Quadratic Kernelization for Convex Recoloring of Trees*

Hans L. Bodlaender[1], Michael R. Fellows[23], Michael A. Langston[4], Mark A. Ragan[35], Frances A. Rosamond[23], and Mark Weyer[6]

[1] Department of Information and Computing Sciences, Utrecht University, Utrecht, the Netherlands hansb@cs.uu.nl

[2] Parameterized Complexity Research Unit, Office of the DVC(Research), University of Newcastle, Callaghan NSW 2308, Australia, {michael.fellows,frances.rosamond}@newcastle.edu.au

[3] Australian Research Council Centre in Bioinformatics

[4] Department of Computer Science, University of Tennessee, Knoxville TN 37996-3450 and Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831-6164 U.S.A. langston@cs.utk.edu

[5] Institute for Molecular Bioscience, University of Queensland, Brisbane, QLD 4072 Australia, m.ragan@imb.uq.edu.au

[6] Institut für Informatik, Humboldt-Universität zu Berlin, Berlin, Germany mark.weyer@informatik.hu-berlin.de

Abstract. The Convex Recoloring (CR) problem measures how far a tree of characters differs from exhibiting a so-called "perfect phylogeny". For input consisting of a vertex-colored tree T, the problem is to determine whether recoloring at most k vertices can achieve a convex coloring, meaning by this a coloring where each color class induces a connected subtree. The problem was introduced by Moran and Snir, who showed that CR is NP-hard, and described a search-tree based FPT algorithm with a running time of $O(k(k/\log k)^k n^4)$. The Moran and Snir result did not provide any nontrivial kernelization. Subsequently, a kernelization with a large polynomial bound was established. Here we give the strongest FPT results to date on this problem: (1) We show that in polynomial time, a problem kernel of size $O(k^2)$ can be obtained, and (2) We prove that the problem can be solved in linear time for fixed k. The technique used to establish the second result appears to be of general interest and applicability for bounded treewidth problems.

Topics: Algorithms and Complexity

* This research has been supported by the Australian Research Council Centre in Bioinformatics, by the U.S. National Science Foundation under grant CCR–0311500, by the U.S. National Institutes of Health under grants 1-P01-DA-015027-01, 5-U01-AA-013512-02 and 1-R01-MH-074460-01, by the U.S. Department of Energy under the EPSCoR Laboratory Partnership Program and by the European Commission under the Sixth Framework Programme. The second and fifth authors have been supported by a Fellowship to the Institute of Advanced Studies at Durham University, and hosted by a William Best Fellowship to Grey College during the preparation of the paper.

1 Introduction

The historically first and most common definition of *fixed-parameter tractability* for parameterized problems is solvability in time $O(f(k)n^c)$, where n is the input size, k is the parameter, and c is a constant independent of n and k. Background and motivation for the general subject of parameterized complexity and algorithmics can be found in the books [9, 10, 14]. This basic view of the subject is by now well-known.

Less well-known is the point of view that puts an emphasis on FPT kernelization. Kernelization is central to FPT as shown by the lemma:

Lemma 1. *A parameterized problem Π is in FPT if and only if there is a transformation from Π to itself, and a function g, that reduces an instance (x, k) to (x', k') such that:*

1. *the transformation runs in time polynomial in $|(x, k)|$,*
2. *(x, k) is a yes-instance of Π if and only if (x', k') is a yes-instance of Π,*
3. *$k' \leq k$, and*
4. *$|(x', k')| \leq g(k)$.*

The lemma is trivial, but codifies a shift of perspective. To see how this works, consider the Moran and Snir FPT result for Convex Recoloring, with the running time of $O^*(k/\log k)^k)$. When $n > (k/\log k)^k$, then the Moran and Snir algorithm runs in polynomial time, in fact, in time $O(n^5)$. (So we can run the algorithm and determine the answer, and "transform" the input to either a canonical small NO instance or a canonical small YES instance accordingly.) If $n \leq (k/\log k)^k$ then we simply do nothing; we declare that the input is "already" kernelized to the bound $g(k) = (k/\log k)^k$. If a parameterized problem is FPT, that is, solvable in time $f(k)n^c$, then the lemma above gives us the trivial P-time kernelization bound of $g(k) = f(k)$. Normally for FPT problems, $f(k)$ is some exponential function of k, so the subclasses of FPT

$$Lin(k) \subseteq Poly(k) \subseteq FPT$$

of parameterized problems that admit P-time kernelization to kernels of size bounded by *linear* or *polynomial* functions of k would seem to be severe restrictions of FPT.

It is surprising that so many parameterized problems in FPT belong to these much more restrictive subclasses. The connection to heuristics goes like this: *FPT kernelization* is basically a systematic approach to *polynomial-time* preprocessing. Preprocessing plays a powerful role in practical approaches to solving hard problems. For any parameterized problem in FPT there are now essentially two different algorithm design competitions with independent payoffs:
(1) to find an FPT algorithm with the best possible exponential cost $f(k)$, and
(2) to find the best possible polynomial-time kernelization for the problem.

A classic result on FPT kernelization is the Vertex Cover $2k$ kernelization due to Nemhauser and Trotter (see [14]). The linear kernelization for Planar

Dominating Set is definitely nontrivial [1]. The question of whether the Feedback Vertex Set (FVS) problem for undirected graphs has a $Poly(k)$ kernelization was a noted open problem in parameterized algorithmics, only recently solved. The first $Poly(k)$ kernelization for FVS gave a bound of $g(k) = O(k^{11})$ [6]. Improved in stages, the best kernelization bound is now $O(k^3)$ [4].

We prove here a polynomial time kernelization for the Convex Recoloring problem, to a reduced instance that has $O(k^2)$ vertices. The basic problem is defined as follows.

Convex Recoloring
Instance: A tree $F = (V, E)$, a set of colors $\mathcal{C}$, a coloring function $\Gamma : V \to \mathcal{C}$, and a positive integer k.
Parameter: k
Question: Is it possible to modify Γ by changing the color of at most k vertices, so that the modified coloring Γ' is *convex* (meaning that each color class induces a single monochromatic subtree)?

The Convex Recoloring problem is of interest in the context of maximum parsimony approaches to evaluating phylogenetic trees [12, 11]. Some further applications in bioinformatics are also described in [12]; see also [13].

2 Preliminaries

Our kernelization algorithm is shown for the following generalized problem.

Annotated Convex Recoloring
Instance: A forest $F = (V, E)$ where the vertex set V is partitioned into two types of vertices, $V = V_0 \cup V_1$, a set of colors $\mathcal{C}$, a coloring function $\Gamma : V \to \mathcal{C}$, and a positive integer k.
Parameter: k
Question: Is it possible to modify Γ by changing the color of at most k vertices in V_1, so that the modified coloring Γ' is *convex*?

A *block* in a colored forest is a maximal set of vertices that induces a monochromatic subtree. For a color $c \in \mathcal{C}$, $\beta(\Gamma, c)$ denotes the number of blocks of color c with respect to the coloring function Γ. A color c is termed a *bad color* if $\beta(\Gamma, c) > 1$, and c is termed a *good color* if $\beta(\Gamma, c) = 1$.

A recoloring Γ' of coloring Γ is *expanding*, if for each block of Γ', there is at least one vertex in the block that keeps its color, i.e., that has $\Gamma'(v) = \Gamma(v)$. Moran and Snir [12] have shown that there always exists an optimal expanding convex recoloring. It is easy to see that this also holds for the variant where we allow annotations and a forest, as above.

As in [3], we define for each $v \in V$ a set of colors $S(v)$: $c \in S(v)$, if and only if there are two distinct vertices w, x, $w \neq v$, $x \neq v$, such that w and x have color c (in Γ), and v is on the path from w to x.

We introduce another special form of convex recoloring, called *normalized*. For each vertex $v \in V$, we add a new color c_v. We denote $\mathcal{C}' = \mathcal{C} \cup \{c_v \mid v \in V\}$. A recoloring $\Gamma' : V \to \mathcal{C}'$ of coloring $\Gamma : V \to \mathcal{C}$ is *normalized*, if for each $v \in V$: $\Gamma'(v) \in \{\Gamma(v), c_v\} \cup S(v)$.

Lemma 2. *There is an optimal convex recoloring that is normalized.*

Proof. Take an arbitrary optimal convex recoloring Γ'. Define Γ'' as follows. For each $v \in V$, if $\Gamma'(v) \in \{\Gamma(v)\} \cup S(v)$, then set $\Gamma''(v) = \Gamma'(v)$. Otherwise, set $\Gamma''(v) = c_v$. One can show that $\Gamma''(v)$ is convex. Clearly, it is normalized, and recolors at most as many vertices as $\Gamma'(v)$. □

3 Kernelizing to a Linear Number of Vertices Per Color

In this section, we summarize a set of rules from [6] that ensure that for each color, there are at most $O(k)$ vertices with that color. There are two trivial rules: if F is an empty forest and $k \geq 0$, then we return YES; if $k < 0$, then we return NO if and only if the given coloring is not already convex.

Rule 1 *Suppose there are $\alpha \geq 2k + 3$ vertices with color c. Let v be a vertex, such that each connected component of $F - v$ contains at most $\frac{\alpha}{2}$ vertices of color c. Assume that $v \notin V_0$, or $\Gamma(v) \neq c$. Now*

- *If v has a color, different from c, and $v \in V_0$, then return NO.*
- *If v has a color, different from c, and $v \notin V_0$, then set $\Gamma(v) = c$ and decrease k by one.*
- *Fix the color of v: put $v \in V_0$.*

Rule 2 *Suppose $v \in V_0$ has color c. Suppose one of the connected components of F contains at least $k + 1$ vertices with color c, but does not contain v. Then return NO.*

Rule 3 *Let v be a vertex of color c. Suppose $F - v$ has a component with vertex set W that contains at least $k + 1$ vertices with color c, that contains a neighbor w of v. Assume that $w \notin V_0$ or $\Gamma(w) \neq c$. Then*

- *If w has a color, different from c, and $w \in V_0$, then return NO.*
- *If w has a color, different from c, and $w \notin V_0$, then give w color c, and decrease k by one.*
- *Fix the color of w: put $w \in V_0$.*

Rule 4 *If there is a tree T in the colored forest that contains two distinct vertices u and v, both of which have fixed color c, then modify T by contracting the path between u and v (resulting in a single vertex of fixed color c). If r denotes the number of vertices of this path that do not have color c, then $k' = k - r$.*

Rule 5 *If there are two distinct trees in the colored forest that both contain a vertex of fixed color c, then return NO.*

Let $v \in V_0$ have color c. A connected component of $F - v$ with vertex set W is said to be *irrelevant*, if both of the following two properties hold: (1) The vertices with color c in W form a connected set that contains a neighbor of v, and (2) For each color $c' \neq c$, if there are vertices with color c' in W, then c' is not bad. If a component is not irrelevant, it is said to be *relevant*.

Rule 6 *Let $v \in V_0$. Let W be the vertex set of an irrelevant connected component of $F - v$. Then remove W, and all edges incident to vertices in W, from the tree.*

Rule 7 *Suppose $F - v$ contains at least $2k+1$ components, and each component of $F - v$ is relevant. Then return NO.*

Rule 8 *Let $v \in V_0$ be a vertex with fixed color c. Let W be the vertex set of a relevant component of $F - v$ that contains vertices with color c. Suppose there are vertices with color c, not in $\{v\} \cup W$. Then*

- *Take a new color c', not yet used, and add it to $\mathcal{C}$.*
- *Take a new vertex v', and add it to F.*
- *Set $v' \in V_0$. Color v' with c'.*
- *For all $w \in W$ with color c, give w color c'. (Note that k is* not *changed.)*
- *For each edge $\{v, w\}$ with $w \in W$: remove this edge from F and add the edge $\{v', w\}$.*

Theorem 1. *An instance that is reduced with respect to the above Rules has at most $2k + 2$ vertices per color.*

4 Kernelizing to a Quadratic Number of Vertices

In this section, we work in a series of steps towards a kernel with $O(k^2)$ vertices. A number of new structural notions are introduced.

4.1 Bad colors

Rule 9 *If there are more than $2k$ bad colors, then return NO.*

The colors that are not bad are distinguished into three types: gluing, stitching, and irrelevant.

4.2 Gluing colors

A vertex v is *gluing* for color c, if it is adjacent to two distinct vertices x and y, both of color c, but the color of v is unequal to c. Note that if v is gluing for color c, then v separates (at least) two blocks of color c: x and y belong to different blocks. When we recolor v with color c, then these blocks become one block (are 'glued' together). A vertex v is *gluing*, if v is gluing for some color, and if the color of v is not bad. A color c is *gluing*, if there is a gluing vertex with

color c, and c is not bad. (Note that the vertex will be gluing for some other color $c' \neq c$.)

For a color c, let W_c be the set of vertices that have color c or are gluing for color c. A *bunch* of vertices of color c is a maximal connected set of vertices in W_c, i.e., a maximal connected set of vertices that have either color c or are gluing for color c. The following lemma is not difficult, and establishes the soundness of the next Rule.

Lemma 3. *Suppose W is a bunch of color c that contains ℓ vertices that are gluing for color c. Then in any convex recoloring, at least ℓ vertices in W are recolored, and for each, either the old, or the new color is c.*

Rule 10 *If there are more than $2k$ vertices that are gluing, then return NO.*

As a result, there are $O(k)$ colors that are gluing, and thus $O(k^2)$ vertices that have a gluing color. We next bound the number of bunches.

Lemma 4. *Suppose there are ℓ bunches with color c. Then the number of recolored vertices whose old or new color equals c, is at least $\ell - 1$.*

Proof. Omitted due to space limitations. □

Rule 11 *Suppose that for each bad color c, there are ℓ_c bunches. If the sum over all bad colors c of $\ell_c - 1$ is at least $2k + 1$, then return NO.*

4.3 Stitching colors

We now define the notion of stitching vertices. To arrive at an $O(k^2)$ kernel, we have to define this notion with some care.

We first define the *cost* of a path between two vertices with the same color that belong to different bunches. Let v and w be two vertices with color c, in the same subtree of F, but in different bunches of color c. The *cost* of the path from v to w is the sum of the number of vertices with color unequal to c on this path, and the sum over all colors $c' \neq c$ such that

- c' is not bad and c' is not gluing, and
- there is at least one vertex with color c' on the path from v to w in T

of the following term: consider the blocks with color c' in the forest, obtained by removing the path from v to w from T. If one of these blocks contains a vertex in V_0 (i.e., it has fixed color c'), then sum the sizes of all other blocks. Otherwise, sum the sizes of the all blocks except one block with the largest number of vertices.

A *stitch* of color c is a path between two vertices v, w, both of color c, such that: (1) v and w are in different bunches, (2) all vertices except v and w have a color, different from c, and do not belong to V_0, and (3) the cost of the path is at most k. A vertex v is *stitching* for color c, if: (1) the color of v is different from c, (2) v is not gluing for color c, and (3) v is on a stitch of color c. A vertex

is *stitching*, if it is stitching for at least one color. (Note that this vertex will be stitching for a bad color.) A color c is *stitching* if there is at least one vertex with color c that is stitching, and it is not bad or gluing. A color is *irrelevant*, if it is neither bad, gluing, or stitching. Summarizing, we have the following types of vertices: (a) vertices with a bad color, (b) vertices with a gluing color, (c) vertices with a stitching color that belong to a stitch, (d) vertices with a stitching color, that do not belong to a stitch — these will be partitioned into pieces in Section 4.4, and (e) vertices with an irrelevant color.

Proofs of the next two lemmas can be found in the full paper, and establish the soundness of the next two rules.

Lemma 5. *Suppose there is a convex recoloring with at most k recolored vertices. Then there is an optimal convex recoloring such that for each bad color c, all vertices that receive color c have color c in the original coloring or are gluing for c or are stitching for c.*

Lemma 6. *Suppose there is a convex recoloring with at most k recolored vertices. Then there is an optimal convex recoloring such that no vertex with an irrelevant color is recolored.*

Rule 12 *Let $v \in V_1$ be a vertex with an irrelevant color. Put v into V_0 (i.e., fix the color of v.)*

Rule 13 *Let $v \in V_1$ be a vertex with a stitching color. Suppose v is not vulnerable. Then put v into V_0.*

The following rule, in combination with earlier rules, helps us to get a reduced instance without any vertex with an irrelevant color.

Rule 14 *Let $v \in V_0$ be the only vertex with some color c. Then remove v and its incident edges.*

Soundness is easy to see: the same recolorings in the graph with and without v are convex. The combination of Rules 4, 12, and 14 causes the deletion of all vertices with an irrelevant color: all such vertices first get a fixed color, then all vertices with the same irrelevant color are contracted to one vertex, and then this vertex is deleted. So, we can also use instead the following rule.

Rule 15 *Suppose c is an irrelevant color. Then remove all vertices with color c.*

4.4 Pieces of a stitching color

For a kernel of size $O(k^2)$, we still can have too many vertices with a stitching color. In order to reduce the number of such vertices we introduce the concept of a *piece of color c*. Consider a stitching color c, and consider the subforest of the forest, induced by the vertices with color c. If we remove from this subtree all vertices that are on a stitch, then the resulting components are the pieces, i.e., a

piece of color c is a maximal subtree of vertices with color c that do not belong to a stitch. Assume that we have exhaustively applied all of the rules described so far, and therefore we do not have vertices with an irrelevant color. The next lemma, given here without proof, shows that the next Rule is sound.

Lemma 7. *Suppose W is the vertex set of a piece of stitching color c. Suppose there is a vertex $v \in W \cap V_0$. Then if there is a convex recoloring with at most k recolored vertices, then there is a convex recoloring that does not recolor any vertex in W.*

Rule 16 *Let W be a vertex set of a piece of stitching color c, that contains at least one vertex in V_0. Then put all of the vertices of W into V_0.*

As a result of this rule and Rule 4, a piece containing a vertex with a fixed color will contain only one vertex. We omit the soundness proof for the following rule from this version. If there is a large piece, found by Rule 17, then it will be contracted to a single vertex by Rule 4.

Rule 17 *Suppose c is a stitching color, and there are α vertices with color c. Suppose there is no vertex in V_0 with color c. If W is the vertex set of a piece of color c, and $|W| > \alpha/2$, then put all of the vertices of W into V_0.*

4.5 Kernel size

We now *tag* some vertices that have a stitching color. A tag is labeled with a bad color. Basically, when we have a stitch for a bad color c, we tag the vertices that count for the cost of the stitch with color c. Tagging is done as follows. For each stitch for bad color c, and for each stitching color c' with a vertex on the stitch with color c', consider the blocks of color c' obtained by removing the vertices on the stitch. If there is no vertex with color c' in V_0, then take a block with vertex set W such that the number of vertices in this piece is at least as large as the number of vertices in any other piece. Tag all vertices in $Q - W$ with color c. If there is a vertex v with color c' in V_0, then tag all vertices with color c', except those that are in the same block as v.

Comparing the tagging procedure with the definition of the cost of a stitch, we directly note that the number of vertices that is tagged equals the cost of the stitch, i.e., for each stitch of bad color c, we tag at most k vertices with c.

We now want to count the number of vertices with a stitching color. To do so, we first count the number of vertices with a stitching color that are tagged. To do this, we consider a bad color c, and count the number of vertices, with a stitching color, that are tagged with c.

The following three lemmas are proved in the full paper.

Lemma 8. *Let c be a bad color with ℓ_c bunches. A reduced instance has at most $2k(\ell_c - 1)$ vertices that are tagged with c.*

Lemma 9. *Let c be a stitching color. There is at most one piece of color c that contains a vertex that is not tagged with a bad color.*

Lemma 10. *In a reduced instance, there are at most $8k^2$ vertices with a stitching color.*

Theorem 2. *In time polynomial in n and k, a kernel can be found with $O(k^2)$ vertices.*

Proof. With standard techniques, one can observe that all rules can be carried out in time polynomial in n and k.

In the reduced instance, there are at most $2k$ bad colors, and at most $2k$ gluing colors. For each of these colors, there are at most $2k + 2$ vertices with that color. So, in total at most $4k^2 + 4k$ vertices have a bad or gluing color. There are at most $8k^2$ vertices with a stitching color, and no vertices with an irrelevant color, so in total we have at most $12k^2 + 4k$ vertices. □

5 Linear Time FPT with Treewidth and MSOL

We describe an algorithm that solves the Convex Recoloring problem in $O(f(k) \cdot n)$ time. There are four main ingredients of the algorithm: (1) the construction of an auxiliary graph, the *vertex-color* graph, (2) the observation that for yes-instances of the Convex Recoloring problem this graph have bounded treewidth, (3) a formulation of the Convex Recoloring problem for fixed k in Monadic Second Order Logic, and (4) the use of a famous result of Courcelle [8], see also [2, 5]. For ease of presentation, we assume that there are no vertices with a fixed color, and that the input is a tree. The "trick" of using such an auxiliary graph seems to be widely applicable. For another example of this new technique, see [7].

Suppose we have a tree $T = (V, E)$, and a coloring $\Gamma : V \to \mathcal{C}$. The *vertex-color graph* has as vertex set $V \cup \mathcal{C}$, i.e., we have a vertex for each vertex in T, and a vertex for each color. The edge set of the vertex-color graph is $E \cup \{\{v, c\} \mid \Gamma(v) = c\}$, i.e., there are two types of edges: the edges of the tree T, and an edge between a vertex of T and the vertex representing its color. Recall that $S(v)$ denotes the set of colors c for which there are vertices w and x, distinct from v, with v on the path from w to x in T, and $\Gamma(w) = \Gamma(x) = c$.

Lemma 11. *Suppose there is a convex recoloring with at most k recolored vertices. Then for each $v \in V$, $|S(v)| \leq k + 1$.*

Proof. Suppose Γ' is a convex recoloring with at most k recolored vertices of Γ. Consider a vertex $v \in V$, and a color $c \in S(v)$. Suppose $c \neq \Gamma'(v)$. There are vertices w, x with $\Gamma(w) = \Gamma(x) = c$, and the path from w to x uses v. As v is not recolored to c, either w or x must be recolored. So, there can be at most k colors $\neq \Gamma'(v)$ that belong to $S(v)$. □

Lemma 12. *Suppose there is a convex recoloring with at most k recolored vertices. Then the treewidth of the vertex-color graph is at most $k + 3$.*

Proof. Without loss of generality, we suppose that Γ is surjective, i.e., for all $c \in \mathcal{C}$, there is a $v \in V$ with $\Gamma(v) = c$. (If Γ is not surjective, then colors c with $\Gamma^{-1}(c) = \emptyset$ are isolated vertices in the vertex-color graph, and removing these does not change the treewidth.)

Take an arbitrary vertex $r \in V$ as root of T. For each $v \in V$, set $X_v = \{v, p(v), \Gamma(v)\} \cup S(v)$, where $p(v)$ is the parent of v. For $v = r$, we take $X_v = \{v, \Gamma(v)\} \cup S(v)$.

It is easy to verify directly that $(\{X_v \mid v \in V\}, T)$ is a tree decomposition of the vertex-color graph of width at most $k + 3$. □

Theorem 3. *For each fixed k, there is a linear time algorithm that, given a vertex-colored tree T, decides if there is a convex recoloring of T with at most k recolored vertices.*

Proof. We show that for a fixed number of recolored vertices k, the Convex Recoloring problem can be formulated as a property of the vertex-color graph that can be stated in Monadic Second Order Logic. The result then directly follows by the result of Courcelle [8], that each such property can be decided in linear time on graphs with bounded treewidth. (See also [2, 5]. We use Lemma 12.)

We assume we are given the vertex-color graph $(V \cup \mathcal{C}, E')$, and have sets V and $\mathcal{C}$ distinguished. A recoloring Γ' that differs from Γ at most k vertices can be represented by these vertices $v_1, \ldots, v_k$ and by the corresponding values $c_1 := \Gamma'(v_1), \ldots, c_k := \Gamma'(v_k)$. Then, for a given color c, consider the set V_c defined as follows:

- If $v = v_i$ for some $1 \leq i \leq k$, then $v \in V_c$ if and only if $c_i = c$.
- If $v \notin \{v_1, \ldots, v_k\}$, then $v \in V_c$ if and only if (v, c) is an edge of the vertex-color graph.

Note, that V_c is the color class of c in the coloring Γ'. Hence Γ' is convex if and only if V_c is connected for each $c \in \mathcal{C}$. It is a standard fact, that MSOL can express connectedness, say by a formula $\varphi(X)$, where the set variable X represents V_c. Furthermore, the definition of V_c given above can be expressed even in first-order logic, say by a formula $\psi(x_1, \ldots, x_k, y_1, \ldots, y_k, z, X)$, where additionally $x_1, \ldots, x_k$ represent $v_1, \ldots v_k$, $y_1, \ldots, y_k$ represent $c_1, \ldots c_k$, and z represents c. Then, for this k, the Convex recoloring problem can be expressed by the formula

$$\exists x_1, \ldots, x_k \exists y_1, \ldots, y_k \Big(\bigwedge_{1 \leq i \leq k} V x_i \wedge \bigwedge_{1 \leq i < j \leq k} x_i \neq x_j \wedge \bigwedge_{1 \leq i \leq k} \mathcal{C} y_i \wedge \forall z \forall X (\psi \rightarrow \varphi) \Big).$$

The algorithms works as follows. Given a colored tree (T, Γ), it constructs the vertex-color graph and computes a tree-decomposition of width at most $k+3$. If this fails, the algorithm returns NO, in accordance with Lemma 12. Otherwise, using the tree-decomposition, it evaluates the above formula on the vertex-color graph. Each step uses at most linear time. □

References

1. J. Alber, M. R. Fellows, and R. Niedermeier. Polynomial-time data reduction for dominating sets. *J. ACM*, 51:363–384, 2004.
2. S. Arnborg, J. Lagergren, and D. Seese. Easy problems for tree-decomposable graphs. *J. Algorithms*, 12:308–340, 1991.
3. R. Bar-Yehuda, I. Feldman, and D. Rawitz. Improved approximation algorithm for convex recoloring of trees. In *Proceedings Third Workshop on Approximation and Online Algorithms WAOA 2005*, pages 55–68, 2005.
4. H. L. Bodlaender. A cubic kernel for feedback vertex set. In W. Thomas and P. Well, editors, *Proceedings 24th International Symposium on Theoretical Aspects of Computer Science, STACS 2007*, pages 320–331. Springer Verlag, Lecture Notes in Computer Science, vol. 4393, 2007.
5. R. B. Borie, R. G. Parker, and C. A. Tovey. Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families. *Algorithmica*, 7:555–581, 1992.
6. K. Burrage, V. Estivill-Castro, M. R. Fellows, M. A. Langston, S. Mac, and F. A. Rosamond. The undirected feedback vertex set problem has a poly(k) kernel. In H. L. Bodlaender and M. A. Langston, editors, *Proceedings 2nd International Workshop on Parameterized and Exact Computation, IWPEC 2006*, pages 192–202. Springer Verlag, Lecture Notes in Computer Science, vol. 4169, 2006.
7. B. Chor, M. Fellows, M. Ragan, F. Rosamond, I. Razgon, and S. Snir. Connected coloring completion for general graphs: algorithms and complexity. In Proceedings COCOON 2007.
8. B. Courcelle. The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Information and Computation*, 85:12–75, 1990.
9. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.
10. J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.
11. J. Gramm, A. Nickelsen, and T. Tantau. Fixed-parameter algorithms in phylogenetics. To appear in *The Computer Journal*.
12. S. Moran and S. Snir. Convex recolorings of strings and trees: Definitions, hardness results, and algorithms. In F. K. H. A. Dehne, A. López-Ortiz, and J.-R. Sack, editors, *Proceedings WADS 2005: 9th International Workshop on Algorithms and Data Structures*, pages 218–232. Springer Verlag, Lecture Notes in Computer Science, vol. 3608, 2005.
13. S. Moran and S. Snir. Efficient approximation of convex recolorings. In *Proceedings APPROX 2005: 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, published with Proceedings RANDOM 2005*, pages 192–208. Springer Verlag, Lecture Notes in Computer Science, vol. 3624, 2005.
14. R. Niedermeier. *Invitation to fixed-parameter algorithms*. Oxford Lecture Series in Mathematics and Its Applications. Oxford University Press, 2006.