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

Shlomo Moran[1] and Sagi Snir[2]

[1] Computer Science dept., Technion, Haifa 32000, Israel
moran@cs.technion.ac.il
[2] Mathematics dept. University of California, Berkeley, CA 94720, USA
ssagi@math.berkeley.edu

Abstract. A coloring of a tree is convex if the vertices that pertain to any color induce a connected subtree. 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 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 *uniform weighted* model and the *non-uniform* model. Our first algorithms find optimal convex recolorings of strings and bounded degree trees under the non-uniform model in linear time for any fixed number of colors. Next we improve these algorithms for the uniform model to run in linear time for any fixed number of *bad* colors. Finally, we generalize the above result to hold for trees of unbounded degree.

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

* This research was supported by the Technion VPR-fund and by the Bernard Elkin Chair in Computer Science.

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 posses 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 make 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. [10, 12, 1, 13, 4, 17]), 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) [8, 15] 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). [9] 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 [3, 2] 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 [11], 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 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 ([5]) 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.

Due to space limitation, figures, proofs of theorems and some of the results were removed. However, all of these can be found at the full paper at *http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-info.cgi?2004/CS/CS-2004-14*

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 $\{C(u) : u \in U\}$. For a subtree $T' = (V(T'), E(T'))$ of T, $C(T')$ denotes the set $C(V(T'))$. 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 [7] 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 [16]). 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 [6] 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 vertices 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 [15], 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 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 $C'(v) = d' \neq d$. Therefore, in the uniform cost model, 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. 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.

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. 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. It is easy to see that in any convex recoloring that is at most at distance k from C, non of the junk segments is recolored, what implies that the order of the segments, informative and non informative, does not matter. 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 $c_{i,\ell_1}, c_{i,\ell_2}, c_{i,\ell_3}$

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 $c_{i,x_j}, i = 1, \ldots, m$. 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).

Theorem 2. *Let (S, C) be the colored string defined by the above reduction. Then $OPT(S, C) \leq k$ iff F is satisfiable.*

The following two results pertain to restricted versions of the original problem which model specific problem in phylogenetics:

Theorem 3. *Minimal unweighted convex recoloring of a tree is NP-Complete even when the coloring is restricted to the leaves of the tree only.*

Theorem 4. *Minimal unweighted convex recoloring of a totally colored tree is NP-Hard even when recoloring a complete block is considered as a single operation.*

Note: In a Zebra string, overwriting a single vertex is also a block recoloring. Thus Theorem 4 also implies that the problem of minimizing the total number of vertex recoloring *and* block recoloring needed to transform a colored string to 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 optimal convex recolorings returned by the algorithms 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*. The set of bad colors for the input (T, C) is denoted by $\mathcal{C}^*$, and $|\mathcal{C}^*|$ is denoted by n_c^*.

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.

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 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).*

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

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

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

Theorem 6 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 $O(n \cdot n_c 2^{n_c})$.

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 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. *For a set of colors $\mathcal{D} \subseteq \mathcal{C}$, a color $d \in V$ 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 7. *Let v be an internal vertex with children $v_1, \ldots, v_k$. Then, for a color set $\mathcal{D}$ and a color $d \notin \mathcal{D}$:*

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

Theorem 7 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.

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

Note that Observation 8 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 9. *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 $v \in V(S)$:

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

where $\hat{d}$ is a new color.

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).

Theorem 10. *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 $(S, \hat{C})$.*

Theorem 10 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 by:

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

where R is defined in Section 4.1, 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.

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.

Let $(T = (V, E), C)$ be a colored tree. 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 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 $d \in \mathcal{C}_v^* \backslash \mathcal{D}$, 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 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 overwritten only by 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 1. *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.*

Let $\hat{C}$ be a convex total recoloring satisfying Lemma 1. 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 1, 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 2. *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:*

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

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

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

Lemma 2 implies a dynamic programming algorithm similar to the one presented in Section 4.1. 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 2(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 2(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 3. *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 2 (1) and (2) are computed by $v_1, \ldots, v_k$.*

Combining the results so far, we have

Theorem 11. *Optimal convex recoloring of totally colored trees with n vertices can be computed in $O(n \cdot n^*_c \Delta^{n^*_c+2})$ time, where n^*_c is the number of bad colors and Δ is the maximum degree of vertices in T.*

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, motivated by the scenario of introducing a new character to an existing phylogenetic tree. 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:

– Similarly to the generalization of the small parsimony question to the general one: Given a set of characters (colorings) such that the number of colors of each character is bounded by a fixed small constant, 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, 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 are 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.

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. A. Ben-Dor, N. Friedman, and Z. Yakhini. Class discovery in gene expression data. In *RECOMB*, pages 31–38, 2001.
3. M. Bittner and et.al. Molecular classification of cutaneous malignant melanoma by gene expression profiling. *Nature*, 406(6795):536–40, 2000.
4. H.L. Bodlaender, M.R. Fellows, and T. Warnow. Two strikes against perfect phylogeny. In *ICALP*, pages 273–283, 1992.
5. R. G. Downey and M. R. Fellows. newblock *Parameterized Complexity.* Springer, 1999.
6. A. Dress and M.A. Steel. Convex tree realizations of partitions. *Applied Mathematics Letters,*, 5(3):3–6, 1992.
7. D. Fernndez-Baca and J. Lagergren. A polynomial-time algorithm for near-perfect phylogeny. *SIAM Journal on Computing*, 32(5):1115–1127, 2003.
8. W. M. Fitch. A non-sequential method for constructing trees and hierarchical classifications. *Journal of Molecular Evolution*, 18(1):30–37, 1981.
9. 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.
10. D. Gusfield. Efficient algorithms for inferring evolutionary history. *Networks*, 21:19–28, 1991.

11. 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.
12. S. Kannan and T. Warnow. Inferring evolutionary history from DNA sequences. *SIAM J. Computing*, 23(3):713–737, 1994.
13. 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.
14. S. Moran and S. Snir. Convex recoloring of strings and trees. Technical Report CS-2003-13, Technion, November 2003.
15. D. Sankoff. Minimal mutation trees of sequences. *SIAM Journal on Applied Mathematics*, 28:35–42, 1975.
16. C. Semple and M.A. Steel. *Phylogenetics.* Oxford University Press, 2003.
17. M. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification*, 9(1):91–116, 1992.