Partial Convex Recolorings of Trees and Galled Networks: Tight Upper and Lower bounds

SHLOMO MORAN
Technion, Haifa
SAGI SNIR
University of California, Berkeley
and
WING-KIN SUNG
National University of Singapore

A coloring of a graph is convex if the vertices that pertain to any color induce a connected subgraph; a partial coloring (which assigns colors to a subset of the vertices) is convex if it can be completed to a convex (total) coloring. Convex coloring has applications in fields such as phylogenetics, communication or transportation networks, etc.

When a coloring of a graph is not convex, a natural question is *how far* it is from a convex one. This problem is denoted as *convex recoloring* (CR).
While the initial works on CR defined and studied the problem on trees, recent efforts aim at either generalizing the underlying graphs or specializing the input colorings.

In this work we extend the underlying graph and the input coloring to *partially colored galled networks*. We show that although determining whether a coloring is convex on an arbitrary network is hard, it can be found efficiently on galled networks. We present a fixed parameter tractable algorithm which finds the recoloring distance of such a network whose running time is quadratic in the network size and exponential in that distance. This complexity is achieved by amortized analysis that uses a novel technique for contracting colored graphs which seems to be of independent interest.

Categories and Subject Descriptors: F.2.0 [**Analysis of algorithms and problem complexity**]: General; G.2.1 [**Combinatorics**]: Combinatorial algorithms; G.2.2 [**Graph Theory**]: Trees

General Terms: Algorithms

Additional Key Words and Phrases: Convex recoloring, NP-hardness, partially colored galled networks, partially colored trees

1. INTRODUCTION

Given a set of colors $\mathcal{C}$ and a graph $G = (V, E)$ with n nodes, a (total) coloring of G is a function from the set of nodes V to the set of colors C. The coloring C is

Author's address: S. Moran, Computer Science dept., Technion, Haifa 32000, Israel.
S. Snir, Mathematics dept. University of California, Berkeley, CA 94720, USA.
W.-K. Sung, Department of Computer Science, National University of Singapore, Singapore.

called *convex* if the nodes that are colored with each given color induce a connected subgraph. A *partial* coloring of G is a coloring which maps some nodes to a "null color" $\lambda \notin \mathcal{C}$. A partial coloring is convex if it can be transformed to a total convex coloring by assigning colors to the nodes colored by λ.

Given an input coloring C of G, any other coloring C' of G can be viewed as a *recoloring* of C. The *recoloring distance* between C and C' is the number of nodes $v \in V$ such that $C(v) \neq \lambda$ and $C(v) \neq C'(v)$. The *minimum convex recoloring problem* or simply convex recoloring (CR), is to find for a given input coloring C a convex recoloring C' which minimizes the recoloring distance to C. The corresponding minimal recoloring distance is denoted as $opt(C)$.

Convex recoloring is a fundamental problem in combinatorial optimization and is related to practical applications in several fields. In phylogenetic analysis, the *perfect phylogeny* problem is defined as follows: Given a set of colorings on a set of items (species), find a tree such that the items label the tree leaves and each such (partial) coloring is convex on that tree [Semple and Steel 2003; Moran and Snir 2008]. In communication, vertices represent routers and a common color represents a common interest or a subnetwork. The task is to decide whether, under the given coloring, routers belonging to the same network are connected (see, e.g. [Chen et al. 2006] for a short discussion of these applications).

CR was introduced in [Moran and Snir 2008] where it was shown that the problem is NP-hard for totally colored and leaf-colored trees (and even paths). It was also shown there that when the tree T is totally colored and the degree is bounded by Δ, T can be convex recolored in $O(n\beta\Delta^{\beta+2})$ time, where β is the number of colors that violate convexity in the input tree. This yields a fixed parameter tractable (FPT) algorithm [Downey and Fellows 1999] for CR when the parameter is the recoloring distance. Subsequently, several new FPT criteria were devised or improved [Razgon 2007; Bodlaender et al. 2007; 2006]. See [Ponta 2007] for details on FPT approaches to CR.

Approximation of CR was also investigated. In [Moran and Snir 2007] a 3-approximation algorithm to convex recolor a tree which is partially colored is given. The running time of the algorithm is $O(cn^2)$, where c is the total number of colors in the tree. This was improved in [Bar-Yehuda et al. 2005], which gives a $(2+\epsilon)$-approximation algorithm whose running time is $O(n^2 + (1/\epsilon)^2 4^\epsilon)$.

Due to applicability of convex coloring in more general graphs than trees, recent studies extend CR to other graph families [Chor et al. 2007; Ponta 2007] where r-connectivity that generalizes CR is studied. In this paper we show that finding optimal convex recoloring in general graphs is NP-hard, even when there are only 2 colors and the coloring is total (ie there are no uncolored nodes). A galled network is a graph in which a node is a member in at most one cycle. A positive result we show here is an $O(n^2|\mathcal{C}|\Delta^{\alpha+2})$-time algorithm for the convex recoloring problem on both partially colored trees and partially colored galled networks, where α is the number of colors that violate convexity in some strong sense (α is different from β above). When the maximal degree Δ is unbounded, we improve the algorithm by a technique based on maximum weighted matching, and give an $O(poly(n)(\frac{\alpha}{\log \alpha})^\alpha)$-time algorithm for partially colored galled networks (and hence also trees). Finally, we show that α is linearly bounded from above by the size of the solution, i.e. the

number of recolored nodes in an optimal solution. This implies that our algorithm is an FPT algorithm for the problem, where the parameter is the size of the optimal solution. A byproduct of this result is a polynomial time algorithm for deciding whether a colored galled network is convex. Amortized analysis [Cormen et al. 2001] is an approach for analyzing iterative algorithm complexities when complexity might change substantially between different iterations. Our upper bound on the complexity of the algorithm is achieved by a novel technique of amortized analysis, which provides a *lower* bound on the size of the optimal solution. Specifically, we show that although uncoloring a single node can result in "rehabilitation" of many "compound" colors (to be defined), the average number of such colors per a single uncoloring is constant. As CR can be perceived as a covering problem (see [Moran and Snir 2008; 2007] for the formal definition) where techniques for such problems are applicable [Moran and Snir 2007; Bar-Yehuda et al. 2005], we believe that this technique might be of independent interest and can be utilized in different applications.

The rest of this paper is organized as follows: In the next section we provide formal definitions that will be used throughout the rest of the paper. In Section 3 we show that the problem is NP-hard even when the set of colors is of cardinality two. In Section 4 we provide a dynamic programming algorithm for obtaining an optimal convex coloring for a galled network and in Section 5 we present a variant of this algorithm which is more efficient when the maximal degree is large. Section 6 gives a lower bound on the size of the solution by the number of compound colors. We conclude in Section 7 with conclusion and future research directions.

2. PRELIMINARIES

In this section we give formal definitions to the notions introduced in the previous section. We consider a special type of graph: a galled network, which is an undirected graph where every node occurs in at most one cycle. See Figure 1 for an example.

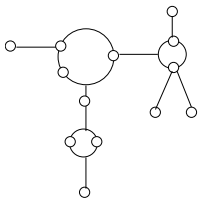

Fig. 1. A galled network: every node is a member in at most one cycle.

Throughout the paper we will use the notion of an induced graph. We here define it formally: Let $G = (V, E)$ be a graph. For a subset $V' \subseteq V$, the *graph induced by* V', $G(V')$ is the graph (V', E') where E' is all the edges in E with two endpoints in V'.

Since we focus on connected sets of vertices with a common feature (color) the notion of a *carrier* is central throughout the whole paper. Informally, a carrier of a

set of vertices U represents a minimal connected subgraph containing U. A formal definition follows.

Definition 2.1. Let $U \subseteq V$ be a set of nodes. Then a set of nodes W is a *carrier* of U if

(1) $U \subseteq W$,
(2) $G(W)$, the subgraph induced by W, is connected,
(3) No proper subset of W has the above two properties

We comment that in general there can be many possible carriers for a given set U, but carriers on trees are uniquely defined - which seems to be a main reason why convex recoloring on tree is easier than on more complicated graphs.
In the algorithmic part as well as in the analysis
(Sections 4,6), we will have a special treatment for cycles. Therefore, we make this distinction:

Definition 2.2. An edge (node) is a *tree edge (node)* if it is not on a cycle, otherwise it is a *cycle edge (node)*.

Connected subgraphs whose removal from the graph leaves at most two connected components are used both in our lower and upper bounds. For this we define:

Definition 2.3. For a subset of nodes $W \subseteq V$, the *super degree* (or *s-degree* for short) of W is the number of connected components in $G(V \setminus W)$.

Throughout, we will consider super degrees of sets W for which the induced graph $G(W)$ is connected.

2.1 Colored Graphs

Consider a set of colors $\mathcal{C}$, and a *null color* $\lambda \notin \mathcal{C}$. A colored graph is a pair $(G = (V, E), C)$ where C is a coloring of G, that is, a function $C : V \to \mathcal{C} \cup \{\lambda\}$. C is said to be *total* if $C(v) \neq \lambda$ for every $v \in V$ and *partial* otherwise. Consider a coloring C and a color $d \in \mathcal{C}$. We say that a set $W \subseteq V$ is a *carrier*(C, d) under C (or just *d-carrier* for short when it is clear from the context) if W is a carrier of $C^{-1}(d)$ (i.e., W is a carrier of all the nodes colored by d).

Definition 2.4. A colored galled network (GN, C) is convex if there exists a set $\{CA_d : d \in C(V) \setminus \{\lambda\}\}$ such that for each d, CA_d is a d-carrier, and for each pair of colors $\{d, d'\}$, it holds that $CA_d \cap CA_{d'} = \emptyset$.

Alternatively, we say that a coloring C is convex if the null-colored nodes can be colored (or C can be *completed* to a total coloring), such that every color induces a connected subgraph.

The distinction between "easy to handle" colors and "hard to handle" colors is crucial in this work. Informally, a color d is easy (called *pure*) if G can be splitted to at most three connected components, such that one component contains all the vertices colored by d and no other color (but possibly null colored vertices). See Figure 2. A color which is not pure is *compound*.

Definition 2.5. A color $d \in \mathcal{C}$ is called *pure* in a partially colored graph (G, C) if there exists a d-carrier W with s-degree at most two and which does not contain nodes colored by a color $d' \neq d$ (See Figure 2.). Otherwise, d is *compound*.

The number of compound colors in a colored graph (G, C) is strongly related to both upper and lower bounds of our algorithms, and is denoted by α.

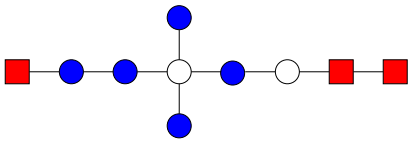

Fig. 2. Empty triangles are uncolored nodes. The color ● is pure since it has a (single) carrier containing only ● or uncolored nodes, and the number of neighbors of this carrier is two ($s - degree = 2$). The color ■ is compound since it has no such carrier (its sole carrier contains nodes of different color).

2.2 Recoloring

Given a colored graph (G, C), any other coloring C' of G is viewed as a *recoloring* of C. A recoloring C' of C is said to *retain* (the color of) a node v if $C(v) = C'(v)$. If $C(v) \neq \lambda$ and $C(v) \neq C'(v)$, C' *overwrites* v. We say that C' *uncolors* a node v if $C(v) \neq \lambda$ and $C'(v) = \lambda$ (that is, an erasure — a special type of an overwrite). For a subset U of V, the restriction of C to U, $C|_U$, is the coloring obtained from C by uncoloring all the nodes in $V \setminus U$.

For each recoloring C' of C we associate a cost, $cost_C(C')$ (or $cost(C')$ when C is clear), which is the number of nodes overwritten by C'. The convex-recoloring cost of a coloring C is defined by $opt(C) = min_{C'}\{cost_C(C')\}$, where the minimum is taken over all convex recolorings of C. A convex recoloring C^* with $cost(C^*) = opt(C)$ is an *optimal convex recoloring of* C.

Observe that if a colored graph $(G = (V, E), C)$ can be convex recolored by overwriting a subset X of V, then the colored graph can also be convex recolored by uncolored X. Hence, computing an optimal (convex) recoloring can be defined as a minimum cost covering problem, as follows: Let $(G = (V, E), C)$ be a colored graph. We say that a set of nodes X is a *convex cover* (or just a cover) for (G, C) if the (partial) coloring $(G, C|_{[V \setminus X]})$ is convex (i.e., C can be transformed to a convex coloring by overwriting the nodes in X).

We complete this section with a definition and a simple observation which will be useful in the sequel. We consider only special types of convex recolorings of the given coloring C. Informally, these recolorings do not introduce new colors rather use only original colors. Moreover, for every color in the recoloring, at least one vertex retained its original color. Therefore we say that this color was expanded from this (or others) vertex. Formally: let (G, C) be a (partially) colored graph and let C' be a total convex recoloring of C with $\mathcal{C}' = C'(V)$. A color $c \in \mathcal{C}'$ is *expanding* with respect to C and C' (or simply *expanding*) if there is at least one node v s.t. $C(v) = C'(v) = c$. The coloring C' is an *expanding* recoloring of C if every color $c \in \mathcal{C}'$ is expanding.

OBSERVATION 2.6. *Let (G, C) be a colored graph. Then there exists an optimal convex recoloring of C which is an expanding 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 for every color $c \in C'(V)$ there is a node v s.t. $C(c) = C'(v) = c$.

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

3. NP HARDNESS

This section shows that computing the convex recoloring cost of a colored graph is NP-hard even when we have only 2 colors (i.e. $\mathcal{C} = \{1, 2\}$) and the coloring is total (ie $C(v) \neq \lambda$ for all nodes v). Our proof is based on a polynomial-time reduction from the set cover problem (which is an NP-complete problem, see [Garey and Johnson 1979]).

Set Cover Problem: Given a set V, a family $\mathcal{F}$ of subsets of V, and an integer k ($k < |\mathcal{F}|$), can we find at most k sets $F_1, F_2, \ldots, F_k$ in $\mathcal{F}$ such that $V = F_1 \cup F_2 \cup \ldots F_k$?

An input to the Set Cover problem as above is reduced to a colored graph (G, C) as follows.

—$G = (V \cup \mathcal{F}, E)$ where $E = \{(v, F) \mid F \in \mathcal{F}, v \in F\} \cup \{(F_1, F_2) \mid F_1, F_2 \in \mathcal{F}\}$.
—C is the total coloring defined by $C(v) = 1$ for $v \in V$ and $C(F) = 2$ for $F \in \mathcal{F}$.

It is easy to check that (G, C) can be constructed in $O(|V||\mathcal{F}|)$ time.

LEMMA 3.1. *Suppose $\mathcal{F}' = \{F_{i_1}, F_{i_2}, \ldots, F_{i_k}\} \subseteq \mathcal{F}$ is a set cover of V. Then, $\mathcal{F}'$ is a convex cover of (G, C).*

PROOF. We will show that if all nodes in $\mathcal{F}'$ are recolored by 1, the coloring becomes convex. Since $\mathcal{F}$ is a clique, the 2-nodes still induce a clique. The 1-nodes from a connected subgraph since $\mathcal{F}'$ is a set cover. □

LEMMA 3.2. *Let $S \subseteq V \cup \mathcal{F}$ be a convex cover of (G, C). Then there exists a set $T \subseteq \mathcal{F}$ s.t. $|T| \leq |S|$ and T is a set cover of V.*

PROOF. If $|V| \leq 1$ then C is convex and the lemma holds, so assume that $|V| > 1$. Let $S_1 = S \cap \mathcal{F}$ and $S_2 = S \cap V$. Suppose $S_2 = \{v_1, v_2, \ldots, v_x\}$. For every $v_i \in S_2$, let F_i be a set in $\mathcal{F}$ such that $v_i \in F_i$. Let $T = \{F_1, F_2, \ldots, F_x\} \cup S_1$. Note that $|T| \leq |S|$.

By definition, $S_2 \subseteq F_1 \cup F_2 \cup \ldots \cup F_x$. Since $|V| > 1$ and S is a convex cover, for every $v \in V \setminus S_2$, there exists $F \in S_1$ such that $v \in F$. Hence, $V \setminus S_2 \subseteq \cup_{F \in S_1} F$. Thus, T is a set cover of V. □

By the above two lemmas, we have the following theorem.

THEOREM 3.3. *The convex recoloring of a graph is NP-hard even when restricted to graphs which are totally colored by two colors.*

4. A DYNAMIC PROGRAMMING ALGORITHM FOR COMPUTING A COVER

4.1 High Level Overview

We start with a very high-level overview of the algorithm. The general structure of the algorithm is a standard bottom-up dynamic programming type (see e.g. [Sankoff 1975]) but with proper modifications to handle convexity and/or cycles. We root the graph arbitrarily and work from leaves to top. At every node, we keep for certain subsets of colors, the cost of the optimal recoloring of the graph rooted at that node, using colors from these subsets solely. The main challenge we encounter is to minimize the number of subsets we keep, while guaranteeing that there is an optimal recoloring with one of these subsets. We handle cycles in some novel technique of "untying" the cycle at each of its edges and subsequently solve it with our tree algorithm.

4.2 The Formal Algorithm

Consider a colored galled network (G, C) where G is of size n and maximum degree Δ. Let $U = (V, E)$ be a rooted acyclic galled network formed by rooting G at an arbitrary node r, from which all other nodes are accessible by directed paths. For any cycle CY (in the underlying graph), there is a single node in CY which is either the root r or it has an incoming tree edge. This node is denoted as a *split node*. Cycle edges are directed in an acyclic manner so that every node in the cycle is reachable from the split node.

Let A be the set of all split nodes and tree nodes in U. For every $v \in A$, let U_v be the subgraph reachable from v (i.e. $U_r = U$). $V(U_v)$ denotes the set of nodes in U_v.

Definition 4.1. For a set of colors $\mathcal{D}$, $opt(\mathcal{D}, U_v)$ is the minimum cost of a convex recoloring of the colored galled network $(U_v, C|_{V(U_v)})$ with the constraint that the recolored graph uses colors from $\mathcal{D}$ only.

Definition 4.2. $opt(\mathcal{D}, U_v, d)$ is the minimum cost of a convex recoloring of the colored galled network $(U_v, C|_{V(U_v)})$ with the constraint that the recolored graph uses colors from $\mathcal{D}$ only and the color of node v is d (for some $d \in \mathcal{D}$).

Note that our aim is to compute $opt(\mathcal{C}, U_r)$. We also have the following observation.

OBSERVATION 4.3. *For all $v \in A$, $\mathcal{D} \subseteq \mathcal{C}$, $opt(\mathcal{D}, U_v) = \min_{d \in \mathcal{D}} opt(\mathcal{D}, U_v, d)$.*

Our algorithm is based on the improvement of [Bar-Yehuda et al. 2005] to the basic dynamic programming algorithm of [Moran and Snir 2008], based on the following observation: According to Observation 2.6, the convex recoloring cost can be computed by considering only *expanding* convex recolorings. Hence it suffices to compute $opt(\mathcal{D}, U_v, d)$ only for sets $\mathcal{D}$ of colors which are used in some expanding convex recoloring of U. We call such sets of colors *legal*. The following definition characterizes these legal sets ($\overline{\mathcal{D}}$ denotes $C(U) \setminus \mathcal{D}$, i.e. the complement of $\mathcal{D}$).

Definition 4.4. A set of colors $\mathcal{D}$ is *legal for* U_v *and* d if $\mathcal{D} \setminus \{d\} \subseteq C(U_v)$ and $\overline{\mathcal{D}} \setminus \{d\} \subseteq C(U \setminus U_v)$.

Example: let $C(U_v) = \{1,2\}$, $C(U \setminus U_v) = \{1,3,4\}$ and d=3. Then $\mathcal{D}_1 = \{1,2,3\}$ is legal for U_v and d but $\mathcal{D}_2 = \{1,3\}$ is not (since $\overline{\mathcal{D}_2} \setminus \{d\} = \{2,4\} \not\subseteq \{1,3,4\} = C(U \setminus U_v)$). The following observation follows directly from the definition of expanding convex recoloring.

OBSERVATION 4.5. *Let C' be an expanding convex recoloring of (U,C), and assume that for some tree node $v \in A$, $C'(v) = d$. Then $C'(U_v) \subseteq \mathcal{D}$ for some set $\mathcal{D}$ which is legal for U_v and d.*

Below, we describe a recursive formula for computing $opt(\mathcal{D}, U_v, d)$ for all $v \in A$ and all sets $\mathcal{D}$ which are legal for U_v and d. By Observations 4.5 and 2.6, this suffices for computing optimal convex recolorings.

For any $v \in A$, we distinguish between the cases where v is a leaf, a tree node, or a split node.

Leaf Nodes

For every leaf node v of U, $\mathcal{D} \subseteq \mathcal{C}$, and $d \in \mathcal{D}$, we have $opt(\mathcal{D}, U_v, d) = \delta(d, C(v))$ where $\delta(c,c') = 1$ if $c \neq c'$ and 0, otherwise.

Tree Nodes

For defining the recursive formula for a tree node v, we use the following notations.

Definition 4.6. An *ordered partition* of a set S into k subsets is a tuple $(S_1, \ldots, S_k)$ of disjoint subsets of S, some of which may be empty, s.t. $\cup_{i=1}^{k} S_i = S$.

Definition 4.7. Let v be a tree node with k children $v_1, \ldots, v_k$, d be a color, and $\mathcal{D}$ be a legal set for U_v and d. A *legal ordered partition for v, $\mathcal{D}$ and d* is an ordered partition $(\mathcal{D}_1, \ldots, \mathcal{D}_k)$ of $\mathcal{D} \setminus \{d\}$ s.t. for each i, $\mathcal{D}_i$ is legal for U_{v_i} and d.

The set of legal ordered partitions for v, $\mathcal{D}$ and d is denoted $\mathcal{LP}(v, \mathcal{D}, d)$. $N_{\mathcal{P}}(v, \mathcal{D}, d)$ is the cardinality of $\mathcal{LP}(v, \mathcal{D}, d)$. By convention, $N_{\mathcal{P}}(v, \mathcal{D}, d) = 0$ if $\mathcal{D}$ is not legal for U_v and d.

Legal ordered partitions are related to expanding recoloring by the following observation, which is implied directly by Observation 4.5.

OBSERVATION 4.8. *Let C' be an expanding convex recoloring of (G,C), let v be a tree node with k children $v_1, \ldots, v_k$, and let $C'(v) = d$. Then there is a set $\mathcal{D}$ which is legal for U_v and d, and a legal ordered partition $(\mathcal{D}_1, \ldots, \mathcal{D}_k) \in \mathcal{LP}(v, \mathcal{D}, d)$ s.t. $C(U_v) \subseteq \mathcal{D}$ and for all i, $C(U_{v_i}) \subseteq \mathcal{D}_i \cup \{d\}$.*

By Observation 2.6, it suffices to consider only colorings which correspond to legal ordered partitions as in Observation 4.8 above. We bound the complexity of the algorithm by bounding the number of legal ordered partitions $N_{\mathcal{P}}(v, \mathcal{D}, d)$. For this, let v be a node and c be a color.

Definition 4.9. $index(v,c)$ is the number of connected components of $U(V \setminus \{v\})$ (the subgraph of U induced by $V \setminus \{v\}$), which contain nodes colored by c.

$index(v,c)$ can be found easily by a simple BFS search from v. For the singular case that v is the only node of its color class, we define $index(v, C(v)) = 1$, so we get $index(v,c) \geq 1$ for every color c. Observe that for a tree node v, $index(v,c) > 1$ iff v lies in a path connecting two nodes of color c, which implies the following.

OBSERVATION 4.10. *If $|\mathcal{C}| > 2$ then for each tree node v there is at most one pure color c for which $index(v,c) > 1$.*

PROOF. Assume by contradiction that $|\mathcal{C}| > 2$ and there are two pure colors c, c' for which $index(v,c) > 1$ and $index(v,c') > 1$ (see Figure 3 left). Consider any two carriers of c and of c'. Then the s-degree of each of these carriers is at least 2 since v is a tree node and these carriers intersect at v. Adding a third color c'' will increase the s-degree of at least one of the carriers (see Figure 3 right), which is the desired contradiction. □

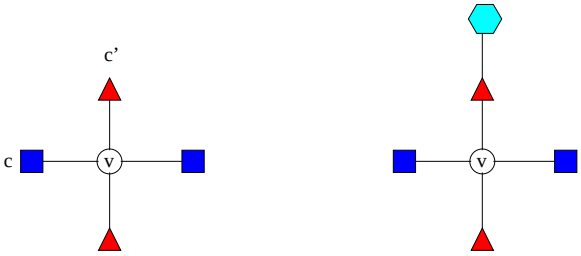

Fig. 3. For $|\mathcal{C}| > 2$, v is in the carriers of at most one pure color.

OBSERVATION 4.11. *Assume $|\mathcal{C}| > 2$. Then for $v \in V$ and $d \in \mathcal{C}$,*

$$\sum_{\mathcal{D} \subseteq \mathcal{C}} N_{\mathcal{P}}(v, \mathcal{D}, d) = \prod_{c \in \mathcal{C} \setminus \{d\}} index(v,c) \leq \Delta^{\alpha+1}$$

where Δ is the maximum degree of the tree T.

PROOF. The equality follows by the fact that each legal partition for v, $\mathcal{D}$ and d is defined by allocating each color $c \in \mathcal{C} \setminus \{d\}$ to one of the $index(v,c)$ components of $U(V \setminus \{v\})$ which contains a node colored by c. The inequality follows from the following facts: Only colors c for which $index(v,c) > 1$ contributes to $\prod_{c \in \mathcal{D} \setminus \{d\}} index(v,c)$, each such color contributes a multiplicative factor of at most Δ. There are α compound colors, and by Observation 4.10 at most one pure color c may have $index(v,c) > 1$. □

The algorithm proceeds in a bottom up fashion from leaves to root. For a tree node v, the algorithm computes $opt(\mathcal{D}, U_v, d)$ for every $d \in \mathcal{C}$ and for every $\mathcal{D}$ which is legal for U_v and d. Then by Observation 4.3, it computes $opt(\mathcal{D}, U_v)$.

Let $\hat{R}(\mathcal{D}, U_v, d) = \min\{opt(\mathcal{D} \setminus \{d\}, U_v), opt(\mathcal{D} \cup \{d\}, U_v, d)\}$, that is: $\hat{R}(\mathcal{D}, U_v, d)$ is the minimal cost of a convex recoloring of U_v, which uses only colors from $\mathcal{D} \cup \{d\}$ and if it uses the color d then $C'(v) = d$.

LEMMA 4.12. *Let v be a tree node with k children $v_1, v_2, \ldots, v_k$, d be a color, and $\mathcal{D}$ be legal for U_v and d. Then*

$$opt(\mathcal{D}, U_v, d) = \delta(d, C(v)) + \min_{(\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_k) \in \mathcal{LP}(\mathcal{D} \setminus \{d\}, v)} \sum_{i=1}^{k} \hat{R}(\mathcal{D}_i \cup \{d\}, U_{v_i}, d)$$

PROOF. The proof of the lemma goes along the lines of the proof of Theorem 4.3 in [Moran and Snir 2008].

$\leq$: Let $(\mathcal{D}'_1, \mathcal{D}'_2, \ldots, \mathcal{D}'_k)$ be a legal ordered partition for v $\mathcal{D}$ and d minimizing the r.h.s. of the equation. Then, the coloring C' of U_v obtained from the respective recolorings of the subtrees U_{v_i}, $i = 1, \ldots k$, and in addition setting $C'(v) = d$, is a convex recoloring of U_v satisfying $C'(U_v) \subseteq \mathcal{D} \cup \{d\}$ which attains the same cost.

$\geq$: By observation 2.6 there is an expanding convex recoloring C' s.t. $cost(C') = opt(\mathcal{D}, U_v, d)$. By Observations 4.8, there is a legal ordered partition $\mathcal{D}_1, \ldots, \mathcal{D}_k$ of $\mathcal{D}$ s.t. for each i, $C'(U_{v_i}) \subseteq \mathcal{D}_i \cup \{d\}$, and hence $cost(C')$ is bounded by the r.h.s. of the equation. □

COROLLARY 4.13. *Let v be a tree node with children $v_1, \ldots, v_k$. Assume that, for every $d \in \mathcal{C}$, every child v_i of v, every $\mathcal{D}_i$ legal for U_{v_i} and d, the values of $\hat{R}(\mathcal{D}_i, U_{v_i}, d)$ are given. Then, the values $opt(\mathcal{D}, U_v, d)$ for all $d \in \mathcal{C}$ and for all legal $\mathcal{D}$ with respect to d and U_v can be computed in $O(k|\mathcal{C}|\Delta^{\alpha+1}) = O(|\mathcal{C}|\Delta^{\alpha+2})$ time.*

PROOF. By Lemma 4.12 and observation 4.11. □

Split Nodes

Let v be a split node of U, and suppose it corresponds to a length $1 + \ell_v$ cycle $CY_v = (v = v_0, v_1, v_2, \ldots, v_{\ell_v}, v = v_0)$ in U. Note that for each $v_i \in CY_v$, all the children of v_i which are not in CY_v are either tree nodes or split nodes.

LEMMA 4.14. *Let v and CY_v be as above. Assume that for each node u which is not in CY_v but is a child of a node in CY_v, for all $d \in \mathcal{C}$, and for all $\mathcal{D}$ which are legal for U_u and d, the values of $\hat{R}(\mathcal{D}, U_u, d)$ are given. Then we can compute $opt(\mathcal{D}, U_v, d)$ in $O(\ell_v^2|\mathcal{C}|\Delta^{\alpha+2})$ time.*

PROOF. $opt(\mathcal{D}, U_v, d)$ is computed (for each legal set of colors $\mathcal{D}$ and each $d \in \mathcal{D}$) as follows: For each edge $e = (v_i, v_{i+1}), i = 0 \ldots \ell - 1$, we consider the subnetwork rooted at v in which e is deleted. In each of these l_v subnetworks, the split node v and all cycle nodes are tree nodes, so for each of these nodes we use the algorithm for tree nodes in Lemma 4.12 above, and then select the coloring of minimum cost among these ℓ_v colorings. Since this minimum coloring is convex for the subnetwork U_v with some edge (v_i, v_{i+1}) removed, it is also convex for U_v. We now prove that this coloring is optimal. This is an immediate corollary of the following claim.

CLAIM 4.15. *Let C' be any optimal expanding recoloring for $(U_v, C|_{V(U_v)})$ (recall that $C|_{V(U_v)}$ is the coloring induced by C on $V(U_v)$, the set of nodes of U_v). Then the algorithm described above returns a recoloring with cost at most the cost of C'.*

PROOF. We divide into two cases:

—All nodes of CY_v are colored by C' with a single color: Then, by the inductive arguments of Lemma 4.12, the cost of the coloring returned by the algorithm when *any* of the edges (v_i, v_{i+1}) is removed is at most the cost of C'.

—Not all nodes $V(CY_v) \setminus \{v\}$ are colored with the same color under C': Then for some $0 \leq i \leq \ell - 1$, v_i and v_{i+1} are colored with different colors under C'. Then, by the inductive arguments of Lemma 4.12, the coloring returned by the algorithm when edge (v_i, v_{i+1}) is removed has cost at most the cost of C'.

□

The bound on the running time of the algorithm follows from Corollary 4.13 and by the fact that we apply algorithm described in Lemma 4.12 ℓ_v times, each time on paths of length $O(\ell_v)$.

For computing the total complexity of the algorithm, let $l_v = 1$ for a tree node v. Then the sum of l_v^2 over all tree and split nodes is bounded by n^2. Thus we have the following theorem.

THEOREM 4.16. *$opt(\mathcal{C}, U_r)$ can be computed in $O(n^2|\mathcal{C}|\Delta^{\alpha+2})$ time.*

5. FIXED PARAMETER TRACTABLE ALGORITHM FOR UNBOUNDED DEGREE TREES

When $\Delta = O(\alpha)$ (and in particular for fixed Δ) the algorithm from previous section yields a Fixed Parameter Tractable (FPT) algorithm for the problem, where the parameter is the number of compound colors α (see [Downey and Fellows 1999]). However, when $\Delta = \Omega(n)$, the bound on the running time of the algorithm from previous section becomes $\Omega(n^{\alpha+4})$ which is *not* FPT in α. This section gives an algorithm whose running time is $poly(n)f(\alpha)$ (specifically $O(n^7)f(\alpha)$), making it FPT in α. In Section 6 we will show that for a given input coloring C, $opt(C) = O(\alpha)$ turning this algorithm into an FPT algorithm, where the parameter is the cost of the optimal solution.

Our improvement is based on a faster way to compute the recursive formula in Lemma 4.12 when $\Delta \gg \alpha$. Suppose v has children $v_1, v_2, \ldots, v_k$, and let $U_i = U_{v_i}$, $i = 1, \ldots, k$, and $U_0 = U \setminus U_v$. A straightforward computation of the formula in Lemma 4.12 requires the computation of the sum on the right hand side $N_{\mathcal{P}}(v, \mathcal{D}, d)$ times - once for each legal ordered partition for $v, \mathcal{D}$ and d. This may require checking order of k^α legal ordered partitions. When $k \gg \alpha$, a faster solution is obtained by using minimum weight matching as follows. Let $F = \{c : index(v, c) = 1\}$, and for each of the $k + 1$ connected components $U_0, \ldots, U_k$, let $F_i = F \cap C(V(U_i))$. Informally, the set F contains colors that appear only in one component of $G(V \setminus \{v\})$ and the set F_i contains the colors that appear only in U_i (note that $F_i \cap F_j = \emptyset$ for every $i \neq j$). The algorithm is based on the observation that the allocation of colors in F is fixed in legal ordered partitions. Specifically: Let $(\mathcal{D}_1, \ldots, \mathcal{D}_k)$ be a legal ordered partition for v, d and $\mathcal{D}$. Then for $i = 1, \ldots, k$, $\mathcal{D}_i$ must include F_i. Thus, a legal ordered partition as above is completely determined by the allocation of colors not in F. For this, we write each $\mathcal{D}_i$ as $\mathcal{D}_i = F_i \cup H_i$, where H_i consists of all the colors c in $\mathcal{D}_i$ for which $index(v, c) > 1$. Let $H = \cup_{i=1}^k H_i$ be the set of all colors c in $\mathcal{D} \setminus \{d\}$ for which $index(v, c) > 1$. By Observation 4.10, H contains at most one pure color, and hence $|H| \leq \alpha + 1$. Due to this inequality we modify the algorithm for a tree node as follows:

For a node v with k children we first compute $|H|$, i.e. the number of colors c for which $index(v, c) > 1$. If $|H| > k$ then, by Observation 4.10, $k \leq \alpha$. In this case

we proceed as in the previous section, and by Corollary 4.13 obtain running time $O(|\mathcal{C}|\alpha^{\alpha+2})$ for tree nodes.

If $|H| \leq k$, we sketch below how to compute $OPT(\mathcal{D}, d, v)$ by considering all *unordered* partitions of H. The number of such partitions is $Bell(|H|)$, where $Bell(t) = O((\frac{t}{\log t})^t)$ is the number of unordered partitions of t elements to any number of nonempty sets [Weisstein].

Let $\mathcal{P}(H)$ be the set of all unordered partitions of H and let $S = \{Z_1, \ldots, Z_\ell\} \in \mathcal{P}(H)$ be one such a partition. First we observe that $\ell \leq |H|$, and since $|H| \leq k$ it holds that $\ell \leq k$ as well.

Given the partition $S = \{Z_1, \ldots, Z_\ell\} \in \mathcal{P}(H)$, we define a weighted complete bipartite graph B_S of $2k$ nodes. The nodes in one side of B_S correspond to the k children of v, while the nodes in the other side correspond to sets of colors (with $index(v, c) > 1$) $Z_1 \ldots, Z_k$, where $Z_1, \ldots, Z_\ell$ are given by S, and $Z_{\ell+1} = \ldots = Z_k = \phi$. B_S is the complete bipartite graph $B_S = \{v_1, \ldots, v_k\} \times \{Z_1, \ldots, Z_k\}$. The edge weights $w(v_i, Z_j)$ are defined by:

$$w(v_i, Z_j) = \begin{cases} \hat{R}(F_j \cup Z_j \cup \{d\}, U_{v_i}, d) & \text{if } Z_j \subseteq H_i \\ \infty & \text{otherwise} \end{cases}$$

The weight of the edge holds the cost of coloring U_{v_i} with the colors $F_j \cup Z_j \cup \{d\}$. As the matchings in B_S correspond to the expanding convex recolorings defined by the partition S of H, we have:

Observation 5.1. *Consider a node v, a color d and a set $\mathcal{D}$ which is legal for v and d. Let $H = \{c \in \mathcal{D} \setminus \{d\} : index(v, c) > 1\}$. Then*

$$opt(\mathcal{D}, U_v, d) = \delta(d, C(v)) + \min_{S \in \mathcal{P}(H)} MWM(B_S)$$

where $MWM(B_S)$ is the minimum weight perfect bipartite matching of B_S.

Theorem 5.2. *Using the above technique the running time of the algorithm is $O(n^2\Delta^4|\mathcal{C}|Bell(\alpha+1)) \leq O(n^7 Bell(\alpha+1))$.*

Proof. Let H_v denote the set H for node v. First we show that using a preprocessing of $O(n|\mathcal{C}|)$ time, we can compute the set H_v for a given node v in $O(n|\mathcal{C}|)$ time:
For a given color c, $index(v, c)$ is computed for all $v \in V$ in $O(n)$ time by a post order search of the tree. Hence computing $index(v, c)$ for all $v \in V$ and $c \in \mathcal{C}$ can be done in $O(n|\mathcal{C}|)$ preprocessing time. Once this is done, computing the set H_v (for any $v \in V$) is doable in $O(n|\mathcal{C}|)$ time. When H_v is given, we compute the minimum weight perfect bipartite matching of B_S in $O(k^3) \leq O(\Delta^3)$ time [Kuhn 1955]. Since $|\mathcal{P}(H_v)| \leq Bell(|H_v|) \leq Bell(\alpha+1)$, the theorem follows by replacing the factor of $\Delta^{\alpha+1}$ in Theorem 4.16, which counts for the number of ordered partitions, by $\Delta^3 Bell(\alpha+1)$, which bounds the time needed to compute minimum weight perfect matchings over all unordered partitions. □

6. LOWER BOUND ON THE COVER SIZE

The previous section showed an algorithm for finding optimal convex cover (as defined in Section 2.2) whose running time is $poly(n)exp(\alpha)$, where n is the number

of nodes and α is the number of compound colors. In this section we show that the running time is bounded by $poly(n)exp(k)$, where k is the size of an optimal cover, by showing that the size of an optimal cover k is at least linear in the number of compound colors α. This implies that the algorithm in Section 5 is fixed parameter tractable when the parameter is the cover size. To simplify the presentation, we first show the lower bound for trees, and then extend it to arbitrary galled networks.

We first introduce the notion of a colored forest and an iterative operation, called elimination, acting on the forest. The elimination process is applied on a set of nodes of the forest. We show that a set of nodes is a (convex) cover if and only if the elimination process applied on it terminates with the empty forest.

Recall that for a tree $T = (V, E)$, each subset $U \subseteq V$ has a unique carrier, denoted by $carrier_T(U)$ (or $carrier(U)$ when T is clear). We will sometime use $carrier(U)$ to denote the (unique) subtree induced by $carrier(U)$. For a colored graph (G, C) and a color $d \in \mathcal{C}$, $carrier_G(C, d)$ is a carrier of $C^{-1}(d)$. When G is clear from the context, we just use $carrier(C, d)$.

Definition 6.1. We say that a colored tree (T, C) satisfies 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 of a tree is convex iff it satisfies the disjointness property.

The following observation is used later in this section.

Observation 6.2. *Let $T = (V, E)$ be a tree and let $U \subseteq V$. Then for each $v \in U$, $Carrier(U \setminus \{v\}) \neq carrier(U)$ iff v is a leaf in $carrier(U)$, and in this case $carrier(U \setminus \{v\})$ is obtained by removing from $carrier(U)$ a maximal path which starts from v and contains (other) nodes in $Carrier(U) \setminus U$ of degree ≤ 2 in $G(carrier(U))$.*

Figure 4 gives an example how a carrier is changed as a result of a node uncoloring.

6.1 Proper forests and their contractions

A partially colored forest (F, C) is a set of partially colored trees. A *proper forest* is a partially colored forest in which $C(T) \cap C(T') = \emptyset$ for each pair of distinct trees T, T' (in particular, any partially colored tree is a proper forest). A proper forest is *convex* if the coloring of each tree T in it is convex. An edge e in a proper forest is *neutral* if for every color d, $e \notin carrier(C, d)$. In the sequel, we will restrict ourselves only to proper forests, denoted by (F, C), where $F = (V, E)$.

Definition 6.3. The *contraction* of (F, C), denoted by $CON((F, C))$, is a proper forest obtained by:

—removing from F all the neutral edges, and then

—removing from the resulting forest all colorless and monochromatic trees (i.e. trees of at most one color)

Observation 6.4. *If (F, C) is proper, then $CON((F, C))$ is also proper.*

Proof. Removal of a neutral edge from a tree T will split it into two subtrees T_1 and T_2 where $C(T_1) \cap C(T_2) = \emptyset$. Hence, after the removal of any neutral edge form the forest F, the remaining forest is still proper. The observation follows. □

For a coloring C, $Domain(C) = \{v \in V : C(v) \neq \lambda\}$ is the set of nodes to which C assigns a color $c \in \mathcal{C}$.

LEMMA 6.5. *A proper forest (F, C) is convex if and only if $CON((F, C))$ is the empty graph. Hence, a subset $S \subseteq Domain(C)$ is a cover for a forest (F, C) if and only if $CON((F, C|_{V \setminus S}))$ is the empty graph.*

PROOF. Since (F, C) is proper we can assume W.L.O.G that F contains a single tree T.

$\Longrightarrow$: If (T, C) is convex then $carrier(C, d) \cap carrier(C, d') = \emptyset$ for each pair of distinct colors d, d'. Hence every two color carriers are separated by at least one neutral edge, and the removal of all neutral edges splits T into monochromatic subtrees which, in turn, are removed as well.

$\Longleftarrow$: If (T, C) is not convex then there exist colors d, d' such that $carrier(C, d) \cap carrier(C, d') \neq \emptyset$. Hence, $carrier(C, d) \cup carrier(C, d')$ is a subtree with no neutral edges, and hence is contained in some tree created at the first step of the contraction process. This tree has at least two colors, and hence it will not be removed at the second step. Thus $CON((T, C)) \neq \emptyset$. □

6.2 The elimination process

Let (G, C) be a colored graph and let v be a node of color $d \neq \lambda$. Recall that uncoloring of v denotes the replacement of the coloring C by the coloring $C' = C|_{V \setminus \{v\}}$ (i.e., by the coloring C' which is identical to C except that the color of v is "erased").

Let (F, C) be a colored forest and S be any subset of $Domain(C)$, such that $(s_0, \ldots, s_{|S|-1})$ are the elements of S ordered lexicographically. Then, $((F_0, C_0), \ldots, (F_{|S|}, C_{|S|}))$ is the sequence of colored forests defined by the following *elimination process*:

—$(F_0, C_0) \longleftarrow CON((F, C))$.
—for $i = 0$ to $|S| - 1$ do
(1) Let C'_i be the coloring obtained from C_i by uncoloring the node s_i.
(2) $(F_{i+1}, C_{i+1}) \leftarrow CON((F_i, C'_i))$.

Note that $(F_{|S|}, C_{|S|}) = CON((F, C|_{V \setminus S}))$. $\mathcal{F}(S)$ denotes the set of colored trees which are created during the elimination process, that is

$$\mathcal{F}(S) = \{(T, C) : (T, C) \text{ is a colored tree in } (F_i, C_i) \text{ for some } i \in [0, |S|]\},$$

LEMMA 6.6. *If S is a minimal convex cover, then $F_{|S|}$ is the empty forest and $|\mathcal{F}(S)| = |S|$.*

PROOF. Recall that $(F_{|S|}, C_{|S|}) = CON((F, C|_{V \setminus S}))$. Thus by Lemma 6.5, since S is a cover, the elimination process ends with the empty forest, meaning that each colored tree $(T, C) \in \mathcal{F}(S)$ contains a node which is uncolored during the elimination process. If S is a minimal cover, then for each i it holds that $s_i \in V(F_i)$ (otherwise $S \setminus \{s_i\}$ is also a cover, and S is not minimal). Thus the mapping which maps each s_i on the tree X_i which contains s_i at the beginning of iteration i is a bijection from the set of nodes S to the set of colored trees $\mathcal{F}(S)$. □

For a color $d \in C(V(F_i))$, $carrier_i(d)$ denotes the carrier of d in (F_i, C_i); observe that $carrier_0(d) = carrier(d)$. We say that a color d is compound (see Definition 2.5) if it is compound in (F_0, C_0).

The following lemma provides important properties of the sets of edges which are eliminated in each iteration of the elimination process.

LEMMA 6.7. *Let $d = C(s_i)$ for some $i \in \{0, \ldots, |S| - 1\}$, and let $E' \subseteq E$ be the set of edges removed by the contraction of (F_i, C'_i) at iteration i of the elimination process. Then E' is contained in a path in $carrier_i(d)$, and for any $d' \neq d$, $E' \cap carrier_i(d') = \emptyset$.*

PROOF. The contraction of (F_i, C'_i) eliminates the set of edges which become neutral due to the uncoloring of s_i. This means that each such edge belongs to $carrier_i(d)$ but not to $carrier_i(d')$ for any $d' \neq d$. Also, by Observation 6.2, this set is not empty only if s_i is a leaf in $carrier_i(d)$, and these edges lie on a path which starts at s_i and consists of nodes of degree ≤ 2 in $carrier_i(d)$ (see Figure 4). □

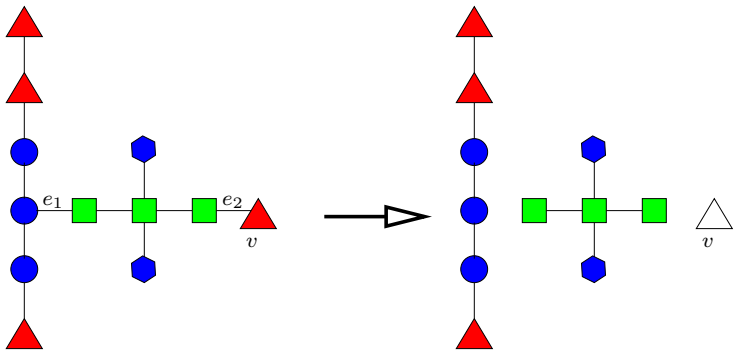

Fig. 4. A red (triangle) node v (left side), which is a leaf in the red carrier, is uncolored. Consequently edges e_1 and e_2 that are in a path starting at v and consists of non-red nodes with degree ≤ 2 in the red carrier become neutral and are eliminated from the tree (right side).

6.3 Lower bound for the optimal cover of a colored forest

In this section we fix some optimal cover S of the colored forest (F_0, C_0) and show that $|S| \geq \alpha/4$. For this, we distinguish between few types of compound colors, according to their relation with the cover S:

Definition 6.8. A compound color d is *active* (w.r.t the cover S) if there is a node $s_i \in S$ so that $C(s_i) = d$. A compound color d is *semi-active* if it is not active but $carrier(d)$ contains a node of S. A compound color d is *passive* if it is neither active nor semi-active, i.e., if $carrier(d)$ does not contain a node of S.

Definition 6.9. A color $d \in C_0(F_0)$ is said to *disappear* in iteration i if $d \in C_i(F_i)$ but $d \notin C_{i+1}(F_{i+1})$. If $d \in C_{i+1}(F_{i+1})$ then d is said to *survive* in iteration i.

Clearly, there are at most $|S|$ compound active colors. Corollary 6.11 below implies that all carriers of semi-active colors are disjoint, and hence that each node in S is contained in at most one such carrier, which implies that there are at most

$|S|$ semi-active compound colors. Claim 6.14 implies that there are at most $2|S|$ passive compound colors. Hence, in total, there are at most $4|S|$ compound colors.

OBSERVATION 6.10. *If d is not active, then for each $i = 0, \ldots, |S| - 1$ it holds that if $carrier_{i+1}(d) \neq \emptyset$, then $carrier_i(d) = carrier_{i+1}(d)$.*

PROOF. By the fact that $C_i^{-1}(d) = C_{i+1}^{-1}(d)$. □

COROLLARY 6.11. *If the compound colors d and d' are not active, then $carrier(d) \cap carrier(d') = \emptyset$. Hence there are at most $|S|$ semi active colors.*

PROOF. By Observation 6.10, $carrier(d) = carrier_{|S|}(d)$ and $carrier(d') = carrier_{|S|}(d')$. The claim follows since S is a cover. The second part is implied by the fact that each carrier of semi active color must contain a node in S, and no two such carriers can contain the same node. □

Next we show that there are at most $2|S|$ passive compound colors.

COROLLARY 6.12. *If d is passive, then $C^{-1}(carrier(d)) \cap \mathcal{C} = \{d\}$.*

PROOF. Assume for contradiction that $carrier(d)$ contains a node v with $C(v) = d' \neq d$ for some $d' \in \mathcal{C}$. Then since $carrier(d) = carrier_i(d)$ for all i, d will survive in all the iterations, contradicting the fact that S is a cover. □

CLAIM 6.13. *A passive color d which is compound in (F_i, C_i) survives in iteration i.*

PROOF. The color d may disappear in iteration i only if $carrier_i(d)$ (which is $carrier(d)$ at iteration i) is eliminated during the contraction of (F_i, C_i'), which means that it is transformed to a (monochromatic) tree by the removal of neutral edges from (F_i, C_i'). Since d is compound in (F_i, C_i), $carrier_i(d)$ either contains nodes of color different from d, or has at least three neighbors (s-degree ≥ 3) in (F_i, C_i). The first case is impossible by Corollary 6.12. In the second case, by Lemma 6.7 the above the set of eliminated edges is included in a path in F_i. Since the intersection of this path and the subtree $carrier_i(d)$ is a path, at most two edges in this path are adjacent to $carrier_i(d)$. Thus the s-degree of $carrier_i(d)$ is reduced by at most two, and hence the s-degree of $carrier_{i+1}(d)$ is at least one, which means that it survives in iteration i. □

By Claim 6.13, each passive compound color d must become pure before it disappears. i.e., there is an i such that d is compound in (F_i, C_i) and it is pure in (F_{i+1}, C_{i+1}), meaning that it becomes pure during iteration i of the elimination process. Thus, the colored forest (F_i, C_i) contains a colored tree (T_1, C_1), such that $s_i \in V(T_1)$, $carrier_i(d) \subseteq T_1$ and d is compound in (T_1, C_1) (see Figure 5). Let C_1' be the coloring resulted by uncoloring s_i in C_1. The contraction of (T_1, C_1') (during iteration i of the elimination process) creates a tree (T, C) in (F_{i+1}, C_{i+1}) so that $carrier_{i+1}(d) \subseteq T$ and d is pure in (T, C). In this scenario, we say that d is *purified at* (T, C). Figure 5 depicts a scenario where two colors are purified at a colored tree (T, C).

By Lemma 6.6, $|\mathcal{F}(S)| = |S|$. Hence in order to prove that there are at most $2|S|$ passive compound colors, it suffices to show that there are at most $2|\mathcal{F}(S)|$ passive compound colors. Since each passive compound color must be purified at some colored tree $(T, C) \in \mathcal{F}(S)$, this will follow from the following claim:

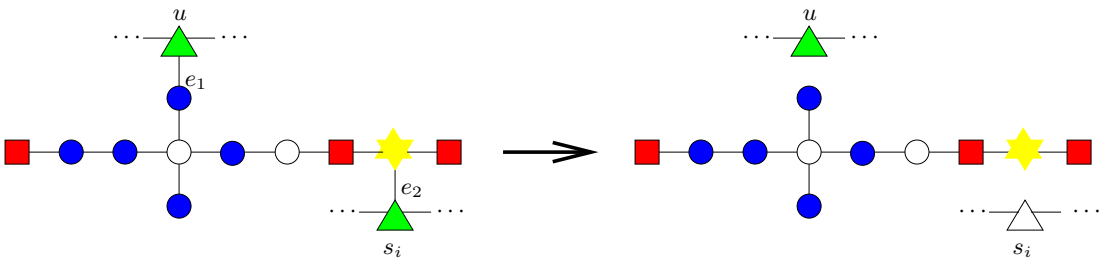

Fig. 5. (T_1, C_1) is shown on the left. On the right, (T, C) is created by the uncoloring of node s_i and the subsequent removal of edges e_1 and e_2. Two colors (the blue circles and pink stars) are purified in (T, C).

CLAIM 6.14. *For every colored tree $(T, C) \in \mathcal{F}(S)$, at most two passive colors are purified at (T, C).*

PROOF. Let d be a passive color which is purified at (T, C), and let (T_1, C_1) and C_1' be the corresponding tree and colorings described above. By Corollary 6.12, $carrier_i(d)$ contains no node of color different from d. Hence d can be purified at (T, C) only if the degree of $carrier_i(d)$ is reduced by at least one during the contraction of (T_1, C_1').

Let e be an edge adjacent to T which is removed by the contraction of (T_1, C_1'). Since, by Corollary 6.11, carriers of passive colors are disjoint, e can be adjacent to at most one carrier of a passive color in (T, C). Hence the removal of e can reduce the s-degree of at most one carrier of such passive color. By Lemma 6.7 at most two edges adjacent to T are removed, and hence (T, C) contains at most two carriers of passive colors whose degrees are reduced in the contraction of (T_1, C_1'). We conclude that at most two passive compound colors are purified in (T, C). Figure 5 depict a scenario where exactly two colors are purified.

□

The above leads to the following theorem:

THEOREM 6.15. $\alpha \leq 4|S|$.

PROOF. There are at most $|S|$ active compound colors, at most $|S|$ semi-active compound colors, and at most $2|S|$ passive compound colors. The theorem follows since each compound color must be either active, semi-active, or passive. □

6.4 Extension to Galled Networks

In this section we extend our results to handle galled networks. The extension proceeds along the same lines of the proof for trees. The discussion that follows refer to some arbitrary colored galled network (GN, C). We start with extending some of the previous definitions.

Definition 6.16. A *cycle pair* is a pair of cycle edges from the same cycle.

OBSERVATION 6.17. *Any removal of either a tree edge or a cycle pair partitions the galled network into exactly two nonempty connected components, and vice versa.*

Recall the convexity property of colorings of galled networks: a colored galled network (GN, C) is convex if and only if there exists a set $\{CA_d : d \in C(V)\}$ such that for each d, CA_d is a d-carrier, and for each pair of colors $\{d, d'\}$, it holds that $CA_d \cap CA_{d'} = \emptyset$.

For a galled network G, $B(G)$ is the graph obtained from G by contracting each cycle in G to a single node - i.e., $B(G)$ is the biconnected graph of G [Golumbic 1980]. Clearly, $B(G)$ is a tree. We can look at $B(.)$ as a function that maps every tree node or tree edge in G to itself and every cycle node or cycle edge to the node in $B(G)$ representing its bicomponent.

Definition 6.18. The graph induced by a set of nodes V' in G is called a *gpath* if the graph induced by the set $B(V')$ is a path in $B(G)$.

Note that if V' is contained in a single cycle, then $B(V')$ contains a single node.

Observation 6.19. *Let GN be a galled network and $u, v \in V(GN)$. Then all the simple paths connecting u and v are contained in a single gpath in GN.*

Proof. Since $B(GN)$ is a tree, there is a single path in $B(GN)$ connecting the corresponding bicomponents of u and v (this path could be a single node). By Definition 6.18 the observation follows. □

Given a set of paths Π, a path $p \in \Pi$ is *minimal* if it does not contain any other path $p' \in \Pi$.

The following observation is used later in this section.

Observation 6.20. *Let V' be a subset of V s.t. $G(V')$ is connected, and let $v \in V \setminus V'$. Then the union of minimal paths from v to V' is a gpath.*

Proof. Let $V'' \subseteq V'$ be the set of nodes in V' contained in minimal paths from v to V'. Then either V'' consists of two nodes on the same cycle, or V'' is a single node. In the first case, if v is in that same cycle the observation follows by the definition of gpaths. Otherwise, either $|V''| = 2$ and v is not in the cycle of V'', or $|V''| = 1$. These later cases are covered by Observation 6.19 (see Figure 6). □

We define a galled forest to be a disjoint union of galled networks and we say that a galled forest is proper similar to the case of regular forests. The notion of a neutral edge is extended to include also a *neutral pair* in the natural way (i.e. a cycle pair whose removal partitions the galled network into two galled networks with disjoint colors). We now redefine the contraction process for a galled forest GF:

Definition 6.21. The *contraction* of (GF, C), denoted by $CON((GF, C))$, is a proper galled forest obtained by:

—Repeatedly removing from GF neutral edges and neutral pairs[1], until no neutral edges and neutral pairs remain, and then

—removing from the resulted galled forest all colorless and monochromatic galled networks (i.e. galled networks of at most one color)

[1]Note that the removal of cycle pair edges turns the remaining edges in the cycle to tree edges

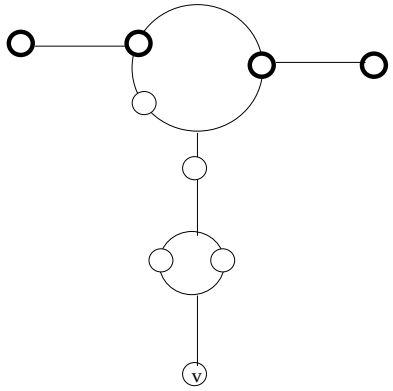

Fig. 6. V' are the bold nodes on the top. V'' consists of two nodes on the same cycle. By Observation 6.19 the paths from V'' are contained in a single gpath.

The proofs of the following observation and lemma are identical to the tree case.

OBSERVATION 6.22. *If (GF, C) is proper, then $CON((GF, C))$ is also proper.*

LEMMA 6.23. *A proper galled forest (GF, C) is convex if and only if $CON((GF, C))$ is the empty graph. Hence, a subset $S \subseteq Domain(C)$ is a cover for (GF, C) if and only if $CON((GF, C|_{V \setminus S}))$ is the empty graph.*

COROLLARY 6.24. *The decision problem whether a colored galled network (GN, C) is convex can be answered in polynomial time.*

PROOF. By Lemma 6.23 we only need to see if $CON((GN, C)) = \emptyset$, which can be done by contracting GN. The first stage of the contraction is done by repeatedly finding and removing neutral cycle pairs and edges, and the second by removing colorless or monochromatic galled networks. Both stages can clearly be done in polynomial time. □

The elimination process in a galled forest is identical to that on a regular forest. We repeat it for the sake of readability:

Let (GF, C) be a proper colored galled forest. For a set of nodes S, let $((GF_0, C_0), \ldots, (GF_{|S|}, C_{|S|}))$ be defined by:

—$(GF_0, C_0) \longleftarrow CON((GF, C))$.

—for $i = 0$ to $|S| - 1$ do

(1) Let C'_i be the coloring obtained from C_i by uncoloring the node s_i.

(2) $(GF_{i+1}, C_{i+1}) \leftarrow CON((GF_i, C'_i))$.

We also define $\mathcal{GF}(S)$ as the set of colored galled networks which are created during the elimination process, identically to the tree case.

LEMMA 6.25. *Let $d = C(s_i)$ for some $i \in \{0, \ldots, |S| - 1\}$, and let $E' \subseteq E$ be the set of edges removed by the contraction of (GF_i, C'_i) at iteration i of the elimination process. Then E' is contained in a gpath in GF_i.*

The proof is followed directly from Observation 6.20.

Here again we fix a set S that is an optimal cover and give a lower bound on its size in terms of the number of compound colors.

We keep the definition of an active color as is, i.e. A color d is *active* (w.r.t the cover S) if there is a node $s_i \in S$ such that $C(s_i) = d$. We adjust the definitions of semi-active and passive colors to galled forests as follows:

Let d be a color which is not active. Then the set of its carriers is fixed throughout the elimination process (until d disappears). We select for each such color d one of these carriers as follows: Let C' be any extension of $C|_{V \setminus S}$ to a total convex recoloring of the input network GN. Then for each inactive color d, $\widehat{carrier}(d) = C'^{-1}(d)$. That is: $\widehat{carrier}(d)$ is the carrier of d in the total convex coloring C'. An inactive color d is *semi-active* if $\widehat{carrier}(d)$ contains a node in S, and it is *passive* otherwise. With these definitions, the proofs of the analogues of Observation 6.10 and Corollaries 6.11 and 6.12 are similar to the tree case, and are omitted. These proofs imply that the carriers $\widehat{carrier}(d)$ of inactive colors d are disjoint, and hence that there are at most $|S|$ semi-active colors. They also imply that for a passive color d, $\widehat{carrier}(d)$ contains no node of color $d' \neq d$.

CLAIM 6.26. *A passive color d which is compound in (GF_i, C_i) survives in iteration i.*

PROOF. The color d may disappear in iteration i only if $\widehat{carrier}(d)$ is separated from all other colors by the removal of neutral edges and neutral pairs from (GF_i, C'_i). Since d is compound in (GF_i, C_i), $\widehat{carrier}(d)$ may either contain a node of other color or have an s-degree at least three. The former case is impossible since d is passive, so the s-degree of $\widehat{carrier}(d)$ is at least three. By Lemma 6.25 the set of edges removed is contained in a gpath. Since each tree edge or cycle pair adjacent to $\widehat{carrier}(d)$ contributes one to the s-degree, the removal of that gpath decreases the s-degree of $\widehat{carrier}(d)$ by at most two. Hence after the contraction d is not separated from all other colors, meaning that it survives the iteration.

□

Recall that a color d in a graph is pure if there is a d-carrier with no other d'-nodes and with s-degree two.

By Claim 6.26, for each passive compound color d there is an i such that d is compound in (GF_i, C_i) and it becomes pure in (GF_{i+1}, C_{i+1}) as follows: The colored galled forest (GF_i, C_i) contains a colored galled network (GN_1, C_1), such that $s_i \in V(GN_1)$ and d is compound in (GN_1, C_1). Let C'_1 be the coloring resulted by uncoloring s_i in C_1. The contraction of (GN_1, C'_1) (during iteration i of the elimination process) creates a galled network (GN, C) in (GF_{i+1}, C_{i+1}) such d is pure in (GN, C). In this scenario, we say that d is *purified at* (GN, C).

The following claim is similar to Claim 6.14, and it shows that at most two compound colors are purified at each galled network:

CLAIM 6.27. *For every colored galled network $(GN, C) \in \mathcal{GF}(S)$, at most two passive colors are purified at (GN, C).*

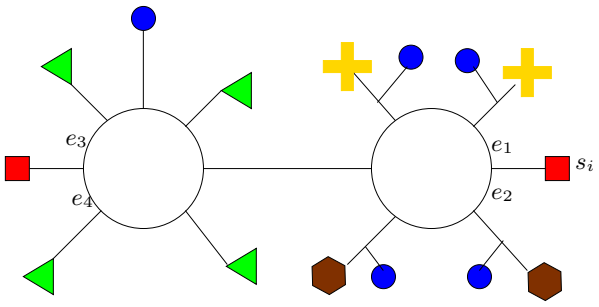

Fig. 7. Uncoloring node s_i in (GN_1, C_1) causes the removal of the two neutral pairs e_1 and e_2 on the right and e_3 and e_4 on the left. It can be seen that the color (green triangle) at the left side of the new galled network obtained is purified whereas the two compound passive colors at the right (the yellow cross and brown hexagon) remain with s-degree 3.

PROOF. Let d be a passive color which is purified at (GN, C), and let (GN_1, C_1) and C_1' be the corresponding galled network and colorings described above. As noted above, $\widehat{carrier}(d)$ contains no node of color different from d. Hence the s-degree of $\widehat{carrier}(d)$ is at least three, and d can be purified at (GN, C) only if the s-degree of $\widehat{carrier}(d)$ is reduced by at least one during the contraction of (GN_1, C_1').

The case of galled networks is more involved than trees since the contraction process removes not only neutral edges but also neutral pairs. Therefore we need the following auxiliary observation to complete the proof:

OBSERVATION 6.28. *Let $CP = \{e_1, e_2\}$ be a neutral cycle pair, separating the galled network into two galled networks GN_1 and GN_2. Then there is at most one (passive) color d in GN_1 (GN_2), such that the removal of CP decreases the s-degree of $\widehat{carrier}(d)$.*

PROOF. We first note that the s-degree of a carrier is decreased only if it is adjacent to a removed edge. We divide the proof into two cases:

Case 1: The two edges are adjacent to the same carrier. Then by the previous argument only that carrier can be affected.

Case 2: Each edge of the pair is adjacent to a different carrier. Then, it can be noted that the s-degree of both carriers is not affected.

□

Figure 7 shows this pictorially.

By Lemma 6.25, all the removed edges are contained in a gpath. Therefore GN is created either by removing a neutral edge or a cycle pair at both sides. If GN was created by removing a cycle pair, then by Observation 6.28, the s-degree of at most one carrier $\widehat{carrier}(d)$ for a pure color d in GN can be reduced. The same holds for removing a neutral tree-edge as in the tree case.
We conclude that at most two passive compound colors are purified in (GN, C).

The above leads to the following theorem:

THEOREM 6.29. $\alpha \leq 4|S|$.

PROOF. Same as for Theorem 6.15 □

7. CONCLUDING REMARKS AND FURTHER RESEARCH DIRECTIONS

In this work we extended the previous works on convex recoloring to handle partial coloring and galled networks. We first showed that the decision problem of whether a coloring of general graph is convex is NP-hard. Next we defined the notion of compound colors in a partial coloring of galled networks and provided an algorithm for optimal convex recoloring of galled networks that runs in time polynomial in the size of the graph and exponential in the number of compound colors. Finally, by using the notions of gpaths and cycle pairs, we showed that the size of an optimal cover in a galled network is bounded from below by a linear function of the number of compound colors.
Few further research directions come to mind:

—Is there a simple generalization of the algorithm presented here to more involved cases of networks?

— [Bar-Yehuda et al. 2005; Moran and Snir 2007] studied the approximability of convex recoloring for trees and paths. It is of interest whether these technique can be employed on the graphs studied here.

ACKNOWLEDGMENTS

The second author would like to thank Satish Rao for helpful discussions. We would also like to thanks the referees for very helpful comments.

REFERENCES

BAR-YEHUDA, R., FELDMAN, I., AND RAWITZ, D. 2005. Improved approximation algorithm for convex recoloring of trees. In *WAOA*.

BODLAENDER, H. L., FELLOWS, M. R., LANGSTON, M. A., RAGAN, M. A., ROSAMOND, F. A., AND WEYER, M. 2006. Kernelization for convex recoloring. In *ACiD*. 23–35.

BODLAENDER, H. L., FELLOWS, M. R., LANGSTON, M. A., RAGAN, M. A., ROSAMOND, F. A., AND WEYER, M. 2007. Quadratic kernelization for convex recoloring of trees. In *COCOON*. LNCS, vol. 4598. Springer, 86–96.

CHEN, X., HU, X., AND SHUAI, T. 2006. Inapproximability and approximability of maximal tree routing and coloring. *Journal of Combinatorial Optimization 11,* 2, 219–229.

CHOR, B., FELLOWS, M. R., RAGAN, M. A., RAZGON, I., ROSAMOND, F. A., AND SNIR, S. 2007. Connected coloring completion for general graphs: Algorithms and complexity. In *COCOON*. LNCS, vol. 4598. Springer, 75–85.

CORMEN, T., LEISERSON, C., RIVEST, R., AND STEIN, C. 2001. *Introduction to algorithms*. MIT Press.

DOWNEY, R. G. AND FELLOWS, M. R. 1999. *Parameterized Complexity*. Springer.

GAREY, M. AND JOHNSON, D. 1979. *Computers and Intractability – A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York.

GOLUMBIC, M. C. 1980. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press.

KUHN, H. 1955. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 83–97.

MORAN, S. AND SNIR, S. 2007. Efficient approximation of convex recolorings. *Journal of Computer and System Sciences 73,* 7, 1078–1089. preliminary version appeared in APPROX-RANDOM, 2005, pp. 192-208.

Moran, S. and Snir, S. 2008. Convex recolorings of strings and trees: Definitions, hardness results and algorithms. *Journal of Computer and System Sciences 74,* 5, 850–869. preliminary version appeared in WADS, 2005, pp. 218-232.

Ponta, O. 2007. The fixed-parameter approach to the convex recoloring problem. Ph.D. thesis, Mathematisches Institut, Ruprecht-Karls-Universität Heidelberg.

Razgon, I. 2007. A $2^{O(k)}$poly(n) algorithm for the parameterized convex recoloring problem. *Information Processing Letters 104,* 2, 53–58.

Sankoff, D. 1975. Minimal mutation trees of sequences. *SIAM Journal of Applied Mathematics 28,* 1, 35–42.

Semple, C. and Steel, M. 2003. *Phylogenetics*. Oxford University Press.

Weisstein, E. W. Bell number. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/BellNumber.html.

...