On the Parameterized Complexity of the Perfect Phylogeny Problem

Jorke M. de Vlas

Utrecht University, the Netherlands; Linköping Universitet, Sweden

Abstract. This paper categorizes the parameterized complexity of the algorithmic problems PERFECT PHYLOGENY and TRIANGULATING COLORED GRAPHS when parameterized by the number of genes and colors, respectively. We show that they are complete for the parameterized complexity class XALP using a reduction from TREE-CHAINED MULTICOLOR INDEPENDENT SET and a proof of membership. We introduce the problem TRIANGULATING MULTICOLORED GRAPHS as a stepping stone and prove XALP-completeness for this problem as well. We also show that, assuming the Exponential Time Hypothesis, there exists no algorithm that solves any of these problems in time $f(k)n^{o(k)}$, where n is the input size, k the parameter, and f any computable function.

Keywords: Perfect phylogeny · Triangulated graphs · XALP · Parameterized complexity · W-hierarchy.

1 Introduction

A phylogeny is a tree that describes the evolution history of a set S of species. Every vertex corresponds to a species: leafs correspond to species from S, and internal vertices correspond to hypothetical ancestral species. Species are characterized by their gene-variants, and the quality of a phylogeny is determined by how well it represents those variants. In particular, a phylogeny is *perfect* if each gene-variant was introduced at exactly one point in the tree. That is, the subset of vertices that contain the variant is connected. PERFECT PHYLOGENY is the algorithmic problem of determining the existence of a perfect evolutionary tree. It has large implications on determining the evolutionary history of genetic sequences and is therefore of major importance. This application is not limited to biology: it can also be used to determine the history of languages or cultures.

The concept of phylogenies as an algorithmic problem has been well researched since the 60s. The first formal definition of PERFECT PHYLOGENY was given by Estabrook [10]. In 1974, Buneman showed that the problem can be reduced to the more combinatorial TRIANGULATING COLORED GRAPHS [6]

which by itself has also become an important, well-studied problem. An inverse reduction, and thus equivalence, was given by Kannan and Warnow [12]. In 1992, Bodlaender et al showed that PERFECT PHYLOGENY is NP-complete [3].

After Downey and Fellows introduced parameterized complexity [8], people have tried to determine the complexity of PERFECT PHYLOGENY when seen as a parameterized problem. There are two main ways to parameterize the problem: either by using the number of genes or by using the maximum number of variants for each gene. In the second case, the problem becomes FPT [13]. In the first case, the parameterized complexity was unknown. There are some partial results: On one hand, it was shown that the problem is $W[t]$-hard for every t [2]. On the other hand, there exists an algorithm that runs in $\mathcal{O}(n^{k+1})$ time and space (where n is the input size and k the parameter) which implies that the problem is contained in XP [14].

In this paper we will close this gap and show that PERFECT PHYLOGENY is complete for the complexity class XALP, which is a relatively new parameterized complexity class that was introduced by Bodlaender et al in [5]. We will show XALP-completeness by giving a reduction from the XALP-complete problem TREE-CHAINED MULTICOLOR INDEPENDENT SET, using TRIANGULATING MULTICOLORED GRAPHS as a stepping stone. This makes PERFECT PHYLOGENY the first example of a "natural" problem that is XALP-complete and allows it to be used as a starting point for many other XALP-hardness proofs. Finally, we use the same reduction to give some lower bounds dependent on the Exponential Time Hypothesis.

2   Definitions and Preliminary Results

All problems in this paper are parameterized. This means that the input contains a parameter separate from the rest of the input which allows us to analyze the runtime as a function of both the input and the parameter. If a parameterized problem with input size n and parameter k can be solved in $\mathcal{O}(f(k)n^c)$ time (with f any computable function and c any constant), we say that it is Fixed Parameter Tractable (FPT). A *parameterized reduction* is an algorithm that transforms instances of one parameterized problem into instances of another parameterized problem, runs in FPT time, and whose new parameter is only dependent on the old parameter. A *log-space reduction* is a parameterized reduction that additionally only uses $\mathcal{O}(f(k)\log(n))$ space. These reductions form the base of all parameterized complexity classes: all classes are defined up to equivalence under one of these reductions.

We use the following definition of PERFECT PHYLOGENY, which is a parameterized version of the original definition from Estabrook [10].

PERFECT PHYLOGENY (PP)

Input: A set G of genes, for each gene $g \in G$ a set V_g of variants, and a set S of species, where each species is defined as a tuple of gene-variants (exactly one per gene)

Parameter: The number of genes

Question: Does there exist a tree T of species (not necessarily from S) that contains all species from S and where the subtree of species containing a specific gene-variant is connected?

Triangulated and Colored Graphs. A graph is *colored* if every vertex is assigned a color. The graph is *properly colored* if there are no edges between vertices of the same color. For any cycle C in a graph, a *chord* is an edge between two vertices of C that are not neighbors on C. A graph is *triangulated* if every cycle of length at least four contains a chord. A *triangulation* of a graph is a supergraph that is triangulated. We now define the problem TRIANGULATING COLORED GRAPHS, which was first given by Buneman [6].

TRIANGULATING COLORED GRAPHS (TCG)

Input: A colored graph G

Parameter: The number of colors used

Question: Does there exist a properly colored triangulation of G?

We now introduce a multicolored variant of this problem. A graph is *multicolored* if every vertex is assigned a (possibly empty) set of colors. The graph is *properly multicolored* if there are no edges between vertices which share a color. This gives us the following problem:

TRIANGULATING MULTICOLORED GRAPHS (TMG)

Input: A multicolored graph G

Parameter: The number of colors used

Question: Does there exist a properly multicolored triangulation of G?

This problem is equivalent to TRIANGULATING COLORED GRAPHS under parameterized reductions. The general idea is to replace every multicolored vertex with a clique of normally colored vertices. A full proof is given in appendix section A. We now define a tree decomposition and state some well-known properties of triangulated colored graphs.

Definition 1 (Tree Decomposition). *Given a graph $G = (V, E)$, a tree decomposition is a tree T where each vertex (bag) is associated with a subset of vertices from T. This tree must satisfy three conditions:*

- *For each vertex $v \in V$, there is at least one bag that contains v.*
- *For each edge $e \in V$, there is at least one bag that contains both endpoints of e.*
- *For each vertex $v \in V$, the subgraph of bags that contain v is connected.*

Proposition 1. *Let G be a (multi)colored graph and C be a cycle.*

(i) *Suppose there exist two colors such that every vertex from C is colored with at least one of these colors. Then G admits no properly (multi)colored triangulation.*

(ii) *Let v be any vertex from C. In every triangulation of G there is either an edge between v's neighbors (in C) or a chord between v and some non-neighbor vertex from C.*

(iii) *G admits a properly colored triangulation if and only if G admits a tree decomposition where each bag contains each color at most once.*

Proof. Omitted from main text. See appendix section C.   □

XALP. A new complexity class in parameterized complexity theory is XALP [5]. Intuitively, it is the natural home of parameterized problems that are $W[t]$-hard for every t and contain some hidden tree-structure. For PERFECT PHYLOGENY, this tree-structure is the required phylogeny. For TRIANGULATING COLORED GRAPHS, it is the tree decomposition arising from Proposition 1(iii).

Formally, XALP is the class of parameterized problems that are solvable on an alternating Turing machine using $\mathcal{O}(f(k)\log(n))$ memory and at most $\mathcal{O}(f(k) + \log(n))$ co-nondeterministic computation steps, where n is the input size and k is the parameter. It is closed under log-space reductions. On Downey and Fellows' W-hierarchy, it lies between $W[t]$ and XP: XALP-hardness implies $W[t]$-hardness for every t and XALP membership implies XP-membership.

An example of an XALP-complete problem is TREE-CHAINED MULTICOLOR INDEPENDENT SET [5]. It is defined as a tree-chained variant of the well-known MULTICOLOR INDEPENDENT SET problem.

MULTICOLOR INDEPENDENT SET (MIS)
Input: A colored graph G
Parameter: The number of colors used
Question: Does G contain an independent set consisting of exactly one vertex of each color?

TREE-CHAINED MULTICOLOR INDEPENDENT SET (TCMIS)
Input: A binary tree T, for each vertex (bag) $B \in T$ a colored graph $G_B = (V_B, E_B)$ which we view as an instance of MULTICOLOR INDEPENDENT SET, and for each edge $e \in T$ a set of extra edges E_e between the graphs corresponding to the endpoints of e.
Parameter: The maximum number of colors used in each instance of MIS
Question: Does there exist a solution to each instance of MIS such that for each of the extra edges at most one of the endpoints is contained in the solution?

---

3   Main Results

In this section we will state the main result and explore some of its corollaries. We postpone the proof to the next sections.

Theorem 1. TRIANGULATING COLORED GRAPHS *is contained in XALP.*

Theorem 2. *There exists a log-space reduction from* TREE-CHAINED MULTI-COLOR INDEPENDENT SET *to* TRIANGULATING MULTICOLORED GRAPHS. *This reduction has a linear change of parameter.*

We will prove Theorem 1 in section 4. For Theorem 2, we describe the reduction in section 6 and prove correctness of this reduction in appendix section B.

Theorem 3 (Main Result). *The problems* PERFECT PHYLOGENY, TRIANGULATING COLORED GRAPHS *and* TRIANGULATING MULTICOLORED GRAPHS *are all XALP-complete.*

Proof. Combine Theorem 1, Theorem 2 and the equivalences between these three problems. □

We now use these complexity results to show some lower bounds on the space and time usage of PERFECT PHYLOGENY. In the remainder of this section, let n be the input size, k the parameter, f any computable function and c any constant. We start with a bound on the runtime based on the Exponential Time Hypothesis.

Proposition 2. *Assuming ETH, the problems* PERFECT PHYLOGENY, TRIANGULATING COLORED GRAPHS *and* TRIANGULATING MULTICOLORED GRAPHS *cannot be solved in* $f(k)n^{o(k)}$ *time.*

Proof. We use as a starting point that, assuming ETH, the problem MULTICOLOR INDEPENDENT SET cannot be solved in $f(k)n^{o(k)}$ time [7]. A trivial reduction to TREE-CHAINED MULTICOLOR INDEPENDENT SET using a single-vertex tree then shows the same for that problem. Since the reduction given in Theorem 2 has a linear change in parameter we obtain the same lower bound for TRIANGULATING MULTICOLORED GRAPHS. Finally, using the equivalences proven in section A and the known equivalences between TCG and PP (all with no change in parameter), the result follows. □

We now bound the space usage based on the Slice-wise Polynomial Space Conjecture (SPSC). This conjectures that LONGEST COMMON SUBSEQUENCE cannot be solved in both $n^{f(k)}$ time and $f(k)n^c$ space [15].

Corollary 1. *Assuming SPSC, the problems* PERFECT PHYLOGENY, TRIANGULATING COLORED GRAPHS *or* TRIANGULATING MULTICOLORED GRAPHS *cannot be solved in both* $n^{f(k)}$ *time and* $f(k)n^c$ *space.*

Proof. This proof uses the parameterized complexity class XNLP, which is defined as the class of parameterized problems that are solvable on a determenistic Turing machine using $\mathcal{O}(f(k)\log(n))$ memory. Comparing this with the definition of XALP shows that XALP-hardness implies XNLP-hardness. Since LARGEST COMMON SUBSEQUENCE is XNLP-complete [9], SPSC applies to all XNLP-hard problems and consequently also to all XALP-hard problems such as the three problems from this corollary. □

Compared with the existing algorithm that runs in $\mathcal{O}(n^{k+1})$ time and space [14], these are close but not tight gaps.

4   XALP Membership of Triangulating Colored Graphs

In this section we will prove Theorem 1.

Recall that TRIANGULATING COLORED GRAPHS asks us to determine whether a colored graph can be triangulated. Because of Proposition 1(iii), this is equivalent to finding a tree decomposition where each bag contains each color at most once. We claim that it is equivalent to find a tree decomposition where each bag contains each color *exactly* once.

Lemma 1. *A colored graph admits a tree decomposition where each bag contains each color at most once, if and only if it admits a tree decomposition where each bag contains each color exactly once.*

Proof. Omitted. See appendix section C.   □

We can now prove XALP membership.

Proof (of Theorem 1). We construct an alternating Turing machine (ATM) that, given an instance of TRIANGULATING COLORED GRAPHS, determines whether there exists a tree decomposition that contains each color exactly once. As a refresher, an ATM is a Turing machine that has access to both nondetermenistic and co-nondeterministic branching steps. A nondetermenistic step leads to ACCEPT if at least one successor state leads to ACCEPT and a co-nondetermenistic step leads to ACCEPT if all successor states lead to ACCEPT.

Our Turing machine is based on the XP-time algorithm we mentioned before [14]. We use the following claim without proof: given a graph G, a deterministic Turing machine can determine the whether two vertices belong to the same connected component in logarithmic space and polynomial time [16]. Repeated application of this result allows us to branch on all connected components of a graph using several co-nondeterministic steps.

Let G be any colored graph. The Turing machine will use nondeterministic steps to determine how to modify each bag compared to its parent and co-nondeterministic steps to simultaneously verify all subtrees. A precise formulation is given below:

- Using k nondeterministic steps, determine an initial bag S which contains one vertex of each color. During computations that lead to ACCEPT, each S will be a bag from the tree decomposition.
- Keep track of some vertex i that is initially NULL. This will signify the parent of the current bag S.
- Repeat the following until an ACCEPT or REJECT state is reached:
  - Determine all components of $G \setminus S$. Using a co-nondeterministic step, we branch into every component except the one that contains i. If this results in zero branches (e.g. when there are no other components), ACCEPT.
  - Let C be the component our current branch is in. We determine a vertex $v \in C$ with a nondeterministic step.
  - Determine the vertex $w \in S$ that has the same color as v. Since S contains one vertex of every color, w exists.

- If w is adjacent to any vertex from C, REJECT. This means that the current guess for how to modify S is incorrect.
- Modify S by adding v and removing w. Set i to w.

Overall, this alternating Turing machine constructively determines a rooted tree decomposition if one exists and thus solves TRIANGULATING COLORED GRAPHS. It also satisfies the memory requirement: the only memory usage is the set S, a constant number of extra vertices, and the memory needed to branch on connected components. Since memory of a vertex uses $\mathcal{O}(\log(n))$ space and $|S| = k$, we need $\mathcal{O}(k \log(n))$ space. We also use polynomial time: the time usage in the computation of each bag is a constant plus the time needed to find the connected components which results in polynomial time overall. Finally, we require at most $\mathcal{O}(n)$ co-nondeterministic computation steps: each co-nondeterministic step corresponds to branching into a subtree of the eventual (rooted) tree decomposition. Since each subtree introduces at least one vertex that is used nowhere else in the tree, there are at most $\mathcal{O}(n)$ subtrees.

Overall, we conclude that TRIANGULATING COLORED GRAPHS is contained in XALP. □

5   Zipper Chains and Gadgets

In this section we will introduce two multicolored graph components, the *zipper chain* and the *zipper gadget*. Their most important property is Proposition 4 which says that a zipper gadget has a fixed number of triangulations. This will be used in the XALP-hardness proof to represent a choice.

Definition 2. *A zipper chain is a multicolored graph that consists of two paths P and Q, not necessarily of the same length. The vertices of P and Q are respectively labeled as $p_1, p_2, \ldots$ and $q_1, q_2, \ldots$.*

The vertices are colored in 7 colors, with 2 colors per vertex. For ease of explanation, the colors are grouped in three groups with sizes 1, 2, and 4. The first group contains one color a which is added to odd-labeled vertices from P and even-labeled vertices from Q. The second group contains the color b_P which is added to even-labeled vertices of P and the color b_Q which is added to odd-labeled vertices of Q. The third group contains four colors c_1, c_2, c_3 and c_4 where c_i is added to vertices in P whose index is equivalent to i (mod 4) and vertices in Q whose index is equivalent to $i + 2$ (mod 4).

To summarize: the colors on path P are $ac_1, b_P c_2, ac_3, b_P c_4, ac_1, \ldots$ and those of Q are $b_Q c_3, ac_4, b_Q c_1, ac_2, b_Q c_3, \ldots$. This is visualized in Figure 1.

This color pattern repeats every four vertices. We call such a repetition a *tooth* of the zipper chain. If a triangulation of the zipper chain contains an edge between some tooth of P and some tooth of Q and at least one endpoint of this edge contains the color a, we say that these two teeth are *locked together*.

Proposition 3. *Let G be a graph containing a zipper chain (P, Q) and assume that there is a cycle that fully contains both P and Q. Any triangulation of G*

satisfies the following properties. Because of symmetry, all properties also hold with P and Q reversed.

(i)   *There is no edge between two non-adjacent vertices of P.*

(ii) *If there exist two edges between P and Q which share an endpoint in P, then the common endpoint in P is connected to all vertices of Q that lie between the other two endpoints.*

(iii) *If (p_i, q_j) is an edge, then either (p_{i+1}, q_j) or (p_i, q_{j+1}) is also an edge (as long as either p_{i+1} or q_{j+1} exists).*

(iv) *If (p_i, q_j) is an edge and p_i contains the color a, then (p_{i+1}, q_{j+1}) is also an edge (as long as both p_{i+1} and q_{j+1} exist). Here, q_{j+1} contains the color a.*

(v)   *If the i-th tooth of P and the j-th tooth of Q are locked together, then the $i+1$-th tooth of P and the $j+1$-th tooth of Q are also locked together.*

(vi) *Each tooth from P is locked together with at most one tooth from Q.*

Proof. We prove the statements in order.

(i)   If, to the contrary, such an edge does exist, then this edge together with the rest of P forms a cycle whose vertices alternate between the colors a and b_P. Because of Proposition 1(i) such a cycle cannot be triangulated.

(ii) Because of part (i), the cycle formed by these two edges and the path between the two endpoints on Q can only be triangulated by adding edges with an endpoint in P.

(iii) If (p_{i+1}, q_j) is not an edge then Proposition 1(ii) shows that p_i must be connected to another vertex in the cycle. Because of part (i) this neighbor is a vertex from Q. Because of part (ii) p_i must then also be connected to q_{j+1}.

(iv) Without loss of generality, say that p_i also contains the color c_1. Then, q_j must have the colors b_Q and c_3: all other color combinations share a color with p_i. Since q_{j+1} and p_i both contain the color a there is no edge between them so part (iii) implies that there is one between p_{i+1} and q_j. Since p_{i+2} and q_j share the color c_3, the same argument implies that there is an edge between p_{i+1} and q_{j+1}.

Fig. 1. A zipper chain.

Fig. 2. A possible triangulation of a zipper chain.

Fig. 3. A zipper gadget of size 2 and skew 1.

(v) Apply part (iv) four times to the edge connecting the i-th and j-th teeth of P and Q (respectively) to obtain an edge connecting the $i+1$-th and $j+1$-th teeth of P and Q (respectively).

(vi) Suppose to the contrary that a tooth from P is locked together with two teeth from Q. After some applications of parts (iii) and (iv) we find that the last vertex from the tooth from P (which has colors b_P and c_3) is connected to the last vertex from both teeth from Q. Because of part (ii), it is connected to all four vertices of the tooth from Q with the higher index. At least one of these also contains the color c_3 so this is a contradiction.

□

Zipper Gadgets. We now introduce the *zipper gadget*. It is a zipper chain with a specific length and a head and tail. An example is given in Figure 3.

Definition 3. *A zipper gadget of size n and skew s (satisfying $n > 0, s \geq 0$) is a zipper chain with the following modifications:*

- *The path P contains $4n - 1$ vertices, and thus n teeth. The last tooth misses one vertex.*
- *The path Q contains $4(n + s)$ vertices, and thus $n + s$ teeth.*
- *There are two additional vertices with just the color b_P: a head h and a tail t. The head is connected to the first vertices of P and Q and the tail to the last vertices of P and Q.*

Proposition 4. *There are exactly $s+1$ ways to triangulate a zipper gadget with skew s. These ways are identified by the offset at which the teeth lock together.*

Proof. Observe that the entire gadget forms a cycle, so Proposition 3 applies. Consider a vertex from P that contains the color a. Its neighbors share the color

Fig. 4. One of the two triangulations of the zipper gadget from Figure 3. This one has offset 0.

b_P, so Proposition 1(ii) shows that this vertex must be connected to some other vertex from the cycle. This cannot be h, t or another vertex from P since that would introduce a cycle containing only the colors a and b_P. Hence, the other endpoint must be a vertex from Q. This shows that each tooth from P is locked together with at least one tooth from Q.

Proposition 3(vi) now shows that each tooth from P is locked together with exactly one tooth from Q. Let Δ be the index of the tooth locked together with the first tooth of P. Proposition 3(v) now shows that any tooth with index i must be connected to tooth $i + \Delta$. Since Q has s more teeth than P, the offset Δ must be between 0 and s. We conclude that there are at most $s + 1$ ways to triangulate a zipper gadget with offset s and that these ways are identified by the offset.

To complete the proof, we now show that each case can actually be extended into a triangulation of the zipper gadget. Let Δ be the target offset. We add the following edges:

- An edge between the head h and every vertex from the first Δ teeth from Q.
- Edges between the i-th tooth from path P and the $i + \Delta$-th tooth from Q according to the pattern described in parts (iii) and (iv) of Proposition 3. This includes one overlap edge between the last vertex of each tooth from P and the first vertex from the next tooth from Q.
- An edge between the tail t and every vertex from the last $s - \Delta$ teeth from Q.

An example of such a triangulation is given in Figure 4. One can observe that this construction indeed triangulates the zipper gadget.   □

6   XALP-hardness of Triangulating Multicolored Graphs

In this section we describe the reduction from Theorem 2. The intuition is as follows. We want to reduce from TREE-CHAINED MULTICOLOR INDEPENDENT SET, which comes down to selecting a vertex from each color for each instance of MULTICOLOR INDEPENDENT SET. These choices must be compatible: we may

not choose two vertices which share an edge. The selection of a vertex will be done by creating zipper gadgets and interpreting each possible triangulation as a choice of a vertex. The compatibility checks will be done by combining two zipper gadgets in a way that makes it impossible to simultaneously triangulate both zipper gadgets in the respective choices. This construction borrows a technique, namely on how to create and combine gadgets from the TCMIS tree, from the XALP-completeness proof for TREE PARTITION WIDTH from [4]. The actual gadgets and their combination procedure are new.

Let an instance of TCMIS be given. Let T and k be the (binary) tree and parameter from this instance. For any node $n \in T$, we have an associated instance of MULTICOLOR INDEPENDENT SET consisting of a set of vertices S_c for each color c. Without loss of generality, we can assume that all sets S_c have the same size, say $r + 1$: if not, then we can add extra vertices to S_c that are connected to all other vertices and thus never occur in an independent set. We also assume that S_c is ordered in some way. This allows us to refer to vertices as $v_{n,c,i}$ where n is the node from T, c is the color, and i is the index in S_c (which, for ease of explanation, is zero-based). We also have a set of edges E, which we again assume to be ordered in some way. Each edge connects two vertices v_{n_1,c_1,i_1} and v_{n_2,c_2,i_2} where n_1 and n_2 are either the same node or neighbors in T and where c_1 and c_2 are distinct if $n_1 = n_2$. Let $m := |E|$ be the total number of edges.

First, we transform T into a rooted tree T' by choosing any node $u \in T$, adding two new nodes v and w and two edges (u, v) and (v, w), and setting w as the root. This way, each node from the original tree T has a parent and a grandparent in T'. We now construct a graph G which will be an instance of TMG. It will consist of several zipper gadgets in which some vertices have been identified with each other: that is, where some vertices with distinct colors are merged into one vertex with the combined set of colors. For each node n in T and each color c in its associated instance of MULTICOLOR INDEPENDENT SET, we add a zipper gadget $z_{n,c}$ of size $2mr + 1$ and skew r. The middle tooth of path P (with index $mr + 1$) is special: we call it the *middle*. We now say that this zipper gadget starts in n, passes through the parent of n and ends in the grandparent of n. This is supported with some vertex identifications: for each node n in T', we identify the heads of all zipper gadgets starting at n, the tails of all zipper gadgets ending at n, and the last vertex of the middles (with colors c_4 and b_P) of all zipper gadgets that pass through n. Observe that the path P of each zipper gadget now consists of m sets of r teeth between its head and middle, and also m sets of r teeth between its middle and tail.

Each zipper gadget is assigned its own set of 7 colors such that no two zipper gadgets which start, pass through, or end in a common node share a color. We claim that this can be done using at most $7k$ sets of 7 colors. Assign colors to nodes in order of distance to the root of T' (closest to the root first). Let n be the current node. All zipper gadgets that have already been assigned colors and intersect with zipper gadgets starting from n are those that start at either: n's parent, the other child of n's parent (n's sibling), n's grandparent, the other child of n's grandparent (n's uncle), or any of the two children from that vertex

(n's cousins). In total, this is at most $6k$ other zipper gadgets. To color the k zipper gadgets starting at n, we can thus use the remaining $7k - 6k = k$ sets of 7 colors.

In a triangulation of G, each zipper gadget will represent a choice of a vertex from S_c: if the zipper gadget is triangulated with offset Δ, then we choose the vertex with index Δ from S_c. Each of the m sets of r teeth between head and middle or between middle and tail will represent a restriction regarding one of the edges. For each edge e_i (with index i) with endpoints v_{n_1,c_1,i_1} and v_{n_2,c_2,i_2} we want to exclude the possibility of simultaneously triangulating the zipper gadget z_{n_1,c_1} with offset i_1 and the zipper gadget z_{n_2,c_2} with offset i_2. This is done as follows.

Let (P_1, Q_1) and (P_2, Q_2) be the paths which form the zipper gadgets. We now identify two vertices from P_1 and P_2 and add a new color d to some vertices from Q_1 and Q_2. This is visualized in Figure 5. The idea is that if we would triangulate both zipper gadgets in a way that adds edges between the vertices with color d and the merged vertex, then any triangulation of both zipper gadgets together forces an edge between the vertices with the color d which is impossible. We now describe exactly which vertices should be modified.

We consider two cases: either n_1 and n_2 are the same node or they are neighbors in T. In the first case, we consider the tooth with index ir from both P_1 and P_2 and identify the first vertex from these teeth with each other. We also consider tooth $ir + i_1$ from Q_1 and tooth $ir + i_2$ from Q_2 and add a new color d to the first vertex of these teeth. In the second case, we assume without loss of generality that n_1 is the parent of n_2. We do almost same as in the first case, except that we use the second half of the zipper gadget z_{n_2,c_2}: we identify the first vertex of tooth ir from P_1 and tooth $mr + 1 + ir$ from P_2, and we add color d to the first vertex of tooth $ir + i_1$ from Q_1 and tooth $mr + 1 + ir + i_2$ from Q_2.

This completes the construction. Observe that this construction uses $49k + 1$ colors ($7k$ sets of 7 colors for the zipper gadgets and one for the extra color d) and thus that the change in parameter is linear. Also observe that the construction can be performed in logarithmic working space since the creation and merging of the zipper gadgets only require local information from the original TCMIS instance. This shows that we indeed have a logspace reduction.

The proof that this TMG instance admits a triangulation if and only if the original TCMIS instance admits a solution is a direct result of the intuitive insights mentioned during the construction and thus omitted from the main text. A full proof is given in appendix section B.

7   Future Research

Let n be the input size, k the parameter, f any computable function, c any constant, and ϵ any small positive constant. We have shown that PERFECT PHYLOGENY and TRIANGULATING COLORED GRAPHS are XALP-complete and that (assuming ETH) there exist no algorithms that solve any of them in $f(k)n^{o(k)}$

Fig. 5. How two zipper gadgets are combined: two vertices from P_1 and P_2 are merged into one vertex and two vertices from Q_1 and Q_2 are given the extra color d.

time. This increases the number of "natural" problems in the complexity class XALP and gives more reason to determine properties of this complexity class. Additionally, these problems can be used as a starting point for XALP-hardness reductions for other parameterized problems.

Another future research direction might be to close or reduce the gaps between the current upper and lower bounds on space and time usage. For the time gap, there is a lower bound of $f(k)n^{o(k)}$ (assuming ETH) and an upper bound of $\mathcal{O}(n^{k+1})$ [14]. For the space gap on algorithms that run in $n^{f(k)}$ time, there is a lower bound of $f(k)n^c$ (assuming SPSC) and an upper bound of again $\mathcal{O}(n^{k+1})$ [14]. One way to close the time gap could be by assuming the Strong Exponential Time Hypothesis (SETH). We expect that, assuming SETH, a lower bound like $f(k)n^{k-\epsilon}$ should be possible.

We also rule out a research direction. Triangulating a colored graph comes down to finding a tree decomposition where each bag contains each color at most once. A similar problem would be to instead look for a *path* decomposition where each bag contains each color at most one. This problem, known as INTERVALIZING COLORED GRAPHS, is already NP-complete for the case $k = 4$ [1].

References

1. Bodlaender, H.L., de Fluiter, B.: On intervalizing k-colored graphs for dna physical mapping. Discrete Applied Mathematics **71**(1), 55–77 (1996). https://doi.org/10.1016/S0166-218X(96)00057-1
2. Bodlaender, H.L., Fellows, M.R., Hallett, M.T., Wareham, H., Warnow, T.J.: The hardness of perfect phylogeny, feasible register assignment and other problems on thin colored graphs. Theoretical Computer Science **244**(1), 167–188 (2000). https://doi.org/10.1016/S0304-3975(98)00342-9
3. Bodlaender, H.L., Fellows, M.R., Warnow, T.J.: Two strikes against perfect phylogeny. In: Kuich, W. (ed.) 19th International Colloquium on Automata, Languages and Programming, ICALP 1992. pp. 273–283. Springer Berlin Heidelberg, Berlin, Heidelberg (1992)

4. Bodlaender, H.L., Groenland, C., Jacob, H.: On the parameterized complexity of computing tree-partitions. In: Dell, H., Nederlof, J. (eds.) 17th International Symposium on Parameterized and Exact Computation, IPEC 2022. LIPIcs, vol. 249, pp. 7:1–7:20. Schloss Dagstuhl — Leibniz-Zentrum für Informatik (2022). https://doi.org/10.4230/LIPIcs.IPEC.2022.7

5. Bodlaender, H.L., Groenland, C., Jacob, H., Pilipczuk, M., Pilipczuk, M.: On the complexity of problems on tree-structured graphs. In: Dell, H., Nederlof, J. (eds.) 17th International Symposium on Parameterized and Exact Computation, IPEC 2022. LIPIcs, vol. 249, pp. 6:1–6:17. Schloss Dagstuhl — Leibniz-Zentrum für Informatik (2022). https://doi.org/10.4230/LIPIcs.IPEC.2022.6

6. Buneman, P.: A characterisation of rigid circuit graphs. Discrete Mathematics **9**(3), 205–212 (1974). https://doi.org/10.1016/0012-365X(74)90002-8

7. Chen, J., Chor, B., Fellows, M., Huang, X., Juedes, D., Kanj, I., Xia, G.: Tight lower bounds for certain parameterized NP-hard problems. In: 19th IEEE Annual Conference on Computational Complexity. vol. 19, pp. 150– 160 (2004). https://doi.org/10.1109/CCC.2004.1313826

8. Downey, R.G., Fellows, M.R.: Fixed-parameter tractability and completeness I: Basic results. SIAM Journal on Computing **24**(4), 873–921 (1995). https://doi.org/10.1137/S0097539792228228

9. Elberfeld, M., Stockhusen, C., Tantau, T.: On the space and circuit complexity of parameterized problems: Classes and completeness. Algorithmica **71**(3), 661–701 (2015). https://doi.org/10.1007/s00453-014-9944-y, https://doi.org/10.1007/s00453-014-9944-y

10. Estabrook, G., Johnson, C., McMorris, F.: A mathematical foundation for the analysis of cladistic character compatibility. Mathematical Biosciences **29**(1), 181–187 (1976). https://doi.org/10.1016/0025-5564(76)90035-3

11. Gavril, F.: The intersection graphs of subtrees in trees are exactly the chordal graphs. Journal of Combinatorial Theory, Series B **16**(1), 47–56 (1974). https://doi.org/10.1016/0095-8956(74)90094-X

12. Kannan, S., Warnow, T.: Inferring evolutionary history from DNA sequences. In: 31st Annual Symposium on Foundations of Computer Science, FOCS 1990. pp. 362–371 vol.1 (1990). https://doi.org/10.1109/FSCS.1990.89555

13. Kannan, S., Warnow, T.: A fast algorithm for the computation and enumeration of perfect phylogenies. SIAM Journal on Computing **26**(6), 1749–1763 (1997). https://doi.org/10.1137/S0097539794279067, https://doi.org/10.1137/S0097539794279067

14. McMorris, F.R., Warnow, T.J., Wimer, T.: Triangulating vertex-colored graphs. SIAM Journal on Discrete Mathematics **7**(2), 296–306 (1994). https://doi.org/10.1137/S0895480192229273

15. Pilipczuk, M., Wrochna, M.: On space efficiency of algorithms working on structural decompositions of graphs. ACM Trans. Comput. Theory **9**(4), 18:1–18:36 (2018). https://doi.org/10.1145/3154856

16. Reingold, O.: Undirected connectivity in log-space. Journal of the ACM **55**(4), 1–24 (2008). https://doi.org/10.1145/1391289.1391291

A   Triangulating Multicolored Graphs

In this appendix section we will prove the equivalence of Triangulating Colored Graphs (TCG) and Triangulating Multicolored Graphs (TMG). We begin with two lemmas.

Lemma 2. *Let G be a graph and T a tree decomposition of G. Then for each clique C there is a bag in T which fully contains C.*

Proof. We use induction on the size of C. For $|C| = 1$ and $|C| = 2$, the result follows directly from the definition of tree decomposition. Now suppose that $|C| > 2$ and let v be a vertex from C. By induction hypothesis, there must be a bag B_r that contains $C \setminus \{v\}$. Consider T as a rooted tree with B_r as root. Let T_v be the subtree of bags that contain v, and let B_v be the lowest common ancestor of all bags in T_v. Because T_v is connected, $v \in B_v$.

For any $w \in C \setminus \{v\}$, (v, w) is an edge so there must be a bag $B_w \in T_v$ that contains both v and w. Now, w is contained in both a descendant (B_w) and ancestor (B_r) of B_v, so $w \in B_v$. Since this holds for any $w \in C \setminus \{v\}$, we conclude that $C \subset B_v$ which completes the induction.   □

Some notation: for a graph $G = (V, E)$, tree decomposition T, and vertex set $C \subset V$, we define T_C as the subset of bags in T that contain C.

Lemma 3. *Let T be a tree where each vertex (bag) is associated with a subset of vertices from G. Then T is a tree decomposition of G if and only if: for each clique C in G, T_C is nonempty and connected.*

Proof. ($\Rightarrow$): Let T be a tree satisfying this condition. All three conditions to being a tree decomposition follow directly from the fact that single vertices and edges are cliques.

($\Leftarrow$): Let T be a tree decomposition. Let C be any clique. Lemma 2 shows that T_C is nonempty. We now show that it is connected. Let B_1, B_2 be two bags that fully contain C. For any vertex $v \in C$, we know that $T_{\{v\}}$ is connected, so every bag on the path between B_1 and B_2 contains v. Since this holds for any $v \in C$, the bags between B_1 and B_2 fully contain C. It follows that T_C contains every bag between B_1 and B_2. Since this holds for any B_1 and B_2, we conclude that T_C is connected.   □

We now prove the equivalence.

Theorem 4. *The problems* Triangulating Colored Graphs *and* Triangulating Multicolored Graphs *are equivalent under parameterized reductions. The parameter does not change under these reductions.*

Proof. The right implication is trivial: each instance of TCG is also an instance of TMG and each corresponding solution to TMG is also a solution to the TCG instance. We focus on the left implication. Let $G = (V, E)$ be an instance of TMG (with k colors). Construct a graph G' as follows. For each vertex $v \in V$ (with k_v colors) we create a clique C_v containing k_v vertices, each colored in one of the colors of v. For each edge $(v, w) \in E$ we add an edge between each pair of vertices from C_v and C_w to turn $C_v + C_w$ into a large clique. The resulting graph $G' = (V', E')$ is now an instance of TCG. We claim that it has a solution if and only if the original TMG instance has a solution.

First Direction. Let T be a tree decomposition of G that respects the multicoloring. For each vertex v, we replace each occurrence of v in bags of T with all the vertices from C_v. Call the result T'. We first show that T' is a tree decomposition of G' using Lemma 3.

Let a clique $C' \subset V'$ be given. For any vertex $v \in V$, we know that a bag of T' contains either all vertices from C_v or none. This implies the following: if C' contains some (but not all) vertices from C_v, then the set of bags that contain C' is the same as the set of bags that contain $C' \cup C_v$. Hence, without loss of generality we may assume that C' contains either all vertices from C_v or none.

Now, let C be the set of vertices $v \in V$ such that $C_v \subset C'$. By construction of T', we have that any bag in T contains C if and only if the corresponding bag in T' contains C'. Since T_C is a nonempty subtree, we find that $T'_{C'}$ must also be a nonempty subtree. As this holds for any C', we conclude that T' is a tree decomposition.

Additionally, T' respects the coloring: each vertex from V splits into a clique $C_v \subset V'$ which contains the same colors. Hence, each bag in T contains precisely the same colors as the corresponding bag in T' which implies that T' cannot contain a color more than once. This completes the first direction.

Second Direction. Let T' be a tree decomposition of G' that respects the coloring. We now construct the tree T as follows: T has the same shape as T', and a bag in T will contain a vertex v if and only if the corresponding bag in T' contains every vertex from C_v. We first show that T is a tree decomposition.

Let a clique $C \subset V$ be given. A bag in T now contains C if and only if the corresponding bag in T' contains C_v for each vertex $v \in C$. This set $C' := \bigcup_{v \in C} C_v$ is a clique: each C_v is a clique itself and for each two cliques C_v, C_w we have that v and w are connected in G (since C is a clique) and consequently that every vertex from C_v is connected to every vertex from C_w. Now, the alternate definition of tree decomposition shows that $T'_{C'}$ is a nonempty subtree. This implies that T_C is one as well. Since this holds for any clique C, we conclude that T is a tree decomposition.

Additionally, T respects the coloring: each bag in T corresponds to a bag in T' that contains the same colors (and possibly some more). Since T' respects the coloring, T must do so as well. This completes the second direction.

We have now shown that the constructed instance of TCG is equivalent to the original instance of TMG. Each vertex in G splits into at most k vertices in G', and each edge splits into at most k^2 edges in G'. Hence, G' contains at most nk vertices, mk^2 edges and k colors. Since this is polynomially many more than the original instance and since the parameter k did not change, the reduction is polynomial. This completes the proof. □

B   Full Proof of XALP-hardness

In this section we will show that the construction given in section 6 admits a triangulation if and only if the original TCMIS instance admits a solution. This completes the proof of Theorem 2.

First Direction. We will first show that if the TMG instance can be triangulated, then there is a solution to the TCMIS instance. Suppose that the TMG instance admits a triangulation. Because of Proposition 1(iii) this triangulation corresponds with a tree decomposition.

We first introduce some lemmas on tree decompositions.

Lemma 4 ([3, Proposition 4]). *Let G be a graph and P a path between two vertices v, w. In any tree decomposition T of G, let B_1 and B_2 be two bags such that $v \in B_1$ and $w \in B_2$. Now, each bag on the path from B_1 from B_2 in T contains at least one vertex from P.*

Lemma 5. *Let G be a multicolored graph and P a path between two vertices v, w whose vertices alternate between two colors. In any tree decomposition T of G, let B_1 and B_2 be two bags such that $v \in B_1$ and $w \in B_2$. Now, each vertex from P is contained in at least one bag on the path from B_1 from B_2 in T.*

Proof. Omitted. See section C.   □

Because of Proposition 4, each zipper gadget $z_{n,c}$ is triangulated with some offset $i_{n,c}$ between 0 and r. We now claim that, for each node n and color c, choosing the $i_{n,c}$-th vertex of S_c forms a solution to the TCMIS instance. To show this, we only need to show that we have chosen at most one endpoint for each edge from the TCMIS instance. Let e_j be an edge with endpoints v_{n_1,c_1,i_1} and v_{n_2,c_2,i_2}, let (P_1, Q_1) and (P_2, Q_2) be the paths of the corresponding zipper gadgets z_{n_1,c_1} and z_{n_2,c_2}, and suppose to the contrary that z_{n_1,c_1} and z_{n_1,c_1} are triangulated with offsets i_1 and i_2 (respectively). We consider two cases, either $n_1 = n_2$ or they are neighbors in T.

In the first case, the head, the middle and the first vertex of tooth jr of the paths P_1 and P_2 are pairwise identified with each other. We label these vertices as h, m, and u respectively. Let B_h and B_m be (any) bags that contain h and m (respectively). Note that there are four paths on G between h and m: each zipper gadget introduces two paths. Also, each of these paths alternates in color. Because of Lemma 5, there must be a bag B' on the path between B_h and B_m that contains u. Because of Lemma 4, B' must contain a vertex from both Q_1 and Q_2, say v_1 and v_2.

We now claim that v_1 and v_2 both contain the color d. First consider v_1, the other case is analogous. Since P_1 was triangulated with offset i_1, the vertex u must be connected to some vertex from tooth $ir + i_1$ from Q_1. Since u has colors a and c_1, the vertex from Q_1 can not have those colors. That leaves only one option: the first vertex (with colors b_Q and c_3). This is precisely the vertex that was given the color d in the construction, which completes the claim.

We now have a contradiction: B' contains two vertices with color d (one from each zipper gadget). This shows that we cannot have chosen both endpoints of e_j and completes the first case.

The second case is almost analogous. We assume without loss of generality that n_1 is a parent of n_2. We now have that the head, middle, and first vertex of tooth $ir + i_1$ from z_{n_1,c_1} are (respectively) identified with the middle, tail, and first vertex of tooth $mr + 1 + ir + i_2$ from z_{n_2,c_2}. We now apply the same reasoning as in the first case on these vertices. Overall, this completes the first direction.

Second Direction. We will now show that if the TCMIS instance has a solution, then the TMG instance can be triangulated. For each node n and color c, the TCMIS solution consists of a choice of some vertex $i_{n,c}$. In the TMG instance, this will correspond to a triangulation of the zipper gadget $z_{n,c}$ with offset $i_{n,c}$. This alone is not enough to obtain a triangulation; we need to add more edges. We will do this by creating a colored tree decomposition, which corresponds to a triangulation because of Proposition 1(iii).

We describe the tree decomposition in three steps. In the first step, we add a bag B_n for each node n of T'. This bag contains the shared vertex from all zipper gadgets that start, pass through, or end in n. It also contains a vertex of the path Q from each of those zipper gadgets: the first vertex for each zipper gadget that starts in n, the last vertex for each zipper gadget that ends in n, and the first vertex of tooth $rm + 1 + \Delta$ (where Δ is the offset of this zipper gadget) for each zipper gadget that passes through n.

In the second step we will add $m - 1$ bags between each pair of bags from the previous step. Let n_1 and n_2 be two adjacent nodes from T' and let B_1 and B_2 be the corresponding bags from the previous step. Without loss of generality, we assume that n_2 is the parent of n_1. We now create a sequence of bags $B'_0, B'_1, \ldots, B'_m$ with $B'_0 := B_1$ and $B'_m := B_2$. The path between B'_{i-1} and B'_i will correspond to the edge i.

Each bag B'_i will contain the first vertex from the ir-th tooth of the path P for every zipper gadget that starts at B_1 and passes through B_2. Additionally, it will contain the first vertex of the $ir + \Delta$-th tooth of the path Q for those zipper gadgets. Similarly, for zipper gadgets that pass through B_1 and end at B_2, it will contain the first vertex from the $mr + 1 + ir$-th tooth of P and the first vertex from the $mr + 1 + ir + \Delta$-th tooth of Q. This completes the second step.

Note that all the bags we have added so far do not contain any color more than once: each two zipper gadgets that share a common start, end, or middle vertex have distinct colors. Furthermore, each bag contains the extra color d at most once: each bag B'_i only contains the color d if n_1 contains an endpoint of edge i and if the zipper gadget corresponding to the color of that endpoint was triangulated with the proper offset. That is, if we chose the endpoint. Since the TCMIS instance chooses at most one endpoint for each edge, B'_i contains the extra color d at most once.

Recall that in the construction of the TMG instance, each edge i resulted in merging a vertex from two paths P for the zipper gadgets that correspond to its endpoints. In the bags we have added so far, this merged vertex does not cause a problem: it ends up in the same bag when viewed from both zipper gadgets.

In the third step, we add more bags between two adjacent bags from the previous step. Let B'_i and B'_{i+1} be two adjacent bags that correspond to the edge i. Recall that these bags each contain the first vertex of some tooth for some set of zipper gadgets. Also, the vertices between them do not contain any merged vertices; those where all handled in the previous steps.

For each zipper gadget on its own, we could easily complete the tree decomposition between these bags using the tree decompositions that correspond to the triangulations of the zipper gadgets (note that these are paths). For the set of zipper gadgets as a whole, we need some care to avoid adding the color d to the same bag twice. We do this by "sliding along" the zipper gadgets one by one: we start with some bags where we partially follow the triangulation of one zipper gadget, then add some bags where follow the triangulation of a second zipper gadget, and so on. We may also slide a bit further along the same zipper gadget multiple times. The exact order is described below.

- First, if B'_i contains the color d because of some zipper gadget, slide along that zipper gadget for 1 tooth. This way, the current bag no longer contains the extra color d.
- Now, loop over every zipper gadget one by one except possibly the one that causes B'_{i+1} to contain the color d. For each such zipper gadget, slide along it until we arrive at the tooth contained in bag B'_{i+1}. During this, there may have been some bags which contain the color d but it will not be in the current bag when we continue to the next zipper gadget.
- Finally, handle the zipper gadget that causes B'_{i+1} to contain the color d (if it exists). We slide over it until we reach the vertex contained in B'_{i+1}. Here, the final bag will contain the color d.

This completes the third step and thus also the construction of the tree decomposition. Because of the arguments given during the construction, each bag contains each color at most once. Since all bags consist of an interlocking of sliding along the zipper gadgets, we have that each vertex is contained in a connected subtree and that the endpoints of each edge are contained in some bag. This proves that the above construction is indeed a tree decomposition. We conclude that a tree decomposition exists, and consequently that the TMG instance has a solution. This completes the second direction.

Overall, we have now shown that the original TCMIS instance admits a solution if and only if the constructed TMG instance does. This completes the proof of Theorem 2. □

C   Remaining Proofs

This section includes some proofs that were omitted from the main text.

Proof (of Proposition 1). We prove the parts in order.

(i) Consider a triangulation of G. We use induction on the size of C. If $|C| = 3$, then one of the two colors must occur at least twice, hence one of the edges from C connects two vertices sharing a color. If $|C| \geq 4$, then C contains a chord. This chord splits C into two smaller cycles which share the chord as a common edge. Applying the induction hypothesis on one of these cycles shows that G cannot be triangulated into a properly colored graph.

(ii) Consider a triangulation of G and suppose to the contrary that both the edge and the chord do not exist. We use induction on the size of C. If $|C| = 3$, then v's neighbors are connected. If $|C| \geq 4$, then C contains a chord. This chord cannot connect v's neighbors or have v as an endpoint. Hence, it splits C into two smaller cycles (which share the chord as a common edge) such that both v and its neighbors are contained in one of them. Applying the induction hypothesis on this cycle completes the proof.

(iii) For colored graphs, the left implication was shown in [6] and the right implication in [11]. The proof for multicolored graphs is very similar to the colored version and thus omitted.

$\square$

Proof (of Lemma 1). The right implication is trivial. We focus on the left implication. Let T be a tree decomposition. If there is a color that is contained in some but not all bags, then there must exist two adjacent bags B_1, B_2 such that B_1 does not contain a vertex with this color and B_2 does. We can then add this vertex from B_2 to B_1 and observe that the result is still a valid tree decomposition where each bag contains each color at most once. By repeating this argument, we must eventually reach a state where every bag contains all colors. $\square$

Proof (of Lemma 5). Let u be a vertex from P and suppose to the contrary that no bag on the path from B_1 and B_2 contains u. We split P into two parts: P_1 from v to u and P_2 from u to w. Let B_3 be (any) bag that does contain u. Because of Lemma 4, each bag on the path from B_1 to B_3 contains some vertex from P_1. Analogously, each bag on the path from B_3 to B_2 contains some vertex from P_2. Since T is acyclic, the paths between B_1, B_2 and B_3 must intersect in some point, so there is a bag B which lies on all three paths. This bag then contains a vertex from P_1, a vertex from P_2, and it does not contain u. In particular, it contains two non-adjacent vertices from P. That means that, in the corresponding triangulation of G, there is an edge between two non-adjacent vertices of P. This edge combined with p induces a cycle which alternates between two colors and that is impossible because of Proposition 1(i). Hence, we arrive at a contradiction. $\square$