Computing Phylogenetic Roots with Bounded Degrees and Errors

Zhi-Zhong Chen [*] Tao Jiang [†] Guohui Lin [‡]

May 8, 2001

Abstract

Given a set of species and their similarity data, an important problem in evolutionary biology is how to reconstruct a phylogeny (also called evolutionary tree) so that species are close in the phylogeny if and only if they have high similarity. Assume that the similarity data are represented as a graph $G = (V, E)$ where each vertex represents a species and two vertices are adjacent if they represent species of high similarity. The phylogeny reconstruction problem can then be abstracted as the problem of finding a (phylogenetic) tree T from the given graph G such that (1) T has no degree-2 internal nodes, (2) the external nodes (*i.e.* leaves) of T are exactly the elements of V, and (3) $(u, v) \in E$ if and only if $d_T(u, v) \leq k$ for some fixed threshold k, where $d_T(u, v)$ denotes the distance between u and v in tree T. This is called the Phylogenetic kth Root Problem (PRk), and such a tree T, if exists, is called a *phylogenetic kth root* of graph G. The computational complexity of PRk is open, except for $k \leq 4$. In this paper, we investigate PRk under a natural restriction that the maximum degree of the phylogenetic root is bounded from above by a constant. Our main contribution is a linear-time algorithm that determines if G has such a phylogenetic kth root, and if so, demonstrates one. On the other hand, as in practice the collected similarity data are usually not perfect and may contain errors, we propose to study a generalized version of PRk where the output phylogeny is only required to be an *approximate* root of the input graph. We show that this and other related problems are computationally intractable.

Keywords: Phylogeny, phylogenetic root, computational biology, efficient algorithm, NP-hard.

1 Introduction

The reconstruction of evolutionary history for a set of species from quantitative biological data has long been a popular problem in computational biology. This evolutionary history is typically

[*]Supported in part by the Grant-in-Aid for Scientific Research of the Ministry of Education, Science, Sports and Culture of Japan, under Grant No. 12780241. Department of Mathematical Sciences, Tokyo Denki University, Hatoyama, Saitama 350-0394, Japan. Work done while visiting the Department of Computer Science, University of California, Riverside. Email: chen@r.dendai.ac.jp.

[†]Supported in part by a UCR startup grant and NSF Grants CCR-9988353 and ITR-0085910. Department of Computer Science, University of California, Riverside, CA 92521. Email: jiang@cs.ucr.edu. On leave from McMaster University.

[‡]Supported in part by NSERC and a Startup Grant from University of Alberta. Department of Computing Science, University of Alberta, Edmonton, Alberta T6G 2E8, Canada. Email: ghlin@cs.ualberta.ca. Work mostly done in McMaster University and University of Waterloo.

modeled by an evolutionary tree or *phylogeny*. A phylogeny is a tree where the leaves are labeled by species and each internal node represents a speciation event whereby an ancestral species gives rise to two or more child species. Both rooted and unrooted trees have been used to describe phylogenies in the literature, although they are practically equivalent. In this paper, we will consider only unrooted phylogenies for the convenience of presentation. [1] The internal nodes of a phylogeny have degrees (in the sense of unrooted trees, *i.e.* the number of incident edges) at least 3. Proximity within a phylogeny in general corresponds to similarity in evolutionary characteristics.

Many phylogenetic reconstruction algorithms have been proposed and studied in the literature [11]. In this paper we investigate the computational feasibility of a graph-theoretic approach for reconstructing phylogenies from similarity data. Specifically, interspecies similarity is represented by a graph where the vertices are the species and the adjacency relation represents evidence of evolutionary similarity. A phylogeny is then reconstructed from the graph such that the leaves of the phylogeny are labeled by vertices of the graph (*i.e.* species) and for any two vertices of the graph, they are adjacent in the graph if and only if their corresponding leaves in the phylogeny are connected by a path of length at most k, where k is a predetermined proximity threshold. To be clear, vertices in the graph are called *vertices* while those in the phylogeny *nodes*. Recall that the length of the (unique) path connecting two nodes u and v in phylogeny T is the number of edges on the path, which is denoted by $d_T(u, v)$. This approach gives rise to the following algorithmic problem [7]:

> Phylogenetic kth Root Problem (PRk):
> Given a graph $G = (V, E)$, find a phylogeny T with leaves labeled by the elements of V such that for each pair of vertices $u, v \in V$, $(u, v) \in E$ if and only if $d_T(u, v) \leq k$.

Such a phylogeny T (if exists) is called a *phylogenetic* kth *root*, or a kth *root phylogeny*, of graph G. Graph G is called the kth *phylogenetic power* of T. For convenience, we denote the kth phylogenetic power of any phylogeny T as T^k. Thus, PRk asks for a phylogeny T such that $G = T^k$.

1.1 Connection to Graph and Tree Roots, and Previous Results

Phylogenetic power might be thought of as a *Steiner* extension of the standard notion of *graph power*. A graph G is the kth *power* of a graph H (or equivalently, H is a kth *root* of G) if vertices u and v are adjacent in G if and only if the length of the shortest path from u to v in H is at most k. An important special case of graph power/root problems is the following:

> Tree kth Root Problem (TRk):
> Given a graph $G = (V, E)$, find a tree $T = (V, E_T)$ such that $(u, v) \in E$ if and only if $d_T(u, v) \leq k$.

If T exists then it is called a *tree* kth *root*, or a kth *root tree*, of graph G.

[1]But some of our hardness proofs will also use rooted trees as intermediate data structures in the construction.

The special case TR2 is also known as the Tree Square Root Problem [8]. Correspondingly, we call PR2 the Phylogenetic Square Root Problem. There is rich literature on graph root and power (see [2, Section 10.6] for an overview), but few results on phylogenetic/tree roots/powers. It is NP-complete to recognize a graph power [9], nonetheless, it is possible to determine if a graph has a kth root tree, for any fixed k, in $O(n^3)$ time, where n is the number of vertices in the input graph [4]. In particular, determining if a graph has a tree square root can be done in $O(n+e)$ time [8], where e is the number of edges in the input graph. Recently, Nishimura, Ragde, and Thilikos [10] presented an $O(n^3)$-time algorithm for a variant of PRk, for $k \leq 4$, where internal nodes of the output phylogeny are allowed to have degree 2. More recently, Lin, Kearney, and Jiang [7] introduced a novel notion of *critical clique* and obtained an $O(n+e)$-time algorithm for PRk, for $k \leq 4$. Unfortunately, both algorithms cannot be generalized to $k \geq 5$.

1.2 Our Contribution

In the practice of phylogeny reconstruction, most phylogenies considered are trees of degree 3 [11] because speciation events are usually bifurcating events in the evolutionary process. In such *fully resolved* phylogenetic trees, each internal node has three neighbors and represents a speciation event that some ancestral species splits into two child species. Nodes of degrees higher than 3 are introduced only when the input biological (similarity) data is not sufficient to separate individual speciation events and hence several such events may be collapsed into a non-bifurcating (super) speciation event in the reconstructed phylogeny. Hence in this paper, we consider a restricted version of PRk where the output phylogeny is assumed to have degree at most Δ, for some fixed constant $\Delta \geq 3$. For simplicity, we call it the Degree-Δ PRk and denote it in short as ΔPRk. Since in the practice of computational biology the set of species under consideration are more or less related, we are mostly interested in connected graphs. The main contribution of this paper is a linear-time algorithm that determines, for any input connected graph G and constant $\Delta \geq 3$, if G has a kth root phylogeny with degree at most Δ, and if so, demonstrates one such phylogeny. The basic construction in our algorithm is a nontrivial application of bounded-width tree-decomposition of certain chordal graphs [2].

Notice that the input graph in PRk is derived from some similarity data, which is usually inexact in practice and may have erroneous (spurious or missing) edges. Such errors may result in graphs that has no phylogenetic roots. Hence, it is natural to consider a more relaxed problem where we look for phylogenetic trees whose powers are *close* to the input graphs. The precise formulation is as follows:

Closest Phylogenetic kth Root Problem (CPRk):
Given a graph $G = (V, E)$ and a nonnegative integer ℓ, find a phylogeny T with leaves labeled by V such that G and T^k differ by at most ℓ edges. That is,

$$\left|E(G) \oplus E(T^k)\right| = \left|\left(E(G) - E(T^k)\right) \cup \left(E(T^k) - E(G)\right)\right| \leq \ell.$$

A phylogeny T which minimizes the above edge discrepancy is called a *closest* kth *root phylogeny* of graph G.

The Closest Tree kth Root Problem (CTRk) is defined analogously. Notice that CTR1 is trivially solved by finding a spanning tree of the input graph. Kearney and Corneil [4] proved that CTRk is NP-complete when $k \geq 3$. The computational complexity for CTR2 had been open for a while and is recently shown to be intractable by Jiang, Lin, and Xu [3]. In this paper, we will show that CPRk is NP-complete, for any $k \geq 2$. Another closely related problem, the Steiner kth Root Problem (where $k \geq 1$), is also studied.

We introduce some notations and definitions, as well as some existing related results, in the next section. Our main result on bounded-degree PRk is presented in Section 3. The hardness of closest phylogenetic root and Steiner root problems is discussed in Section 4. We close the paper with some open problems in Section 5.

2 Preliminaries

We employ standard terminologies in graph theory. In particular, the subgraph of a graph G induced by a vertex set U of G is denoted by $G[U]$, the degree of a vertex v in G is denoted by $deg_G(v)$, and the maximum size of a clique in G is denoted by $\omega(G)$. First, it is obvious that if a graph has a kth root phylogeny, then it must be *chordal*, that is, it contains no induced subgraph which is a cycle of size greater than 3.

Definition 2.1 *A* tree-decomposition *of a graph $G = (V, E)$ is a pair $\mathcal{D} = (\mathcal{T}, \mathcal{B})$ consisting of a tree $\mathcal{T} = (U, F)$ and a collection $\mathcal{B} = \{B_\alpha \mid B_\alpha \subseteq V, \alpha \in U\}$ of* sets *(called* bags*) for which*

- $\bigcup_{\alpha \in U} B_\alpha = V$,
- *for each edge $(v_1, v_2) \in E$, there is a node $\alpha \in U$ such that $\{v_1, v_2\} \subseteq B_\alpha$, and*
- *if $\alpha_2 \in U$ is on the path connecting α_1 and α_3 in $\mathcal{T}$, then $B_{\alpha_1} \cap B_{\alpha_3} \subseteq B_{\alpha_2}$.*

The treewidth *associated with this tree-decomposition $\mathcal{D} = (\mathcal{T}, \mathcal{B})$ is $tw(G, \mathcal{D}) = \max_{\alpha \in U} |B_\alpha| - 1$.*

The treewidth *of graph G, denoted by $tw(G)$, is the minimum $tw(G, \mathcal{D})$ taken over all tree-decompositions $\mathcal{D}$ of G.*

A clique-tree-decomposition *of G is a tree-decomposition $(\mathcal{T}, \mathcal{B})$ of G such that each bag in $\mathcal{B}$ is a maximal clique of G.*

Lemma 2.1 [5] *Every chordal graph has a clique-tree-decomposition.*

From the proof of Lemma 2.1 given in [5], it is not difficult to see that a clique-tree-decomposition $\mathcal{D} = (\mathcal{T}, \mathcal{B})$ of a given chordal graph G can be computed in linear time if $\omega(G) = O(1)$. We can further modify $\mathcal{D}$ so that $deg_{\mathcal{T}}(\alpha) \leq 3$ for each node α of $\mathcal{T}$ [1]. This modification takes linear time too if $\omega(G) = O(1)$.

Hereafter, a tree-decomposition of a chordal graph G always means a clique-tree-decomposition $\mathcal{D} = (\mathcal{T}, \mathcal{B})$ of G such that $deg_{\mathcal{T}}(\alpha) \leq 3$ for all nodes α of $\mathcal{T}$. Furthermore, in the sequel, we abuse the notations to use $\mathcal{D}$ to denote the tree $\mathcal{T}$ in it (since we will use T to denote the kth root phylogeny of graph G), and denote the bag associated with a node α of $\mathcal{D}$ by B_α.

3 Algorithm for Bounded-Degree PRk

This section presents a linear-time algorithm for solving 3PRk. The adaptation to ΔPRk where $\Delta \geq 4$ is straightforward and is hence omitted here.

We assume that the input graph $G = (V, E)$ is connected. We further assume that G is not complete but is chordal; otherwise the problem is trivially solved in linear time. Since every vertex $v \in V$ appears as an external node (*i.e.* leaf) in the kth root phylogeny, the maximum size $\omega(G)$ of a clique in G can be bounded from above by a constant $f(k)$, where

$$f(k) = \begin{cases} 3 \cdot 2^{\frac{k}{2}-1}, & \text{if } k \text{ is even,} \\ 2^{\frac{k+1}{2}} - 1, & \text{if } k \text{ is odd.} \end{cases}$$

So, we can construct a clique-tree-decomposition $\mathcal{D}$ of G in linear time. The basic idea behind our algorithm is to do a dynamic programming on a rooted version of the decomposition $\mathcal{D}$. The dynamic programming starts at the leaves of $\mathcal{D}$ and proceeds upwards. After processing the root, the algorithm will construct a kth root phylogeny of G if there is any. The processing of a node α of $\mathcal{D}$ can be sketched as follows. Let U_α be the union of the bags associated with α and its descendants in $\mathcal{D}$. While processing α, the algorithm computes a set of trees T such that (1) T may possibly be a subtree of a kth root phylogeny of G, (2) all vertices of U_α are leaves of T, and (3) each leaf of T not contained in U_α is not labeled. The unlabeled leaves of T serve as ports from which we can expand T so that it may eventually become a kth root phylogeny of G. The crucial point we will observe is that we only need those ports that are at distance at most k apart from vertices of B_α in T. This point implies that the number of necessary ports only depends on k and hence is a constant.

One more notation is in order. For two adjacent nodes α and β of $\mathcal{D}$, let $U(\alpha, \beta) = \bigcup_\gamma B_\gamma$ where γ ranges over all nodes of $\mathcal{D}$ with $d_{\mathcal{D}}(\gamma, \alpha) < d_{\mathcal{D}}(\gamma, \beta)$. In other words, if we root $\mathcal{D}$ at node β, then $U(\alpha, \beta)$ is the union of the bags associated with α and its descendants in $\mathcal{D}$. A useful property of $\mathcal{D}$ is that for every internal node β and every two neighbors α_1 and α_2 of β in $\mathcal{D}$, G has no edge between any vertex of $U(\alpha_1, \beta) - B_\beta$ and any vertex of $U(\alpha_2, \beta) - B_\beta$.

3.1 Ideas behind the Dynamic Programming Algorithm

Note that since $\Delta = 3$, every internal node in a kth root phylogeny T of G has degree exactly 3.

Definition 3.1 *Let U be a set of vertices of G. A* relaxed phylogeny *for U is a tree R satisfying the following conditions:*

- *The degree of each internal node in R is* 3.
- *Each vertex of U is a leaf in R and appears in R only once. For convenience, we call the leaves of R that are also vertices of U* final leaves *of R, and call the rest leaves of R* temporary leaves *of R.*
- *For every two vertices u and v of U, u and v are adjacent in G if and only if $d_R(u, v) \leq k$.*

- *Each temporary leaf v of R is assigned a pair (γ, t), where γ is a node of $\mathcal{D}$ and $0 \leq t \leq k$. We call γ the* color *of v and call t the* threshold *of v. For convenience, we denote the color of a temporary leaf v of R by $c_R(v)$, and denote the threshold of v by $t_R(v)$.*

Intuitively speaking, the temporary leaves of R serve as ports from which we can expand R so that it may eventually become a kth root phylogeny of G.

Recall that our algorithm processes the nodes of $\mathcal{D}$ one by one. While processing a node α of $\mathcal{D}$, the algorithm finds out all relaxed phylogenies for B_α that are subtrees of kth root phylogenies of graph G. The following lemma shows that such relaxed phylogenies for B_α have certain useful properties.

Lemma 3.1 *Let T be a kth root phylogeny of G. Let α be a node of $\mathcal{D}$. Root T at an arbitrary leaf that is in B_α. Define a* pure *node to be a node w of T such that α has a neighbor γ in $\mathcal{D}$ such that all leaf descendants of w in T are in $U(\gamma, \alpha) - B_\alpha$. Define a* critical *node to be a pure node of T whose parent is not pure. Let R be the relaxed phylogeny for B_α obtained from T by performing the following steps of operations:*

1. *For every critical node w of T, perform the following:*
   (a) *Compute the minimum distance from w to a leaf descendant of w in T; let i_w denote this distance. (*Comment: *$i_w \leq k$ or else the leaf descendants of w in tree T would be unreachable from the outside in graph G.)*
   (b) *Find the neighbor γ of α such that all leaf descendants of w in T are contained in $U(\gamma, \alpha)$.*
   (c) *Delete all descendants (excluding w, of course) of w, and assign the pair (γ, i_w) to w.*
2. *Unroot T.*

Then, the resultant R has the following properties:

- *For every temporary leaf v of R, $c_R(v)$ is a neighbor of α in $\mathcal{D}$.*
- *For every two temporary leaves u and v of R with different colors, it holds that $t_R(u) + t_R(v) + d_R(u, v) > k$.*
- *For every neighbor γ of α in $\mathcal{D}$, every temporary leaf v of R with $c_R(v) = \gamma$, and every final leaf w of R with $w \notin B_\gamma$, it holds that $d_R(v, w) + t_R(v) > k$.*
- *For every internal node v of R, either at least one descendant of v is a final leaf of R, or there is a final leaf u of R with $d_R(u, v) \leq k - 1$.*

PROOF. Obviously, R is a relaxed phylogeny for B_α. Since T is a phylogeny of G, it follows immediately that R has the first three properties in the lemma. To prove the fourth property, it suffices to prove that for every critical node w of T whose parent p in T has no leaf descendant contained in B_α, there is a vertex u in B_α such that $d_T(u, p) \leq k - 1$. To this end, let v be a leaf descendant of p that is closest to p among all leaf descendants of p in T. Let $u_1, u_2, \ldots, u_q$ be all leaves in T that are at distance at most k apart from v in T but are not descendants of p in T. Since

G is connected, $q \geq 1$. We claim that $\{u_1, u_2, \ldots, u_q\} \cap B_\alpha \neq \emptyset$. For the sake of a contradiction, assume that $\{u_1, u_2, \ldots, u_q\} \cap B_\alpha = \emptyset$. Then, some connected component G_1 of $G[V - B_\alpha]$ contains all of $v, u_1, u_2, \ldots, u_q$. Let Q be the set of all leaf descendants of p in T that are not contained in G_1. Since p is not pure and no leaf descendant of p in T is in B_α, we have $|Q| \geq 1$. Also, no vertex of G_1 can be adjacent to any vertex of Q in G. Now, by the choices of u_1 through u_q and the assumption that G_1 contains all of $u_1, u_2, \ldots, u_q$, we conclude that G has no edge between any vertex of Q and any vertex of $V - Q$. This contradicts the connectivity of G. So, the claim holds. By this claim, there is a $u_i \in \{u_1, u_2, \ldots, u_q\} \cap B_\alpha$ such that $d_T(u_i, v) \leq k$. Thus, $d_T(u_i, p) \leq k - 1$, establishing the fourth property. $\Box$

Each relaxed phylogeny R for B_α having the four properties in Lemma 3.1 is called a *skeleton* of α. The following lemma shows that there can be only a constant number of skeletons of α.

Lemma 3.2 *For each node α of $\mathcal{D}$, the number of skeletons of α is bounded from above by a constant depending only on k and $|B_\alpha|$.*

Proof. First note that the color and the threshold of each temporary leaf can be chosen from a constant range. Further note that each internal node v in a skeleton S of α satisfies $deg_S(v) = 3$. So, to prove the lemma, it suffices to prove that the number of temporary leaves in a skeleton S of α is bounded from above by a constant.

Consider a skeleton S of α and root S at an arbitrary final leaf r. We claim that for every temporary leaf u of S, there is a final leaf w with $d_S(u, w) \leq k$. To see this, let u be a temporary leaf and v be the parent of u in S. Since the root of S is a final leaf, v must be an internal node of S. If there is a final leaf w with $d_S(v, w) \leq k - 1$, then $d_S(u, w) \leq k$ and we are done. Otherwise, by the definition of a skeleton, at least one descendant x of v is a final leaf of S. Since $d_S(x, r) \leq k$, we have $\min\{d_S(x, v), d_S(r, v)\} \leq k - 1$ and hence $\min\{d_S(x, u), d_S(r, u)\} \leq k$. This establishes the claim.

Since each pair of final leaves are at distance at most k apart in S and the maximum degree of a node in S is 3, there are only a constant number of temporary leaves by the claim. $\Box$

By Lemma 3.2, while processing a node α of $\mathcal{D}$, our algorithm can find out all skeletons of α in constant time. For each skeleton S of α, if possible, the algorithm then extends S to a relaxed phylogeny for $U(\alpha, \beta)$ where β is the parent of α in rooted $\mathcal{D}$. The algorithm records these relaxed phylogenies of α in the dynamic programming table for later use when processing the parent β. The following definition aims at removing unnecessary relaxed phylogenies of α from the dynamic programming table.

Definition 3.2 *Let α and β be two adjacent nodes of $\mathcal{D}$. Let S be a skeleton of α. The* projection *of S to β is a relaxed phylogeny for $B_\alpha \cap B_\beta$ obtained from S by performing the following steps of operations:*

1. *Change each final leaf $v \notin B_\beta$ to a temporary leaf; Set the threshold of v to be* 0 *and set the color of v to be α.*

2. *Root S at an arbitrary vertex of $B_\alpha \cap B_\beta$.*
3. *Find those nodes v in S such that (i) every leaf descendant of v in S is a temporary leaf whose color is not β, but (ii) the parent of v in S does not have property (i).*
4. *For each node v found in the last step, if v is a leaf in S then set the color of v to be α; Otherwise, perform the following steps of operations:*
   (a) *Set $m_v = \min_u\{t_S(u) + d_S(u, v)\}$ where u ranges over all leaf descendants of v in S.*
   (b) *Delete all descendants of v in S.*
   (c) *Set v to be a temporary leaf, α to be the color of v, and m_v to be the threshold of v.*
5. *Unroot S.*

Obviously, two different skeletons of α may have the same projection to β. For convenience, we say that these skeletons are *equivalent*. Among equivalent skeletons of α, our algorithm will extend only a hopeful one of them to a relaxed phylogeny for $U(\alpha, \beta)$ and record it in the dynamic programming table. This motivates the following definition:

Definition 3.3 *Let α and β be two adjacent nodes of $\mathcal{D}$. A* projection *of α to β is the projection of a skeleton of α to β. Let P be a projection of α to β. An* expansion *of P to $U(\alpha, \beta)$ is a relaxed phylogeny X for $U(\alpha, \beta)$ such that some subtree Y of X is isomorphic to P, and the bijection f from the node set of P to the node set of Y witnessing this isomorphism satisfies the following conditions:*

- *For every final leaf v of P, $f(v) = v$.*
- *For every temporary leaf v of P with $c_P(v) = \beta$, $f(v)$ is a temporary leaf of X with $c_X(f(v)) = c_P(v)$ and $t_X(f(v)) = t_P(v)$.*
- *Suppose that we root X at a vertex in $B_\alpha \cap B_\beta$. Then, for every temporary leaf v of P with $c_P(v) \neq \beta$ (hence $c_P(v) = \alpha$), all leaf descendants of $f(v)$ in X are final leaves and are contained in $U(\alpha, \beta) - B_\beta$, and the minimum distance between $f(v)$ and a leaf descendant of $f(v)$ in X equals to $t_P(v)$.*

Note that a projection of α to β may have no expansion to $U(\alpha, \beta)$. The following lemma shows that if G has a kth root phylogeny T, then some subtree of T is a projection of α to β and another subtree of T is its expansion to $U(\alpha, \beta)$.

Lemma 3.3 *Let α and β be two adjacent nodes in $\mathcal{D}$. Let T be a kth root phylogeny of G. Root T at an arbitrary leaf that is in B_α. Let R be the skeleton of α obtained from T as in Lemma 3.1. Let P be the projection of R to β. Define a β-*pure *node to be a node w of T such that all leaf descendants of w in T are contained in $U(\beta, \alpha) - B_\alpha$. Further define a β-*critical *node to be a β-pure node of T whose parent is not β-pure. Let X be the relaxed phylogeny for $U(\alpha, \beta)$ obtained from T by performing the following steps of operations:*

1. *For every β-critical node w of T, perform the following:*

   (a) Compute the minimum distance from w to a leaf descendant of w in T; Let i_w denote this distance. (Comment: *$i_w \leq k$ or else the leaf descendants of w in tree T would be unreachable from the outside in graph G.)*

   (b) Delete all descendants (excluding w, of course) of w, and assign the pair (β, i_w) to w.

2. Unroot T.

Then, X is an expansion of P to $U(\alpha, \beta)$.

Proof. Straightforward. □

By Lemma 3.3, whenever G has a kth root phylogeny, there is always a projection of α to β that has an expansion to $U(\alpha, \beta)$. While processing α, our algorithm will find out those projections that have expansions to $U(\alpha, \beta)$, and record the expansions in the dynamic programming table.

3.2 Details of Dynamic Programming for 3PRk

To solve the 3PRk problem for G, we perform a dynamic programming on the tree-decomposition $\mathcal{D}$ as follows. To simplify the description of the algorithm, we add a new node r to $\mathcal{D}$, connect r to an arbitrary leaf α of $\mathcal{D}$, and copy the bag at α to r (that is, $B_r = B_\alpha$). Clearly, the resultant $\mathcal{D}$ is still a required tree-decomposition of G. Root $\mathcal{D}$ at r.

The dynamic programming starts at the leaves of $\mathcal{D}$, and proceeds upwards; After the unique child of the root r of $\mathcal{D}$ is processed, we will know whether G has a kth root phylogeny or not. The invariant maintained during the dynamic programming is that after each non-root node α has been processed, for each projection P of α to its parent β, we will have found out whether P has an expansion X to $U(\alpha, \beta)$, and will have found and recorded such an X if any.

Now consider how a non-root node α of $\mathcal{D}$ is processed. Let β be the parent of α in $\mathcal{D}$. First suppose that α is a leaf of $\mathcal{D}$. When processing α, we find and record all possible projections of α to β; Moreover, for each projection P found, we also record a skeleton S of α such that P is the projection of S to β.

Next suppose that α is neither a leaf nor the root node of $\mathcal{D}$, and suppose that all descendants of α in $\mathcal{D}$ have been processed. To process α, we try all possible skeletons S of α. When trying S, for each child γ of α in $\mathcal{D}$, we first compute the projection P_γ of S to γ, and then check whether P_γ is also a projection of γ to α and additionally has an expansion to $U(\gamma, \alpha)$. If the checking fails for at least one child of α, we proceed to try the next possible skeleton of α. Otherwise, we can conclude that the projection P_β of S to β has an expansion to $U(\alpha, \beta)$ because such an expansion can be constructed as follows:

1. For each child γ of α in $\mathcal{D}$, search the dynamic programming table to find the expansion X_γ of P_γ to $U(\gamma, \alpha)$, and find the bijection f_γ (from the node set of P_γ to the node set of some subtree of X_γ) witnessing that X_γ is an expansion of P_γ to $U(\gamma, \alpha)$.
   (*Comment:* To speed up the algorithm, we may have recorded this bijection in the dynamic programming table when processing γ.)

2. For each child γ of α in $\mathcal{D}$, root X_γ at an arbitrary vertex of $B_\gamma \cap B_\alpha$.
3. Modify S as follows: For each temporary leaf v of S with $c_S(v) \neq \beta$, replace v by the subtree rooted at $f_\gamma(v)$ of X_γ, where $\gamma = c_S(v)$.
   (*Comment*: Recall that by Definition 3.2, each temporary leaf v of S with $c_S(v) = \gamma$ is also a temporary leaf of P_γ.)

One can verify that the above construction indeed gives us an expansion of P_β. Since P_β is a possible projection of α to β, we record this expansion for P_β in the dynamic programming table. After trying all possible skeletons of α, if we find no projection of α to β that has an expansion to $U(\alpha, \beta)$, then we can conclude that G has no kth root phylogeny; otherwise, we proceed to the processing of the next node of $\mathcal{D}$.

Finally, suppose that α is the unique child of the root r of $\mathcal{D}$. Further suppose that α has been successfully processed (for otherwise we already knew that G has no kth root phylogeny). Then, by searching the dynamic programming table, we try to find a projection P of α to r such that (i) P has no temporary leaf whose color is r, and (ii) an expansion X of P to $U(\alpha, r)$ has been recorded in the dynamic programming table. If P is found, we can conclude that X is a kth root phylogeny for G; otherwise, we can conclude that G has no kth root phylogeny.

The above discussion justifies the following theorem:

Theorem 3.4 *Let k be a constant integer larger than or equal to 2. There is a linear-time algorithm determining if a given connected graph has a kth root phylogeny in which every internal node has degree 3, and if so, demonstrating one such phylogeny.*

We can easily generalize the above discussion to prove the following:

Corollary 3.5 *Let Δ and k be constant integers such that $\Delta \geq 3$ and $k \geq 2$. There is a linear-time algorithm determining if a given connected graph has a kth root phylogeny in which every internal node has degree in the range $[3, \Delta]$, and if so, demonstrating one such phylogeny.*

4 The Hardness of Closest Phylogenetic Root Problems

We introduce some basic concepts (some of them from [4]) that will be used in the hardness proofs. Consider a set $S = \{s_1, s_2, \ldots, s_n\}$. Let M be a symmetric matrix with rows and columns indexed by the elements of S. M is a *binary dissimilarity matrix* on set S if $M(s_i, s_j) \in \{1, 2\}$ for every pair (s_i, s_j) of distinct elements of S and $M(s_i, s_i) = 0$ for every element $s_i \in S$.

A tree T is a 2-*ultrametric* on set S if T is a *rooted* tree whose leaves are labeled by the elements of S and each leaf-to-root path contains exactly two edges. Call a node in T that is neither a leaf nor the root a *middle* node, to avoid ambiguity. The *half-distance* between two leaves s_i and s_j, denoted by $h_T(s_i, s_j)$, is one half of the number of edges on the unique path in T connecting s_i and s_j. Clearly, $h_T(s_i, s_j) \in \{1, 2\}$ if $i \neq j$, and $h_T(s_i, s_i) = 0$ for every i.

Given a binary dissimilarity matrix M and a 2-ultrametric T on set S, define

$$D(T,M) = \sum_{i<j} |h_T(s_i,s_j) - M(s_i,s_j)|,$$

which measures how well T matches the inter-leaf (half-)distances specified by M. [2] The following Fitting Ultrametric Trees (FUT) problem has been shown to be NP-complete by Křivánek and Morávek [6].

> Fitting Ultrametric Trees (FUT):
> Given a binary dissimilarity matrix M on set S and a nonnegative integer ℓ, decide if there is a 2-ultrametric T on S such that $D(T,M) \leq \ell$.

Kearney and Corneil [4] proved that CTRk is NP-hard when $k \geq 3$ by a (polynomial-time) reduction from FUT (to CTR3). Using a more dextrous reduction, Jiang, Lin, and Xu [3] have recently proved that CTR2 is intractable too.

4.1 CPR2

Given a binary dissimilarity matrix M on a set $S = \{s_1, \ldots, s_n\}$, let $S' = \{s_{n+1}, s_{n+2}, \ldots, s_{2n}\}$ be another set of n elements. Define a binary dissimilarity matrix M' on set $S \cup S'$, from M as follows:

For every pair of (not necessarily distinct) integers $i, j \in \{1, 2, \ldots, n\}$,

- $M'(s_i, s_j) = M(s_i, s_j)$;
- $M'(s_{n+i}, s_{n+j}) = M(s_i, s_j)$;
- $M'(s_i, s_{n+j}) = M(s_i, s_j)$, if $i \neq j$;
- $M'(s_i, s_{n+i}) = 1$.

Lemma 4.1 *If there is a 2-ultrametric T on set S such that $D(T,M) = \ell$, then there is a 2-ultrametric T' on set $S \cup S'$ such that $D(T', M') = 4\ell$.*

Proof. Given a 2-ultrametric T on set S such that $D(T,M) = \ell$, construct a 2-ultrametric T' on set $S \cup S'$ in the following way: The root and middle nodes of T' are the same as those in T; If an $s_i \in S$ is adjacent to a middle node u in T, then both s_i and s_{n+i} are adjacent to u in T'. Clearly, for every pair of (not necessarily distinct) integers $i, j \in \{1, 2, \ldots, n\}$,

- $h_{T'}(s_i, s_j) = h_T(s_i, s_j)$;
- $h_{T'}(s_{n+i}, s_{n+j}) = h_T(s_i, s_j)$;
- $h_{T'}(s_i, s_{n+j}) = h_T(s_i, s_j)$, if $i \neq j$;
- $h_{T'}(s_i, s_{n+i}) = 1$.

[2] So, here the entries in M are supposed to represent the half-distances between species instead of full distances.

Thus, $D(T', M') = 4D(T, M) = 4\ell$. □

Lemma 4.2 *Let T be a 2-ultrametric on set $S \cup S'$, then there is another 2-ultrametric T' on set $S \cup S'$ such that (1) $D(T', M') \leq D(T, M')$ and (2) for every $i \in \{1, 2, \ldots, n\}$, $s_i \in S$ and $s_{n+i} \in S'$ are adjacent to a common middle node in tree T'.*

Proof. Suppose that $s_i \in S$ is adjacent to a middle node u and $s_{n+i} \in S'$ is adjacent to another middle node $u' \neq u$ in tree T. Let

- $\widetilde{S} = (S \cup S') - \{s_i, s_{n+i}\}$;
- a be the number of $s \in \widetilde{S}$ adjacent to u in T with $M'(s, s_i) = 1$;
- b be the number of $s \in \widetilde{S}$ adjacent to u in T with $M'(s, s_i) = 2$;
- a' be the number of $s \in \widetilde{S}$ adjacent to u' in T with $M'(s, s_{n+i}) = 1$; and
- b' be the number of $s \in \widetilde{S}$ adjacent to u' in T with $M'(s, s_{n+i}) = 2$.

If $a' + b \leq a + b'$, we can modify T by deleting edge (s_{n+i}, u') and adding edge (s_{n+i}, u). Otherwise, we can modify T by deleting edge (s_{n+i}, u) and adding edge (s_{n+i}, u'). In either case, $D(T, M')$ does not increase and, s_i and s_{n+i} are adjacent to a common middle node of the modified T. Repeating this process results in a desired 2-ultrametric. □

From now on, we will only consider those 2-ultrametrics on set $S \cup S'$ such that for every $i \in \{1, \ldots, n\}$, $s_i \in S$ and $s_{n+i} \in S'$ are connected to a common middle node.

Lemma 4.3 *Given a binary dissimilarity matrix M on set S, there is a 2-ultrametric T on S such that $D(T, M) \leq \ell$ if and only if there is a 2-ultrametric T' on $S \cup S'$ such that $D(T', M') \leq 4\ell$.*

Proof. The "only if" part is implied by Lemmas 4.1 and 4.2. The "if" is straightforward by observing that deleting elements in S' from T' gives a 2-ultrametric T on set S such that $D(T, M) = D(T', M')/4$. □

Theorem 4.4 CPR2 *is NP-complete.*

Proof. CPR2 is clearly in NP. The hardness proof is done by a reduction from FUT. Consider an instance I of FUT, *i.e.* a dissimilarity matrix M on set $S = \{s_1, s_2, \ldots, s_n\}$ and a nonnegative integer ℓ. Without loss of generality, we may assume that $n \geq 4$ and $\ell \leq n(n-1)/2$. Construct the corresponding set S' and the dissimilarity matrix M' on $S \cup S'$, from M as in the above. Construct a graph G on a set V of $2n$ vertices as follows. For every $s_i \in S \cup S'$, there is a corresponding vertex v_i in V. [3] Two distinct vertices v_i and v_j are adjacent in G if and only if $M'(s_i, s_j) = 1$.

[3] We use a different name v_i instead of s_i here in order to avoid ambiguity, although they should be viewed as identical.

Let the instance of CPR2 consist of graph G and a nonnegative integer $\ell' = 4\ell$. We claim that there is an approximate phylogenetic square root T' of graph G with $|E(G) \oplus E(T'^2)| \le \ell'$ if and only if there is a 2-ultrametric T on set S with $D(T, M) \le \ell$.

To see the "if" part, suppose that there is a 2-ultrametric T on set S such that $D(T, M) \le \ell$. This implies there is a 2-ultrametric T'' on set $S \cup S'$ such that $D(T'', M') \le 4\ell = \ell'$. Recall that for all $i \in \{1, 2, \ldots, n\}$, $s_i \in S$ and $s_{n+i} \in S'$ are adjacent to a common middle node in tree T''. It follows that every middle node in T'' has degree at least 3. Then, replacing every leaf s_i in T'' by vertex v_i gives a tree (still denoted by T'') whose leaves are the elements of V. So, if there are three or more middle nodes, then T'' is a phylogeny on V and we are done by setting $T' = T''$. If there is only one middle node in T'', then we obtain T' from T'' by deleting the root as well as its incident edge. If there are exactly two middle nodes in T'', then we obtain T' from T'' by removing the root and connecting the two middle nodes by an edge. Clearly, the final tree T' is a phylogeny on set V. Moreover, $d_{T'}(v_i, v_j) \le 2$ if and only if $h_{T''}(v_i, v_j) = h_{T''}(s_i, s_j) = 1$, for all distinct $i, j \in \{1, 2, \ldots, n\}$. It follows that edge (v_i, v_j) is in exactly one of $E(G)$ and $E(T'^2)$ if and only if either $h_{T''}(s_i, s_j) = 1$ and $M'(s_i, s_j) = 2$, or $h_{T''}(s_i, s_j) = 2$ and $M'(s_i, s_j) = 1$. That is, the number of such edges is equal to $D(T'', M')$. Thus, $|E(G) \oplus E(T'^2)| = D(T'', M') \le \ell'$.

To prove the "only if" part, let us assume that T' is a phylogeny interconnecting the vertices in V such that $|E(G) \oplus E(T'^2)| \le \ell'$. If T' contains only one internal node, *i.e.* T'^2 is complete, then a 2-ultrametric T'' on set $S \cup S'$ can be constructed to have only one middle node with all elements of $S \cup S'$ attached to it. So, suppose in the following that T' contains two or more internal nodes. We obtain a rooted tree T'' by modifying T' as follows:

1. Select an arbitrary internal edge of T', *i.e.* an edge connecting two internal nodes, and split the edge into two edges by adding a new internal node, say r, on the edge.
2. Root the new tree at r.
3. Delete all internal edges from the tree. This results in a (possibly empty) set of isolated nodes and a set of stars whose centers are internal nodes of the original T'.
4. Connect the centers of the stars to the root r.

Clearly, the leaves of T'' are the elements of V, T'' is of height 2 and every leaf-to-root path is of length exactly 2. Furthermore, $E(T'^2) = E(T''^2)$. Now replacing leaf v_i in T'' by s_i gives a 2-ultrametric (still denoted by T'') on set $S \cup S'$. Again, an edge (v_i, v_j) is in exactly one of $E(G)$ and $E(T'^2)$ if and only if either $h_{T''}(s_i, s_j) = 1$ and $M'(s_i, s_j) = 2$, or $h_{T''}(s_i, s_j) = 2$ and $M'(s_i, s_j) = 1$. Thus, $D(T'', M') = |E(G) \oplus E(T'^2)| \le \ell' = 4\ell$. According to Lemmas 4.2 and 4.3, we may easily obtain a 2-ultrametric T on set S, from T'', such that $D(T, M) \le \ell$. This finishes the proof. □

4.2 CPRk

We extend the above NP-completeness result to CPRk, for $k > 2$. In doing so, we need to design several gadgets that facilitate the proof.

4.2.1 Gadgets

A *critical clique* [7] of a graph is a maximal subset of vertices that are adjacent to each other and have a common neighborhood. As for constructing a phylogenetic root, the vertices in a critical clique can be identified because they are interchangable in every phylogenetic root. When we say that one critical clique C_1 is adjacent to another C_2, we mean every vertex in C_1 is adjacent to every vertex in C_2. Consider a graph $H = (V, E)$ consisting of $4k-7$ critical cliques C_1 through C_{4k-7} such that

- $|C_i| = N$, for $1 \le i \le 4k-7$;
- C_i is adjacent to C_1, C_2, …, C_{k-2+i}, for $1 \le i \le k-1$;
- C_i is adjacent to C_{i-k+2}, C_{i-k+3}, …, C_{i+k-2}, for $k \le i \le 3k-5$;
- C_i is adjacent to C_{i-k+2}, C_{i-k+3}, …, C_{4k-7}, for $3k-4 \le i \le 4k-7$.

Assume that T is an approximate kth root phylogeny of graph H such that $|E(T^k) \oplus E(H)| \le \ell < N$. We present some enforced structural properties of T below.

Lemma 4.5 *In T, no two vertices from different critical cliques can be adjacent to a common internal node.*

Proof. For the sake of a contradiction, assume that $v_i \in C_i$ and $v_j \in C_j$ with $i \ne j$ are both adjacent to an internal node u in T. Then, v_i and v_j have the same neighborhood in the kth phylogenetic power T^k. On the other hand, by the construction of H, there is another critical clique C_h with $h \notin \{i, j\}$ such that C_h is adjacent to exactly one of C_i and C_j in H. So, $|E(T^k) \oplus E(H)|$ is at least as large as N, a contradiction. $\square$

Let R be the skeleton obtained from T by deleting all the leaves. In light of Lemma 4.5, a node of R adjacent to some vertex of C_i in tree T is called a *node for C_i* or simply a *C_i-node*. We also call a vertex in clique C_i a *C_i-vertex*. Let R_i be the minimal subtree of R that contains all C_i-nodes. Clearly, R_i can be obtained from R by deleting some nodes and their incident edges. Call R_i the *C_i-subtree*.

Lemma 4.6 *For every $j \in \{k-1, k, \ldots, 3k-5\}$ and every $i \ne j$, there is no C_i-node in R_j. That is, for every $j \in \{k-1, k, \ldots, 3k-5\}$, the C_j-nodes form a subtree of T.*

Proof. Fix a $j \in \{k-1, k, \ldots, 3k-5\}$. Notice that for every $i \in \{1, 2, \ldots, j-1\}$, if there is a C_i-node, say t_i, in R_j, then t_i must be an internal node in R_j. In tree T, every vertex in C_{j+k-2} is either at distance greater than $k-2$ from some C_j-node or at distance at most $k-2$ from every C_j-node. A C_{j+k-2}-vertex which is at distance at most $k-2$ from every C_j-node is at distance less than $k-2$ from t_i. Therefore, in T^k, each C_{j+k-2}-vertex is either adjacent to no C_j-vertex, or adjacent to some C_i-vertex (which is adjacent to t_i in T). This together with the fact that C_{j+k-2} is adjacent to C_j but adjacent to none of C_1 through C_{j-1} implies that $|E(T^k) \oplus E(H)| \ge N$, a contradiction. Similarly, using C_{j-k+2} instead of C_{j+k-2}, we can show that R_j contains no C_i-node for every $i \geq j+1$. This proves the lemma. $\square$

Lemma 4.7 *For every $i \in \{k-1, k, \ldots, 3k-5\}$, there is exactly one C_i-node, denoted by t_i, in tree T. Moreover, the nodes $t_{k-1}, t_k, \ldots, t_{3k-5}$ appear consecutively in this order on a path of tree T.*

Proof. Let t_{2k-2} be a C_{2k-2}-node. If t_{2k-2} were within distance $k-2$ from all C_{k-1}-nodes, then the vertices of C_{2k-2} adjacent to t_{2k-2} in tree T would be at distance at most k from every vertex of C_{k-1} in tree T. This would result in at least N edges in $E(T^k) - E(H)$, and is thus impossible. Let t_{k-1} be a C_{k-1}-node such that $d_R(t_{k-1}, t_{2k-2}) > k-2$, and let P denote the path in R connecting them. It holds that for every $i \in \{k, k+1, \ldots, 2k-3\}$, there must be a C_i-node, say t_i, such that $d_R(t_i, t_{k-1}) \leq k-2$ and $d_R(t_i, t_{2k-2}) \leq k-2$ (otherwise $|E(T^k) \oplus E(H)|$ would be at least N because both C_{k-1} and C_{2k-2} are adjacent to C_i in H). Therefore, in tree R, t_i is closer to some node on path P than to either of t_{k-1} and t_{2k-2}. If subtree R_i contains some node from the path P, then we choose an arbitrary node in $R_i \cap P$ as the *representative* node for R_i. Otherwise, choose the node on P that is the closest to subtree R_i as the representative node for R_i. Denote the representative node for R_i by r_i.

We claim that $r_i \neq r_j$ for every pair of distinct $i, j \in \{k, k+1, \ldots, 2k-3\}$. To prove the claim, for every $i \in \{k, k+1, \ldots, 2k-3\}$, let t'_i be the closest C_i-node to r_i in R. Then, $d_R(r_i, t'_i)$ should be less than or equal to both $d_R(r_i, t_{k-1})$ and $d_R(r_i, t_{2k-2})$. It follows that if R_i and R_j with $i < j$ share some representative node say $r_i = r = r_j$ and $d_R(r, t'_i) \leq d_R(r, t'_j)$ (respectively, $d_R(r, t'_i) \geq d_R(r, t'_j)$), then for every vertex $x \in C_{j+k-2}$ (respectively, $x \in C_{i-k+2}$), either $d_T(x, t'_i) \leq k-1$ or $\max\{d_T(x, t_{2k-2}), d_T(x, t'_j)\} > k-1$ (respectively, either $d_T(x, t'_j) \leq k-1$ or $\max\{d_T(x, t_{k-1}), d_T(x, t'_i)\} > k-1$). This again contradicts the fact that $|E(T^k) \oplus E(H)| < N$.

A similar argument to the proof of the above claim shows that $k \leq i < j \leq 2k-3$ if and only if $d_R(t_{k-1}, r_i) < d_R(t_{k-1}, r_j)$. Since some C_k-node should be within distance $k-2$ from t_{2k-2}, we conclude that the representative node r_k is in fact a C_k-node. Analogously, for every $i \in \{k, k+1, \ldots, 2k-3\}$, r_i is in fact a C_i-node. It further follows that $d_R(t_{k-1}, r_k) = d_R(r_k, r_{k+1}) = \ldots = d_R(r_{2k-3}, t_{2k-2}) = 1$, that is, the path P connecting t_{k-1} and t_{2k-2} is t_{k-1}-r_k-r_{k+1}-…-r_{2k-3}-t_{2k-2}. It is then easy to argue that there is only one C_i-node, for every $i \in \{k-1, k, \ldots, 2k-3\}$.

The lemma is proved by considering analogously the C_{2k-4}-node and a C_{3k-5}-node, and the index interval $[2k-4, 3k-5]$. $\square$

By Lemma 4.7, letting t_i denote the unique C_i-node, for $i \in \{k-1, k, \ldots, 3k-5\}$, we get a rough structure of R, as shown in Figure 1 (where $k = 5$ and t_i indicates a possible C_i-node, for $i \in \{1, 2, \ldots, k-2\} \cup \{3k-4, 3k-3, \ldots, 4k-7\}$).

4.2.2 Proof of Hardness

Theorem 4.8 CPR*k is NP-complete, when $k \geq 3$.*

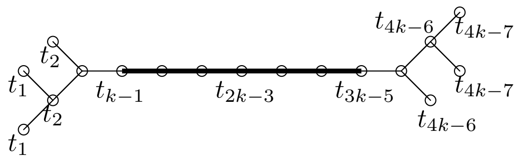

Figure 1: The rough structure of R.

Proof. The proof is also a reduction from FUT, but it is more complicated than that of Theorem 4.4. To simplify the presentation, we assume that k is even. It is trivial to extend the proof to odd k. Given a binary dissimilarity matrix M on a set $S = \{s_1, \ldots, s_n\}$, let $S_h = \{s_{hn+1}, s_{hn+2}, \ldots, s_{(h+1)n}\}$ be another set of n elements for all $h = 1, 2, \ldots, 2^{k/2} - 1$. For convenience, let S_0 denote the set S. Define a dissimilarity matrix M' on set $\widetilde{S} = \bigcup_{h=0}^{2^{k/2}-1} S_h$, from M as follows. For every pair of integers $i, j \in \{1, 2, \ldots, n\}$ and every pair of integers $h_1, h_2 \in \{0, 1, \ldots, 2^{k/2} - 1\}$,

- $M'(s_{h_1n+i}, s_{h_2n+j}) = M(s_i, s_j)$, if $i \neq j$;
- $M'(s_{h_1n+i}, s_{h_2n+i}) = 1$, if $h_1 \neq h_2$;
- $M'(s_{h_1n+i}, s_{h_1n+i}) = 0$.

Similarly to the proofs of Lemmas 4.1, 4.2 and 4.3, we can show the following:

(i) There exists a 2-ultrametric T on set S with $D(T, M) \leq \ell$ if and only if there exists a 2-ultrametric T' on set $\widetilde{S}$ with $D(T', M') \leq 2^k\ell$.

(ii) We can only consider those 2-ultrametrics on set $\widetilde{S}$ in which s_{hn+i}, $h = 0, 1, \ldots, 2^{k/2} - 1$, are adjacent to a common middle node.

Let $\ell' = 2^k\ell$ and $N = \ell' + 1$. The instance of CPRk consists of the nonnegative integer ℓ' and graph G constructed as follows:

- G takes $\widetilde{S}$ as a vertex subset;
- Two distinct vertices $s_i, s_j \in \widetilde{S}$ are adjacent in G if and only if $M'(s_i, s_j) = 1$;
- G also contains the graph H consisting of the $4k - 7$ critical cliques as discussed in the last subsubsection as a subgraph; and
- For every $i \in \{\frac{3k}{2} - 1, \frac{3k}{2}, \ldots, \frac{5k}{2} - 5\}$, each vertex in C_i is adjacent to every vertex in $\widetilde{S}$.

Lemma 4.7 guarantees that if G has an approximate phylogeny T' such that $|E(G) \oplus E(T'^k)| \leq \ell' < N$, then the vertices in each critical clique C_i in subgraph H (where $i \in \{k-1, k, \ldots, 3k-5\}$) are adjacent to the unique internal C_i-node t_i in T'. Since every vertex in $\widetilde{S}$ is adjacent to all vertices in C_i, for all $i \in \{\frac{3k}{2} - 1, \frac{3k}{2}, \ldots, \frac{5k}{2} - 5\}$, and the length of the path consisting of those $k - 3$ C_i-nodes has length $k - 4$, every vertex in $\widetilde{S}$ must be within distance $\frac{k}{2} + 1$ from the (unique) C_{2k-3}-node t_{2k-3} in T'.

Let T'' denote the minimal subtree of T' that contains all vertices in $\widetilde{S}$. The first observation is that T'' contains no vertex outside $\widetilde{S}$. Secondly, since every vertex in $\widetilde{S}$ is adjacent to neither vertex in $C_{\frac{3k}{2}-2}$ nor vertex in $C_{\frac{5k}{2}-4}$, we conclude that every vertex in $\widetilde{S}$ is at distance exactly $\frac{k}{2}+1$ from the C_{2k-3}-node t_{2k-3}. Furthermore, the path connecting any vertex in $\widetilde{S}$ to t_{2k-3} does not intersect the backbone path formed by the C_i-nodes for $i \in \{k-1, k, \ldots, 3k-5\}$ (except at t_{2k-3}).

Therefore, if subtree T'' does not include node t_{2k-3}, then we can construct a 2-ultrametric tree, denoted also by T'', on set $\widetilde{S}$ by connecting all the elements to a single middle node. Otherwise, rooting T'' at t_{2k-3} and letting every leaf be adjacent to its closest child node of the root, which serves as the middle node, give a 2-ultrametric, denoted still by T'', on set $\widetilde{S}$. In any case, the 2-ultrametric T'' on set $\widetilde{S}$ satisfies $D(T'', M') \leq \ell' = 2^k\ell$, which immediately implies that we can construct a 2-ultrametric T on set S such that $D(T, M) \leq \ell$.

On the other hand, if we have a 2-ultrametric T on set S such that $D(T, M) \leq \ell$, we can easily construct a 2-ultrametric T'' on set $\widetilde{S}$ such that $D(T'', M') \leq 2^k\ell = \ell'$ and in which elements s_{hn+i}, $h = 0, 1, \ldots, 2^{k/2} - 1$, are adjacent to a common middle node. It is also easy to transform the subtree of T'' under each middle node into a $\frac{k}{2}$-height subtree rooted at the middle node so that every leaf is at distance exactly $\frac{k}{2}$ from the middle node and every internal node of the whole tree (except its root), still denoted by T'', has degree at least 3. At the same time, we can also easily build a tree for subgraph H in which all vertices in C_i are adjacent to a single C_i-node t_i and these $4k-7$ C_i-nodes are connected consecutively into a path (such that t_i is adjacent to t_{i-1} and t_{i+1}). We then identify the root of T'' with the C_{2k-3}-node t_{2k-3}. This gives a phylogeny, which is denoted by T', such that $|E(G) \oplus E(T'^k)| = D(T'', M') \leq \ell'$. □

4.3 Steiner kth Root Problems

We study another problem closely related to PRk and TRk, which is the Steiner kth Root Problem [7]. Recall that given a graph $G = (V, E)$, TRk asks for a tree whose node set is exactly V and PRk asks for a tree whose leaf-set is exactly V. A more general problem is to ask for a tree T whose node set is a superset of V and whose leaf-set is a subset of V, and such that for every pair of vertices u and v in V, $d_T(u, v) \leq k$ if and only if $(u, v) \in E$. We call a tree T, whose node set is a superset of V and whose leaf-set is a subset of V, a *Steiner tree* on V.

Steiner kth Root Problem (SRk):
Given a graph $G = (V, E)$, find a Steiner tree T on V such that for each pair of vertices $u, v \in V$, $(u, v) \in E$ if and only if $d_T(u, v) \leq k$.

Such a Steiner tree T (if exists) is called a *Steiner* kth *root* or a kth *root Steiner tree* of G. G is called the kth *Steiner power* of T. We also abuse T^k to denote the kth *Steiner power* of T, when there is no confusion from the context.

Notice that we do not require here a non-leaf node in a Steiner tree to have degree at least 3. This requirement is not necessary from the tree root point of view. But, one may do so as this requirement is natural from the phylogenetic root point of view. Steiner trees satisfying this additional requirements are called *restricted Steiner trees.* Graphs having a restricted Steiner kth root, for $k = 1, 2$, can be recognized in linear time [7]. The recognition algorithm can be extended to find an ordinary Steiner kth root, for $k = 1$ and $k = 2$. However, when $k \geq 3$, no polynomial-time recognition algorithm has been reported yet to find either a Steiner kth root or a restricted Steiner kth root. In the following, we will only consider ordinary Steiner roots and show that the closest Steiner kth root problem (CSRk), defined in a straightforward way, is NP-complete when $k \geq 2$.

We call the nodes in a Steiner tree T that are not vertices in V *Steiner nodes*.

For CSR1, we notice that deleting all Steiner nodes from an (approximate) 1st root Steiner tree T results in a collection of subtrees such that vertices in different subtrees are not adjacent in T^1. Therefore, for any input graph G, the best way to build the closest 1st root Steiner tree is to construct a spanning tree for each connected component in G and then connect these spanning trees together via a Steiner node. That is, a closest 1st root Steiner tree can be computed in $O(n)$ time, where n is the number of vertices in the input graph. The complexity changes when k marches from 1 to 2.

4.3.1 Gadgets

Suppose we have an instance of FUT, that is, a set S, a dissimilarity matrix M on S, and a nonnegative integer ℓ. Let $N = \ell + 2$. Gadget H_u, where $u \in S$, is a graph consisting of $N + 1$ cliques:

- $C_0 = \{u, u_1, \ldots, u_N\}$,
- $C_i = \{u, u_i, u_{i1}, u_{i2}, \ldots, u_{iN}\}$, $i = 1, 2, \ldots, N$.

Abusing H_u to denote its vertex set, then $H_u \cap H_v = \emptyset$ when $u \neq v$.

The instance of CSR2 corresponding to the instance of FUT consists of the nonnegative integer ℓ and a graph $G = (V, E)$ which is constructed as follows:

- For each element $u \in S$, there is a vertex u in G (thus, we do not distinguish them as being either an element or a vertex); These vertices are included in the vertex subset named F;
- For $u, v \in F$, $(u, v) \in G$ if and only if $M(u, v) = 1$;
- F includes one additional special vertex r, called *root*; r is adjacent to every other vertex $u \in F$;
- G includes gadgets H_u, for all $u \in F$, where vertex u in gadget H_u identifies vertex u in subset F.

It is easy to check that $|V| = (N^2 + N + 1)(n + 1)$, where $n = |S|$.

Lemma 4.9 *If graph G has an approximate square root Steiner tree T such that $|E(G) \oplus E(T^2)| \leq \ell$, then in every gadget H_u, each clique $C_i \in \{C_0, C_1, \ldots, C_N\}$ induces a subtree T_i of T with diameter* 2.

Proof. Otherwise there would be at least $N - 1 = \ell + 1$ edges in $E(G) - E(T^2)$. □

Let us focus on one gadget first, say H_u. Lemma 4.9 tells us that the subtree T_i of T induced by clique C_i in H_u is a star. Denote the center of T_i by c_i. Then, c_i cannot be a vertex in $C_j - C_i$ for any $j \neq i$. The reason is that otherwise there would be at least $N = \ell + 2$ edges in $E(T^2) - E(G)$. If c_i is a Steiner node, then it cannot be adjacent to any vertex in $C_j - C_i$ with $j \neq i$ either, for the same reason. Let us turn to check out how these $N + 1$ stars (induced by the $N + 1$ cliques in gadget H_u) are connected in T.

Lemma 4.10 *The subgraph of T induced by H_u is a subtree containing no Steiner node. Moreover, u is the center of the star T_0 induced by C_0, and $u_i \in C_0$ is the center of the star T_i induced by C_i, for $1 \leq i \leq N$.*

Proof. Observe that $C_0 \cap C_i = \{u, u_i\}$ for every $1 \leq i \leq N$, and $C_i \cap C_j = u$, for any pair $1 \leq i < j \leq N$. It follows that u cannot be the center of star T_i for any $1 \leq i \leq N$. Therefore, u is adjacent to center c_i of star T_i in T, for every $1 \leq i \leq N$. Furthermore, since u and u_i must present in both T_0 and T_i, u_i has to be the center of T_i and u has to be the center of T_0. *I.e.* $c_i = u_i$ for $0 \leq i \leq N$. This finishes the proof. □

For convenience, we call the subtree induced by H_u the *u-tree*, of which the topology is shown in Figure 2.

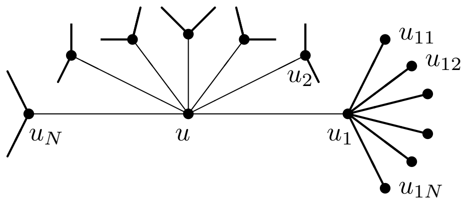

Figure 2: The topology of the u-trees.

Corollary 4.11 *In T, none of the vertices in gadget H_v, where $v \neq u$, could be adjacent to any vertex in clique C_0 in gadget H_u.*

Proof. Otherwise there would be at least N edges in $E(T^2) - E(G)$. □

4.3.2 Proof of Hardness

Theorem 4.12 CSR2 *is NP-complete.*

Proof. CSR2 is clearly in NP. The proof is again a reduction from FUT. Given an instance of FUT, we may construct the corresponding instance of CSR2 as in the above.

Trivially, if there is a 2-ultrametric T on set S such that $D(T, M) \leq \ell$, then

1. replacing every leaf in T by the corresponding vertex in F,
2. replacing the root of T by the root vertex r,
3. taking the middle nodes in T as Steiner nodes, and
4. expanding each vertex $u \in F$ into the u-tree

would turn T into a Steiner tree T' on the vertex set V of graph G. Notice that the u-tree, where $u \in F$, realizes all the edges in gadget H_u. All edges of form (r, u), where $u \in F$, in graph G are also realized since $d_{T'}(r, u) = 2$. Therefore, $|E(G) \oplus E(T'^2)| = D(T, M) \leq \ell$.

On the other hand, suppose that there is an approximate square root Steiner tree T' of G such that $|E(G) \oplus E(T'^2)| \leq \ell$. Then from the fact that $N = \ell + 2$, Lemmas 4.9 and 4.10, and Corollary 4.11, we know that those $N + 1$ u-trees, for all $u \in F$, are connected via either Steiner nodes or edges of form $(u_{i_1 j_1}, v_{i_2 j_2})$, where $u \neq v$ and $u, v \in F$. If vertex u and root vertex r are not adjacent to a common Steiner node, then the edge (r, u) wouldn't be realized in T'^2. Therefore, starting from u-trees that are the farthest from r in T', cutting them off and re-connecting each u to r via a Steiner node will realize edge (r, u) in T'^2. This process may un-realize some edge of form (v, u) (where v is a non-root vertex in F other than u), nonetheless, it does not affect any other edge. In other words, this process doesn't increase $|E(G) \oplus E(T'^2)|$. Thus we may assume without loss of generality that those $N + 1$ u-trees are connected to form into T' in a way that every non-root vertex $u \in F$ is at distance exactly 2 from root r and their common neighbor is a Steiner node.

If we delete all vertices in the gadget H_u except vertex u, for every vertex $u \in F$ (including the root vertex r), we will get a tree, denoted by T, in which every non-root vertex u in F is a leaf whose distance to r is exactly 2. Clearly, T is an approximate square root Steiner tree of the subgraph F (of graph G) and $|E(F) \oplus E(T^2)| = |E(G) \oplus E(T'^2)| \leq \ell$. Rooting T at r and replacing every leaf by the corresponding element in S, we obtain a 2-ultrametric, denoted still by T, on set S. Recall that there is an edge between vertices u and v if and only if $M(u, v) = 1$. $D(T, M) = |E(F) \oplus E(T^2)| \leq \ell$.

In summary, we have shown that there is a 2-ultrametric T on set S such that $D(T, M) \leq \ell$ if and only if there is an approximate Steiner tree T' on the vertex set of graph G such that $|E(G) \oplus E(T'^2)| \leq \ell$. This completes the proof of NP-completeness. □

By employing more gadgets, as in the proof of Theorem 4.8, we can also prove the following.

Corollary 4.13 CSRk *is NP-complete, when* $k > 2$.

5 Open problems

Since CPRk is NP-complete for all $k \geq 2$, it would be interesting to know how well we can approximate the closest phylogenetic kth root. Also, it would be nice to extend Theorem 3.4 and Corollary 3.5 to disconnected graphs.

Acknowledgments

GL would like to thank Paul Kearney for bringing CPRk to his attention, Bin Ma and Jinbo Xu for many helpful discussions.

References

[1] H. L. Bodlaender. Nc-algorithms for graphs with small treewidth. In *The 14th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 1988)*, LNCS 344, pages 1–10, 1989.

[2] A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph Classes: a Survey.* SIAM Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, 1999.

[3] T. Jiang, G.-H. Lin, and J. Xu. On the closest tree kth root problem. Manuscript, Department of Computer Science, University of Waterloo, November 2000.

[4] P. E. Kearney and D. G. Corneil. Tree powers. *Journal of Algorithms*, 29:111–131, 1998.

[5] T. Kloks. *Treewidth.* LNCS 842. Springer-Verlag, Berlin, 1994.

[6] M. Křivánek and J. Moránek. NP-hard problems in hierarchical-tree clustering. *Acta Informatica*, 23:311–323, 1986.

[7] G.-H. Lin, P. E. Kearney, and T. Jiang. Phylogenetic k-root and Steiner k-root. In *The 11th Annual International Symposium on Algorithms and Computation (ISAAC 2000)*, LNCS 1969, pages 539–551, 2000.

[8] Y.-L. Lin and S. S. Skiena. Algorithms for square roots of graphs. *SIAM Journal on Discrete Mathematics*, 8:99–118, 1995.

[9] R. Motwani and M. Sudan. Computing roots of graphs is hard. *Discrete Applied Mathematics*, 54:81–88, 1994.

[10] N. Nishimura, P. Ragde, and D. M. Thilikos. On graph powers for leaf-labeled trees. In *Proceedings of the 7th Scandinavian Workshop on Algorithm Theory (SWAT 2000)*, LNCS 1851, pages 125–138, 2000.

[11] D. L. Swofford, G. J. Olsen, P. J. Waddell, and D. M. Hillis. Phylogenetic inference. In D. M. Hillis, C. Moritz, and B. K. Mable, editors, *Molecular Systematics (2nd Edition)*, pages 407–514. Sinauer Associates, Sunderland, Massachusetts, 1996.