C(v) \in \mathcal{C}^*$, we have that $\mathcal{C}^*_v = \mathcal{C}^* \subseteq \mathcal{C}^*_{v_i}$, and hence these values are computed at v_i's as well. ∎

Combining the results so far, we have

Theorem 4.10 *Optimal convex recoloring of totally colored trees with n vertices can be computed is $O(n \cdot n_c^* \Delta^{n_c^*+2})$ time, where n_c^* is the number of bad colors and Δ is the maximum degree of vertices in T.*

Proof. The correctness of the algorithm follows from Lemma 4.8. The complexity analysis is similar to the one after Theorem 4.3: By Lemma 4.9, the computation at each vertex v with k children can be done by using the formulas of Lemma 4.8, in time which is proportional to $k < \Delta$ for each $k+1$-ordered partition of $\mathcal{C}^*_v$ and color d. As $|\mathcal{C}^*_v| \leq n_c^* + 1$, the number of ordered partitions of $\mathcal{C}^*_v$, is at most $\Delta^{n_c^*+1}$. The theorem follows. ∎

4.3 Fixed Parameter Tractable Recoloring Algorithms

A fixed parameter tractable algorithm for the unweighted convex recoloring problem is one which computes the optimal solution for an input of size n in time which is bounded by $poly(n)f(k)$, where f is an arbitrary function and k is the value of the optimal solution, namely the minimum number of overwrites needed to make the coloring convex. This is a *fixed parameter tractable* solution to the problem, where the parameter is the value of the optimal solution (see [6]). As n_c^*, the number of bad colors, provides a lower bound on the number of overwrites, (effectively, the number of overwrites is at least $\frac{n_c^*}{2}$), the algorithm of the previous section is a fixed parameter tractable algorithm for each class of trees of bounded degree. In this section we remove the degree bound from this result. For this, we show below a modification of the algorithm that replaces the need to inspect ordered-partitions by inspecting *unordered* partitions of sets of colors. The running time is improved to $Poly(n)Bell(n_c^*)$, where $Bell(n)$ is the number of (unordered) partitions of n elements to any number of nonempty subsets[1]. The algorithm, which is based on minumum weight perfect matching algorithms, is presented for the calculation of $\widehat{opt}(\mathcal{D}, d, T_v)$, but it can easily be adapted for the calculation of $\widehat{opt}(\mathcal{D}, *, T_v)$.

Let v be an internal vertex with children $v_1, v_2, \ldots, v_\Delta$. Let $\mathcal{D} \subseteq \mathcal{C}^*_v$ and let $d \in \mathcal{C}^*_v \setminus \mathcal{D}$. Rather than calculating $\widehat{opt}(\mathcal{D}, d, T_v)$ by considering all the Δ-ordered partitions of $\mathcal{C}^*_v \setminus (\mathcal{D} \cup \{d\})$, we consider only *unordered non-empty* partitions of $\mathcal{C}^*_v \setminus (\mathcal{D} \cup \{d\})$ to at most Δ subsets. For each such partition $\{\mathcal{E}_1 \ldots \mathcal{E}_\ell\}$ we construct a complete weighted bipartite graph (V_1, V_2, E, w) as follows:

1. $V_1 = \{v_i : 1 \leq i \leq \Delta\}$ contains a vertex for each child of v,
2. $V_2 = \{\mathcal{E}_j : 1 \leq j \leq \ell\} \cup \{\phi_j : \ell + 1 \leq j \leq \Delta\}$ contains a vertex for each of the ℓ nonempty subsets $\mathcal{E}_i$, and additional $\Delta - \ell$ vertices that represent "copies" of the empty set.

[1] $Bell(n)$ is asymptotically smaller than $\left(\frac{n}{\ln n}\right)^n$. More on Bell numbers can be found, eg, in [21]

3. $w(v_i, \mathcal{E}_j) = \widehat{R}(\mathcal{C}^*_v \setminus \mathcal{E}_j, d, T_{v_i})$, and $w(v_i, \phi_j) = \widehat{R}(\mathcal{C}^*_v, d, T_{v_i})$. That is, the weight function of each edge is the value of $\widehat{R}$ which corresponds to the subtree T_{v_i} and the set of colors $\mathcal{E}_j$.

Observation 4.11 *Given a partition $\{\mathcal{E}_1, \ldots, \mathcal{E}_\ell\}$, a min weight perfect matching on the above graph, outputs the minimum cost recoloring out of all the k-ordered partitions of $\mathcal{C}^*_v \setminus (\mathcal{D} \cup \{d\})$ in which the non empty sets are $\{\mathcal{E}_1, \ldots, \mathcal{E}_\ell\}$.*

We remark that performing the task of Observation 4.11 above requires in the original algorithm to consider $\binom{\Delta}{\ell}\ell!$ distinct ordered partitions, a number which is not necessarilly bounded by a function of the form $Poly(n)f(n^*_c)$.

Theorem 4.12 *Using the above construction, the runing time of the algorithm is $O(n^4 n^*_c Bell(n^*_c))$.*

Proof. We first observe that at a vertex v, the min-weight perfect matching is executed once per every (unordered) partition of $\mathcal{C}^*_v \setminus \mathcal{D}$. The number of such partitions is bounded by $O(Bell(n^*_c))$. Using the Hungarian algorithm for minimum weighted perfect matching in a bipartite graph [16] which runs in time $O(n^3)$, yields the following bound. ∎

We note that applying the same technique to the algorithm for non-uniform cost, provides an FPT algorithm in which the parameter is the number of colors (and not the cost of the optimal solution).

5 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. We considered few natural definitions for that distance, along with few model variants of the problem, and proved that the problem is NP-Hard in each of them. We then presented exact algorithms to solve the problem under the non-uniform and the uniform cost models.

Few interesting research directions which suggest themselves are:

- Is there an efficient algorithm for the "big convex recoloring" problem for any fixed number of colors?
- Similarly to the above, but rather then bounding the number of colors, the bound now is on the number of color changes, which is the recoloring distance from convexity. The goal is to decide whether there is a tree within this distance from a perfect phylogeny over the given set of characters. This corresponds to a fixed parameter tractable algorithm for constructing an optimal tree.
- Can our results for the uniform cost model from Section 4.2 be extended for the non-uniform cost model.

- Phylogenetic network are accumulating popularity as a model for describing evolutionary history. This trend, motivates the extension of our problem to more generic cases such as directed acyclic graphs or general graphs. It would be interesting to explore the properties of convexity on these types of graphs.

Acknowledgments

We would like to thank Mike Fellows, Shiri Moran, Rami Reshef, Mike Steel, Zohar Yakhni and Irad Yavneh For enlightening discussions and helpful comments. We would also like to thanks two anonymous referees for their insightful and constructive remarks.

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] J. H. Badger, P. E. Kearney, M. Li, J. Tsang, and T. Jiang. Selecting the branches for an evolutionary tree.: A polynomial time approximation scheme. *J. Algorithms*, 51.

[3] A. Ben-Dor, N. Friedman, and Z. Yakhini. Class discovery in gene expression data. In *RECOMB*, pages 31–38, 2001.

[4] M. Bittner and et.al. Molecular classification of cutaneous malignant melanoma by gene expression profiling. *Nature*, 406(6795):536–40, 2000.

[5] H.L. Bodlaender, M.R. Fellows, and T. Warnow. Two strikes against perfect phylogeny. In *ICALP*, pages 273–283, 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] O. Eulenstein, D. Chen, J.G. Burleigh, D. Fernandez-Baca, and M.J. Sanderson. Performance of flip supertrees with a heuristic algorithm. *Systematic Biology*, 53(2):299–308, 2004.

[9] D. Fernndez-Baca and J. Lagergren. A polynomial-time algorithm for near-perfect phylogeny. *SIAM Journal on Computing*, 32(5):1115–1127, 2003.

[10] W. M. Fitch. A non-sequential method for constructing trees and hierarchical classifications. *Journal of Molecular Evolution*, 18(1):30–37, 1981.

[11] L.A. Goldberg, P.W. Goldberg, C.A. Phillips, Z Sweedyk, and T. Warnow. Minimizing phylogenetic number to find good evolutionary trees. *Discrete Applied Mathematics*, 71:111–136, 1996.

[12] D. Gusfield. Efficient algorithms for inferring evolutionary history. *Networks*, 21:19–28, 1991.

[13] A. Hirsh, A. Tsolaki, K. DeRiemer, M. Feldman, and P. Small. From the cover: Stable association between strains of mycobacterium tuberculosis and their human host populations. *PNAS*, 101:4871–4876, 2004.

[14] S. Kannan and T. Warnow. Inferring evolutionary history from DNA sequences. *SIAM J. Computing*, 23(3):713–737, 1994.

[15] 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.

[16] H.W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, pages 83–97, 1955.

[17] S. Moran and S. Snir. Convex recoloring of strings and trees. Technical Report CS-2003-13, Technion, November 2003.

[18] D. Sankoff. Minimal mutation trees of sequences. *SIAM Journal on Applied Mathematics*, 28:35–42, 1975.

[19] C. Semple and M.A. Steel. *Phylogenetics*. Oxford University Press, 2003.

[20] M. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification*, 9(1):91–116, 1992.

[21] Eric W. Weisstein. Bell number. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/BellNumber.html.