FPT Algorithms with Polynomial Memory for the Convex String Recoloring Problem

Casper Hagenaars(ICA-3864014)
Supervisor: H.L.Bodlaender
University Utrecht

August 2018

Abstract

The Convex String Recoloring (CSR) problem measures the Hamming distance between a string of colored vertices to a convex string. This is a special case of the Convex Tree Recoloring (CTR) problem, which measures the Hamming distance to a perfect phylogeny, which was popularized by Moran and Snir [17]. It has been shown that even if the input graphs are restricted to strings, the problem is still NP-complete. In 2008 Bar-Yehuda et al. came up with a dynamic programming approach the can solve the CSR in $O^*(2^k)$ using exponential space. In this paper, it is shown that the CSR problem can be solved in $O^*(2^k)$ time using only polynomial space.

1 Introduction

Recoloring problems of graphs are problems that take (G, Γ, C), where $G = (V, E)$ is a graph and $\Gamma : \Gamma(v) \rightarrow c$ is a coloring function that maps a vertex to a color in C, where as few vertices as possible have to be assigned a new color such that the resulting coloring fulfills a certain property [9]. In this thesis this property is confined to convexity, i.e., the problem is to turn a given graph into a convex one. Convexity is the property that for each $x, y, z \in V$: if there is a path from x to z then it holds that if $\Gamma(x) = \Gamma(z)$ then $\Gamma(y) = \Gamma(z)$. So, in other words: A coloring function Γ is convex if each color class induces a connected subtree. One version of this problem that has been studied a lot is the Convex Tree Recoloring Problem (CTR). Here, the input of the problem is restricted to trees. This version was popularized by Morgan and Snir [17]. This problem is well studied as it has important applications in evolutionary biology. A more extensive overview of existing literature will be given in Section 1.6. Another version of the problem, that is also the scope of this paper, is when it is assumed that the input graph is a single path, also called a string. It has been shown that even in this special case the problem is NP-hard [17]. A special case of the string recoloring problem is the CSR-2. Here the graph is also a string, but

each color occurs at most two times. Even with this restriction, the problem is still NP-complete [12]. Another special case is the Convex Leaf Recoloring problem in which only the colors of the leafs are known. This problem has been popularized and been shown to also be NP-hard by Snir in his PhD thesis [20].

1.1 Definition

The problem is defined as follows:

Convex String Recoloring:
Each instance is a string of n vertices, a set of colors C, and a coloring function $\Gamma : V \to C$ together with a positive integer k. The problem is then whether it is possible to recolor at most k vertices such that the string becomes convex, meaning that each color class induces a connected substring.

1.2 Example

In this example yellow is already convex. Blue could be made convex by recoloring all vertices in between, which would require three recolorings. A better solution is to recolor all blue vertices, requiring only two recolorings.

1.3 Notations

In this paper, there are different cases for substrings of strings. For handy notation, these have a different representation, as shown below:

An important concept is a *block*. A *block* is a maximal set of vertices with the same color that induce a connected substring.

A lowercase Latin letter, e.g. y_i, represents a block with a size equal or larger than one. The i denotes which block of its color it is. So different colors will have a different Latin letter, and different blocks of the same color will have a different subscript.

An uppercase Latin letter, e.g. Y_i, represents a single vertex. Different colors will have a different letter, while different vertices with the same color will have a different subscript.

A Greek letter, e.g. α_i, represents a string that might contain several colors. The subscripts are used to differentiate between different strings. This notation is used to show that two blocks are not connected.

So as an example, assume there is a color, x, that occurs an unspecified amount of times in consecutive order in the beginning of the string and on the end of the string it occurs twice with some other color in between. This would be written as:

$$x_1\alpha_1 X_2 y_1 X_3$$

So, it first occurs as a block, then some other colors occur and then it occurs twice with the color y in between, that occurs an arbritary positive number of times.

1.4 Applications

The Convex String Recoloring problem originates from the Convex Tree Recoloring problem, which is an algorithmic problem that has interesting applications in biology. It is a way to reconstruct an evolutionary tree from biological data. The resulting evolutionary tree is also called a phylogeny, which has the natural constraints that each characteristic property has been evolved without reverse convergent transitions. A reverse transition means that a species regains a property that was lost by one of its ancestors. A convergent transitions means that two different species have the same characteristic even though their most recent common ancestor does not [17]. A phylogeny that satisfies these natural constraints is called a perfect phylogeny. These constraints mean that the resulting evolutionary tree has to be convex in each characteristic. Different characteristics can be represented by a color which means that each color has to be convex. From the perfect phylogeny problem also arose the problem of whether it is possible to triangulate a given colored graph such that no edges between vertices of the same color are added. This is strongly related to the problem of recognizing k-partial trees [7][14]. The Convex String Recoloring problem also has some applications, as the convexity is useful in gene expressions, in which the classification of different types of tumors [3][4]. The recoloring distance is used to identify Tuberculose strains as well [22].

1.5 Algorithmic approaches

There are several approaches to solving this problem. Mainly the Exact approach, the Approximate approach and the Fixed Parameter Tractability approach.

1.5.1 Exact algorithms

An exact algorithm is an algorithm that solves an optimization problem to optimality. If the problem is NP-hard then this cannot run in worst case polynomial time unless $P = NP$. The goal of an exact algorithm is firstly to get the exponential part as low as possible and secondly to get the subexponential parts as low as possible. Kanj and Kratsch [12] give an exact algorithm that solves the Convex String Recoloring problem in $O^*(2^{\frac{4n}{9}})$ time.

1.5.2 FPT

Instead of trying to compute the minimum number of recolorings needed, one can wonder whether it is possible to recolor a graph to make it convex with a

given number k of recolorings. This is a different approach to the same problem, but changes the problem from an optimization problem to a decision problem. More generally, given an object x, a fixed parameter tractability (FPT) problem is defined to be solved in $f(k) \cdot |x|^{O(1)}$, for some function f. The $f(k)$ typically stands for an exponential part that does not depend on x, but instead on some other variable k. Background on FPT algorithms can be found in the book Parameterized Complexity [11] or in Parameterized Algorithms [10].

1.5.3 Approximation algorithms

Approximation algorithms are algorithms that solve NP-hard optimization problems within a provable distance to the optimal solution. These algorithm are generally much faster than other algorithms with the downside that they do not guarantee to return an optimal solution. This is because these algorithms are based on the conjecture that $P \neq NP$. Because, if $P \neq NP$, there are no exact polynomial algorithms for NP-hard problems, although there are polynomial algorithms for these problems that return an approximate solution. The concept of an approximation algorithm can be formalized as follows:

A ρ-approximation algorithm A which returns the value $f(x)$ for instance x, is guaranteed to be not more than ρ times the optimal solution, in case of a minimization problem, with $\rho > 1$. i.e.: [13]

$$OPT \leq f(x) \leq \rho OPT$$

Another approximation method is an absolute performance guarantee defined for a bounded error c: [13]

$$(OPT - c) \leq f(x) \leq (OPT + c)$$

In a paper by Bar-Yehuda et al. an approximation algorithm for the Convex Tree Recoloring Problem was given [2]. The algorithm is an $(2 + \epsilon)$-approximation that runs in $O(n^2 + n\frac{1}{\epsilon}^2 4^{\frac{1}{\epsilon}})$ time.

1.5.4 Branching

This paper uses a specialized technique called *branching*, sometimes called *Search Tree Pruning*. Branching algorithms solve problems by recursively applying branching rules and reduction rules, until the problem is sufficiently easy to solve using different techniques. Branching has a natural property that each branch can be looked at individually, so it only uses polynomial space and can be calculated in parallel. A reduction rule takes an instance of a decision problem and modifies it such that the resulting instance returns yes if and only if the original instance returns yes. A reduction rule is defined as safe if the following holds: whenever a string t results from string s by applying a reduction rule x, then string t can be made convex by using k recolorings if and only if string s can be made convex using k recolorings. A branching step takes an instance of a decision problem and creates sub-instances of the problem such that the

original instance returns yes if and only if any of the sub-instances return yes. So, for example, suppose an instance of a problem has k recolorings left, and it could create two new problems each of which has $k-1$ recolorings left. This would be written as a (1, 1) branch as there are two branches that each bring k down by one and would give the following recurrence:

$$T(k) \leq T(k-1) + T(k-1)$$

The complexity of the example results from solving the following equation for x:

$$x^k = x^{k-1} + x^{k-1}$$

This example solves for $x = 2$ and more generally these equations can be solved using mathematical analysis for an exact result or by using an numerical approximation. A branching step is said to have a factor c, if the resulting complexity is $O(c^k)$. In the example this step would give a complexity of $O(2^k)$. The complexity of a branching algorithm is equal to the maximum complexity of all its branching steps. There is more on this in the book Fundamentals of Algorithmics [8].

1.6 Earlier work

There has been a lot of research about the tree variant (CTR) of this problem. It started with Moran and Snir [17] who showed that the problem is NP-hard and gave an FPT algorithm that runs in $O(k(k/\log k)^k n^4)$ time. Bar-Yehuda et al. [2] gave a dynamic programming approach that runs in $O(n^2 + n \cdot k \cdot 2^k)$ time with exponential space usage and a $(2+\epsilon)$-approximation algorithm that runs in $O(n^2 + n(1/\epsilon)^2 \cdot 4^{1/\epsilon})$ time. Kanj and Kratsch [12] gave an exact algorithm that runs in $O(2^{0.454n} n^{O(1)})$ time with exponential space usage. In 2007 Razgon gave an FPT algorithm that runs in $O^*(256^k)$ time [19]. There has also been research into involving other variables as done by Kannan and Warnow in 1996. They gave an algorithm that runs in $O(2^{2r} k^2 n)$ time using a different notation. Here, they use n species with k r-state characteristics [15]. This means that there are k different amount of colors and that each vertex has at most r different colors that are valid in the recoloring. Nederhof gave an $O(2^k c)$ time algorithm, where k stands for the amount of terminals [18]. Terminals, here, are a subset of all vertices. Bodlaender et al. showed that the CTR has a kernel of size $O(k^2)$ [6] for when all vertices have unit weight. The unit weight means that each recoloring has a cost of 1, opposed to some vertices being more expensive to recolor than others. That result was further generalized to be true even if vertices do not have unit weight by Bodlaender et al. [5].

The Convex String Recoloring problem has the following research done: Kanj and Kratsch [12] gave a $O(2^{\frac{4n}{9}} n^{O(1)})$ time algorithm. Moran and Snir gave 2-approximation algorithm for the Convex String Recoloring problem in 2007 that runs in $O(cn)$, where c is the number of colors and n the size of the input [21]. Because the CTR problem is a generalized version of this problem, all results for the CTR problem also apply to the CSR problem.

The Convex Leaf Recoloring problem has the following research done: Bachoore and Bodlaender showed that the CLR problem can be solved in $O(4^{OPT})$ time, where OPT stands for size of the optimal solution [1]. Kanj and Kratsch [12] showed it can be solved in $O(2^{\frac{n}{3}}n^{O(1)})$ time.

The CSR-2 problem has the following research: Lima and Wakabayashi gave a 2/3-approximation algorithm [16]. Kanj and Kratsch [12] gave a $O(2^{\frac{n}{4}}n^{O(1)})$ time algorithm. In this algorithm, Kanj and Kratsch introduce the notion of long pairs, short pairs and singletons, which will be used in the algorithm (see Lemma 11).

1.7 Polynomial memory

For the Convex String Recoloring problem there does not yet exist an algorithm that runs in $O(2^k)$ time that uses only polynomial memory. This is significant because in practice the problem instances can be large and memory is expensive. Therefore, if an algorithm uses exponential memory it quickly becomes unpractical to use.

1.8 Main Theorem

The main theorem of this paper is the following. The proof is given in Section 4.2.

Theorem 1 (Main Theorem)**.** *The Convex String Recoloring Problem can be solved in $O^*(2^k)$ time and polynomial memory.*

2 Preliminaries

To start, some definitions and lemmas are needed. Consider the following problem:

Convex String Removing:
Each instance is a string of n vertices, a set of colors C, and a coloring function $\Gamma : V \to C$ together with a positive integer k. The problem is then whether it is possible to remove at most k vertices such that the string becomes convex.

Lemma 1. *The Convex String Removing problem is equivalent to the Convex String Recoloring problem.*

Proof. If a Convex String Recoloring problem instance returns yes, then remove every vertex that got recolored. This would create a valid yes for this instance of Convex String Removing problem.
If a Convex String Removing problem instance returns yes, then insert each vertex that was removed back at its original position and give it the same color as its nearest neighbor that was not get removed. If two such vertices are equally far apart, then give it the same color as its left nearest neighbor that was not removed. This way a color that is convex will always remain convex and the

amount of recolorings will be equal to the amount of removed vertices. Therefore this will also return a yes for this instance of Convex Recoloring problem.

So the problems are equivalent. □

This is useful because of the following lemma:

Lemma 2. *[Removing vertices] Instead of recoloring vertices, they can be removed from the string.*

Proof. Because the Convex String Recoloring problem and the Convex String Removing problems are equivalent this will result in the same answer. This is done by counting the removal of a single vertex as a single recoloring by decreasing k by one. This guarantees that the amount of recolorings will not surpass k. □

What method will be used in this paper depends on whatever is convenient for obtaining easier arguments.

An important lemma is provided by Kanj and Kratsch [12]:

Lemma 3 (Exchange Lemma [12])**.** *Let (V, Γ) be an instance of CSR-2, and there exists a convex recoloring of V that recolors at most k vertices. Then there exists a convex recoloring of V that recolors at most k vertices such that each color in the range of Γ is used at least once.*

So in other words, if each color occurs at most twice then there always exists a solution in which each color is used at least once.

Lemma 4. *If a color is convex and is either on the start or on the end of the string, then it can be removed from the instance without decreasing k.*

Proof. There is never a need to recolor a convex color unless there is another color that is not convex that occurs on both the left and the right from the convex color. Therefore the leftmost or rightmost color will never be recolored if it is already convex and therefore does not need to be considered and can be removed safely. □

3 Branching on each color depending on how often it occurs

In this section a $O^*(2^k)$ time algorithm for the CSR problem will be given. Instead of considering how often a color occurs, it is interesting to look at how often a block of a certain color occurs. So if a color occurs ten times, but nine of those are consecutive, then it will be considered a two block color. The algorithm consists of a number of branches and reduction steps. For correctness and time analysis, it is assumed that these steps are executed in the given order. Meaning that each step is only considered when all previous steps have been exhausted. The first step branches on all colors that occur as more than four

blocks and each following branches step then assumes that there are no colors that occur as more than four blocks. Finally there will only be four cases that will describe all remaining colors that can then be reduced to only two cases which will then be solved in Lemma 12.

3.1 First branching steps

Lemma 5. *If a color occurs as at least five blocks, a branching step with factor* 1.968 *exists.*

Proof. It can be assumed that each block has size one, as the complexity will only decrease otherwise. If a color x occurs as five blocks, then it can be represented like this:

$$\alpha_1 x_1 \alpha_2 x_2 \alpha_3 x_3 \alpha_4 x_4 \alpha_5 x_5 \alpha_6.$$

Here, α represents a substring that does not contain x and x_i represents each of the blocks of color x.
There are five occurrences of x and there can be branched on all recolorings such that x is convex. For example, recoloring only x_1 and x_2 for some color other than x, means that α_4 and α_5 have to be recolored as well, because x_3, x_4 and x_5 remain the same color and have to become one convex block. The amount of ways to make a color with n blocks convex is $\frac{n(n+1)}{2}$.

For example, if there are n vertices of color x, and all but one are recolored, then there n different ways of doing this. If two vertices of the color x remain uncolored, then there are $n-1$ ways of doing this, as the two kept vertices have to be neighbors in order to become convex. This goes on until all vertices of color x remain uncolored, which can only be done in one way. So in total, this is equal to $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$. Therefore, if a color occurs n times, there are $\frac{n(n+1)}{2}$ ways to recolor them such that the result will be convex. Each of these will use at least $n-1$ recolorings, because the vertices of color x that are not recolored are separated by other vertices that will have to be recolored to make x a convex string. Instead of considering which color they get, the vertices can be removed from the string, as per Lemma 2. Therefore, the complexity of branching on a color that occurs n times will be $O(b^k)$, for the value of b for which the following equation is true:

$$b^k = (\frac{n(n+1)^{\frac{1}{n-1}}}{2})^k$$

For $n = 5$, $\frac{5(5+1)^{\frac{1}{5-1}}}{2} = 15^{\frac{1}{4}} \approx 1.968$. So solving the equation results in a complexity of $O(1.968^k)$. This function is a monotone decreasing function, so the complexity will decrease as n increases. So each color that occurs more than four times has a branching step with factor 2. However, this also means that each color that occurs as less than five blocks will result in a branching factor higher than 2. □

Lemma 6. *If a color occurs as exactly four blocks, then a branching step with factor 2 exists.*

Proof. This case can be represented as follows:

$$\alpha_1 x_1 \alpha_2 x_2 \alpha_3 x_3 \alpha_4 x_4 \alpha_5$$

If the algorithm would naively branch on all possibilities this gives:

$$x^k = 10x^{k-3}$$

This solves for $x \approx 2.15$. This can be improved using another method. This method is to branch on a color without actually making it convex, such that it can be solved more efficiently by another branching step later on. This means that, instead of branching on all ways a certain color can be made convex, some branches can be grouped together. If there are several branches that recolor one certain vertex, then only recoloring that vertex could become a new branch that replaces those other branches. This new branch uses less recolorings than the old ones, but it can replace several branches. This creates a tradeoff as it decreases the quality of a single branch, but it decreases the total amount of branches. By carefully choosing branches that can be merged, the overall complexity can be improved.

In this case by only recoloring x_1 and x_2 but not α_4, or only recoloring x_3 and x_4 but not α_2. This means that there does not need to be branched on any other recoloring that recolors both x_1 and x_2 or both x_3 and x_4. There will still need to be branched as normal on the recoloring that retains x_1, x_2 and x_3, on the recoloring that retains x_2, x_3 and x_4, on the recoloring that retains x_2 and x_3, and finally on the recoloring that retains all xs. These are all the cases in which x_2 and x_3 are both retained, but in these four cases the string will be made convex so they will cost three recolorings each. The complexity will then become:

$$x^k = 4x^{k-3} + 2x^{k-2}$$

This equation solves for $x = 2$, so a branching step with factor 2 exists.

□

Lemma 7. *Suppose x is a color that occurs as three blocks, represented as: $\alpha_1 x_1 \alpha_2 x_2 \alpha_3 x_3 \alpha_4$. If x_1, α_2, α_3, or x_3 has size larger than one, a branching step with factor 2 exists.*

Proof. There are six ways to make this string convex, by recolorings these:
Either x_2 and x_3, or x_1 and x_3, or x_1 and x_2, or α_2 and x_3, or x_1 and α_3, or α_2 and α_3. Branching on these would give a branching step with factor 2.45..., which can be improved with a case analysis.

Case A: ($|\alpha_2| > 1$ or $|\alpha_3| > 1$):
First suppose α_2 has size two: Instead of branching on all six possibilities, x_1 can be recolored such that x only occurs as two blocks which can be branched on more efficiently. This is done later on by a different branching step. From the remaining three branches in which x_1 is not recolored, two of them recolor

α_2. This will give a (1, 2, 3, 3) branch, which solves for $x = 2$ so gives a branching step with factor 2. If α_2 has size larger than two, the complexity will only improve. Because of symmetry, the proof in the Case that α_3 is larger than one is identical.

Case B: ($|x_1| > 1$ or $|x_3| > 1$):
First suppose x_1 has size two:: Then the same technique as in Case 1 can be used. Branching on recoloring x_3 and on all branches in which x_3 is retained gives a (1, 2, 3, 3) branch which has a branching step with factor 2. Because of symmetry, the proof in the case that x_3 is larger than 1 is identical.

□

So, from now on, it can be assumed that for each color x that appears as three blocks, represented as $\alpha_1 x_1 \alpha_2 x_2 \alpha_3 x_3 \alpha_4$, each of x_1, x_3, α_2 and α_3 all have size one. About the size of x_2 no assumptions can be made.

Lemma 8. *A color x that occurs as two blocks, written as: $x_1 \alpha_2 x_2$: if two of x_1, α_2, or x_2 have size larger than 1, a branching step with factor 2 exists.*

Proof. If two of the blocks x_1, α_2 or x_2 have a size larger than 1, then it can be branched on and give at least a $(2, 2, 1)$ branch, which is a branching step with factor 2. So after this branching step, if a color x occurs as two blocks, only one of x_1, α_2 or x_3 can have a size larger than 1. □

3.2 The Remaining Cases

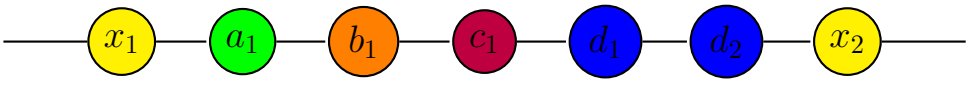
After branching, each instance now only contains colors that occur as one of the following cases:

Case 1: a single block color, a color that occurs as one block.

This is already convex. All the colors that have already been made convex also belong to this case.

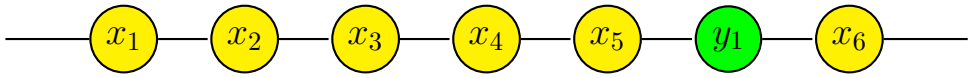
Case 2: a color occurring as two blocks, each of size 1, but with no limit on the amount of vertices in between.

This case can be written as $X_1 \alpha_1 X_2$. This is the first use of a capital letter as explained in Section 1.3. A capital letter X_i represents a single vertex with color x. In the picture below the color x is shown as yellow.



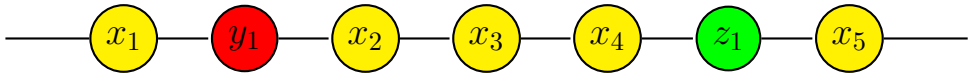
Case 3: a color occurring as two blocks, with only one vertex in between them and one of the two blocks having a size equal to 1 and the other having a size higher or equal to 1.

This can be written as $X_1\alpha_1 x_2$. This could also be $x_1\alpha_1 X_2$, but these are symmetrically equivalent and therefore treated as the same case.



Case 4: a color occurring as three blocks

This can be represented as $X_1Y_1x_2Z_1X_3$. Because of Lemma 7, this is the only way a color can occur as three blocks. This substring consists of only blocks of size 1. Note that Y_1 and Z_1 can be of the same color or of different colors.



4 Reduction Rules

It will now be shown how Cases 3 and 4 can be transformed to Case 1 or 2 using reduction rules, after which only Cases 1 and 2 remain. With only those two cases left, the proof can be finished using a final branching rule.
The following sections all contain proof by contradiction, some of these will use the knowledge that if a vertex Y has not been recolored in an optimal solution, then there cannot be another color that occurs both on the left and the right of Y. Whenever it is not specified to which color a vertex is recolored, it can be assumed that is got removed. This will make the proof easier and is still correct according to Lemma 2.

Reduction Rule 1. *Suppose a color x occurs as Case 4, $X_1Y_1x_2Z_1X_3$ or $X_1Y_1x_2Y_2X_3$, then Y_1 can be recolored.*

Lemma 9. *Reduction Rule 1 is safe.*

Proof. Start with $X_1Y_1x_2Z_1X_3$. The goal of this proof is to show that any optimal solution can be changed such that it remains optimal, but Y_1 has been recolored. Because it then follows that Y_1 can be recolored without excluding all optimal solutions. So in other words, showing that there exists at least one optimal solution in which Y_1 is recolored, guarantees that it is safe to recolor Y_1. So lets assume there is an optimal solution in which Y_1 has not been recolored and then show that the solution can be changed such that it is still convex and the amount of recolorings has not increased. This reduces the problem to four subcases, because Y_1 can be each of the four remaining cases.

Case 1: the color y occurs only once.
In this case, the color y does not occur anywhere else. This means that either X_1 or x_2 has been recolored, because otherwise the optimal solution would not be convex. This gives two subcases:

Case 1.1: only one of X_1 and x_2 has been recolored.
Now, this recoloring could be undone and instead be used to recolor Y_1. This means that x would still be convex and it would require at most the same amount of recolorings.

Case 1.2: both X_1 and x_2 have been recolored.
If both of them have been recolored then there are two possibilities: Either X_3 has been recolored as well and x as a color does not occur anymore or X_3 has not been recolored.

Case 1.2.1: the vertex X_3 has not been recolored.
Then the recolorings of X_1 and x_2 could be undone and instead be used to recolor Y_1 and Z_1. This guarantees that x is still convex and would require an equal amount of recolorings.

Case 1.2.2: the vertex X_3 has been recolored.
This means that all x have been recolored. So X_1 has been recolored, but because Y_1 has not been recolored, it cannot be a color that occurs on both sides of Y_1. Therefore, it is safe to undo X_1, back to x, and then recolor Y_1 to x as well.

Case 2: the color y occurs as either $Y_1\alpha_1Y_2$ or $Y_2\alpha_1Y_1$.
Then there exists one more instance of the color y, which either occurs on the right of Y_1 or on the left of Y_2.

Case 2.1: the color y occurs as $Y_1\alpha_1Y_2$.
It would look like this: $X_1Y_1x_2Z_1X_3\alpha_1Y_2$, or $X_1Y_1x_2Y_2X_3$. Note that in this Case α_1 could also have a size of 0.

Case 2.1.1: the color y occurs as $X_1Y_1x_2Z_1X_3\alpha_1Y_2$.
This means that $Y_2 \neq Z_1$.

Case 2.1.1.1: the vertex Y_2 has not been recolored.
Then all the vertices in between Y_1 and Y_2 must have been recolored. This means that X_1 is uncolored and x is convex. So if x is convex, then the recoloring of x_2 could be undone and instead be used to recolor Y_1. This would mean the string would still be convex and the amount of recolorings would be at least equal.

Case 2.1.1.2: the vertex Y_2 has been recolored.
Then the proof is identical to Case 1.

Case 2.1.2: the color y occurs as $X_1Y_1x_2Y_2X_3$.
This is the case that Y_1 and Z_1 are the same color.

Case 2.1.2.1: the vertex Y_2 has not been recolored.
Then x_2 has been recolored, and one of X_1 or X_3 as well. So these two recolorings can be undone to recolor Y_1 and Y_2.

Case 2.1.2.2: the vertex Y_2 has been recolored.
Then at least one of X_1 or x_2 has been recolored, because Y_1 will now be a single vertex that otherwise separates the color x, so undo the block of the two that got a recoloring to recolor Y_1. Then both Y_1 and Y_2 are recolored and all vertices in this string with color x are uncolored.

Case 2.2: the color y occurs as $Y_2\alpha_1Y_1$.
It would look like this: $Y_2\alpha_1X_1Y_1x_2Z_1X_3$. In the case that it would look like $X_1Y_2x_2Y_1X_3$, the proof would be identical to case 2.1.2.1.

Case 2.2.1: the vertex Y_2 has not been recolored.
Then X_1 and α_1 have been recolored. So reversing those recoloring and recoloring Y_1 would still leave the color y convex. If X_3 has not been recolored but x_2 has, then it x is not convex anymore. The recoloring of x_2 can be undone to recolor X_3 instead. This would result in a convex recoloring that recolors Y_1 without using more recolorings.

Case 2.2.2: the vertex Y_2 has been recolored
Then the rest of the string can be made convex by using the same proof as if Y_1 were Case 1.

Case 3: the color y occurs as $Y_1\alpha_1y_2$.
This could occur one of two ways: $y_2X_1Y_1x_2Z_1X_3$ or $X_1Y_1x_2Z_1X_3y_2$.

Case 3.1: the color y occurs as $y_2X_1Y_1x_2\alpha_1X_3$.
This means that the block y_2 occurs on the left of Y_1.

Case 3.1.1: the vertices y_2 have not been recolored.
Then X_1 has been recolored, in which case the recoloring of X_1 can be undone to recolor Y_1. This could mean that x is not convex anymore if x_2 has been recolored but X_3 has not. But that can be fixed be switching those recolorings.

Case 3.1.2: the vertices y_2 have been recolored.
In this case that recoloring could be undone to recolor Y_1 instead. This would mean that y stays convex, because it occured only once and x is not affected so still convex.

Case 3.2: the color y occurs as $X_1Y_1x_2Z_1X_3y_2$.
This means that the block y_2 occurs on the right of Y_1.

Case 3.2.1: the vertex y_2 has not been recolored.
Then all vertices in between have been recolored. This means all vertices of color x, except X_1, have been recolored. So, reversing the recoloring of x_2 to recolor Y_1 will keep all colors convex.

Case 3.2.2: the vertex y_2 has been recolored.
Then that recoloring could be undone to recolor Y_1. This would mean that y stays convex, because it occured only once and x is not affected so still convex.

Case 4: the color y occurs three times.
This could only happen like this: $Y_2E_1y_3X_1Y_1x_2Z_1X_3$, or symmetrically but then the proof would be equivalent.

Case 4.1: the vertices y_3 have not been recolored.
Then X_1 has been recolored, which can be undone to recolor Y_1. This only works if x remains convex, meaning that if x_2 has been recolored then that can be undone to recolor X_3. If x_2 was not recolored it was already convex. This way x will always remain convex.

Case 4.2: the vertices y_3 have been recolored.
Then that recoloring can be undone to recolor Y_1. This means that Y_1 was the only vertex with color y, because if X_1 is recolored then it would be suboptimal to recolor y_3 as well. □

Reduction Rule 2. *Suppose a color x occurs as Case 3, $X_1Y_1x_2$, then Y_1 can be recolored.*

Lemma 10. *Reduction Rule 2 is safe.*

Proof. Lets assume there is an optimal solution in which Y_1 has not been recolored. Then either X_1 or x_2 has been recolored, or both.

Case A: only X_1 has been recolored.
Then the recoloring of X_1 can be undone, and used to recolor Y_1 instead. This results in a convex string with an equal amount of recolorings.

Case B: x_2 has been recolored.
Then the recoloring of x_2 can be undone, and used to recolor Y_1 instead. This results in a convex string with at most an equal amount of recolorings.

Case C: both have been recolored.
This could not happen in an optimal solution. Because both recolorings can be undone to recolor Y_1 which decreases the amount of recolorings needed. □

4.1 Final Step

At this stage only Case 1 and Case 2 remain in the algorithm. This means there exist colors that occur as a single block, and there exist colors that occur as two blocks of size one. Kanj and Kratsch [12] gave an algorithm to solve CSR-2, in which each color occurs at most twice, in $O(n^{\frac{n}{4}})$ time that has some useful terminology that can be used here. Their first step in solving CSR-2 is by categorizing all colors. If a color occurs only once, it is called a *singleton*, otherwise each color occurs at most twice and can be called a *pair*. Those pairs can be divided into *long pairs* and *short pairs*. A long pair are all the pairs such that between the left and right vertex of the pair at least one of these is contained:

- both vertices of another pair
- a singleton

However, in Case 1 a color can still occur multiple times, so the problem is not yet equivalent to the CSR-2 problem. The next Lemma shows that the problem can be branched on if each color occurs as most twice and then there will be a lemma that shows that the solution of this branching will return the same answer even if a color that occurs as a single block with a size larger than 1.

Lemma 11. *If there are at most two vertices of each color, then a branching step with factor 2 exists.*

Proof. According to the Exchange Lemma 3 each color can be retained. So there is never a need to recolor all vertices contained within a long interval, meaning that for each long interval $X_1 \alpha X_2$ either X_1 or X_2 has to be recolored. Instead of being recolored to specific color, the vertex can also be removed according to Lemma 2. This means there is a branch with factor 2: remove X_1 or remove X_2.

Suppose now, all pairs are short. Consider then the leftmost color. If that color is convex then according to Reduction Rule 4 it can safely be removed from the instance without decreasing k. If the color is not convex, then the left vertex is not in between two other vertices so recoloring it can never make any other color convex. This means there is a branch of factor 2 by either removing the right vertex of the same color, again using Lemma 2, or removing all vertices in between. □

Lemma 12. *If there is an instance in which every color is either Case 1 or Case 2, then a branching step with factor 2 exists.*

Proof. Each color that occurs as only one block can be reduced to a single vertex. Then it can be solved using Lemma 11. This will always result in a correct answer. If Lemma 11 returns yes, then this instance will return yes as well, as no color that occurs as a single block has been recolored so the amount of recolorings are equal. If Lemma 11 returns no, then every recoloring of this

instance will be at least as bad, as adding extra vertices to an instance will never be able to decrease the amount of recolorings needed, so it will also return no. Therefore, a branching step with factor 2 exists. □

4.2 Main Theorem

Theorem 1 (Main Theorem)**.** *The Convex String Recoloring Problem can be solved in* $O^*(2^k)$ *time and polynomial memory.*

Proof. The problem can be solved by branching in order of the branching rules as they occur in this paper. Each branching step has a factor of at most 2, and each branching step and reduction rule can be executed in polynomial time. This means the complexity of the branching algorithm is obtained from the worst case step, which is a branching step with factor 2. Hence, this gives $O^*(2^k)$ time. The branching procedure can be done in polynomial memory. □

5 Discussion and further research

In this paper a branching approach on the FPT version of the Convex String Recoloring problem has been studied. The results are an improvement on existing algorithms in that it has the same runtime complexity as the dynamic programming algorithm described by Bar-Yehuda et al. but uses only polynomial space instead of exponential. Future research might include improving upon the few branching steps with factor 2. It is also possible to improve on the algorithm for specific cases of this problem, for example an algorithm that depends on the amount of colors or the amount of blocks that runs faster if these numbers are small. It would also be interesting to see if the methods used in this paper can be extended for the more general case of the Convex Tree Recoloring problem. For example in an algorithm that solves branches independently and somehow merges them together. Another advantage of the methods used in this paper is that most of them are relatively easy to program, but it could be an interesting topic to optimize the implementation of each branching and reduction step.

References

[1] E. Bachoore and H. Bodlaender. Convex recoloring of leaf-colored trees. TR UU-CS-2006-010, Department of Informatics and Computer Science, Utrecht University, 2006.

[2] R. Bar-Yehuda, I. Feldman, and D. Rawitz. Improved approximation algorithm for convex recoloring of trees. In *Approximation and Online Algorithms*, volume 3879, pages 55–68, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[3] A. Ben-Dor, N. Friedman, and Z. Yakhini. Class discovery in gene expression data. In *Proceedings of the Fifth Annual International Conference on*

Computational Biology, RECOMB '01, pages 31–38, New York, NY, USA, 2001. ACM.

[4] M. Bittner, P. Meltzer, Y. Chen, Y. Jiang, E. Seftor, M. Hendrix, M. Radmacher, R. Simon, Z. Yakhini, A. Ben-Dor, N. Sampas, E. Dougherty, E. Wang, F. Marincola, C. Gooden, J. Lueders, A. Glatfelter, P. Pollock, J. Carpten, E. Gillanders, D. Leja, K. Dietrich, C. Beaudry, M. Berens, D. Alberts, and V. Sondak. Molecular classification of cutaneous malignant melanoma by gene expression profiling. *Nature*, 2000.

[5] H. Bodlaender and M. Comas. A kernel for convex recoloring of weighted forests. In *SOFSEM 2010: Theory and Practice of Computer Science*, volume 5901, pages 212–223, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[6] H. Bodlaender, M. Fellows, M. Langston, M. Ragan, F. Rosamond, and M. Weyer. Quadratic kernelization for convex recoloring of trees. In *Computing and Combinatorics*, volume 4598, pages 86–96, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[7] H. Bodlaender and T. Kloks. A simple linear time algorithm for triangulating three-colored graphs. *J. Algorithms*, 15:160–172, 1992.

[8] Gilles Brassard and Paul Bratley. *Fundamentals of Algorithmics*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.

[9] M. Campêlo, A. Freire, R. Lima, F. Moura, and Y. Wakabayashi. The convex recoloring problem: Polyhedra, facets and computational experiments. *Math. Program.*, 156(1-2):303–330, March 2016.

[10] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer Publishing Company, Incorporated, 1st edition, 2015.

[11] Rodney G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer Publishing Company, Incorporated, 2012.

[12] I. Kanj and D. Kratsch. Convex recoloring revisited: Complexity and exact algorithms. In *Computing and Combinatorics*, volume 5609, pages 388–397, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[13] V. Kann. *On the Approximability of NP-complete Optimization Problems*. Ph.D. Thesis, Royal Institute of Technology, 1992.

[14] S. Kannan and T. Warnow. Inferring evolutionary history from dna sequences. *Proceedings of the 31th Annual Symposium on Foundations of Computer Science*, pages 362 – 371, 1990.

[15] S. Kannan and T. Warnow. A fast algorithm for the computation and enumeration of perfect phylogenies. *University of Philadelphia*, 1996.

[16] K. Lima and Y. Wakabayashi. Convex recoloring of paths. *Electronic Notes in Discrete Mathematics*, 37:165 – 170, 2011. LAGOS'11 – VI Latin-American Algorithms, Graphs and Optimization Symposium.

[17] S. Moran and S. Snir. Convex recolorings of strings and trees: Definitions, hardness results and algorithms. *Journal of Computer and System Sciences*, 74(5):850 – 869, 2008.

[18] J. Nederlof. Fast polynomial-space algorithms using inclusion-exclusion. *Algorithmica*, 65(4):868–884, Apr 2013.

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

[20] S. Snir. Phd thesis, department of computer science, technion, haifa, israel, 2004.

[21] S. Moran S. Snir. Efficient approximation of convex recolorings. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 192–208, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[22] A. Hirshand A. Tsolaki, K. DeRiemer, M. Feldman, and P. Small. Stable association between strains of mycobacterium tuberculosis and their human host populations. *Proceedings of the National Academy of Sciences*, 101(14):4871–4876, 2004.