Strictly chordal graphs are leaf powers

William Kennedy\textsuperscript{a,1}, Guohui Lin\textsuperscript{b,\ast,2}, Guiying Yan\textsuperscript{c,3}

\textsuperscript{a} Department of Computing Science, University of Alberta, Edmonton, Alberta T6G 2E8, Canada
\textsuperscript{b} Bioinformatics Research Group, Department of Computing Science, University of Alberta, Edmonton, Alberta T6G 2E8, Canada
\textsuperscript{c} Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100080, China

Available online 5 July 2005

Abstract

A fundamental problem in computational biology is the phylogeny reconstruction for a set of specific organisms. One of the graph theoretical approaches is to construct a similarity graph on the set of organisms where adjacency indicates evolutionary closeness, and then to reconstruct a phylogeny by computing a tree interconnecting the organisms such that leaves in the tree are labeled by the organisms and every organism appears as a leaf in the tree. The similarity graph is simple and undirected. For any pair of adjacent organisms in the similarity graph, their distance in the output tree, which is measured by the number of edges on the path connecting them, must be less than some pre-specified bound. This is known as the problem of recognizing leaf powers and computing leaf roots. Graphs that are leaf powers are known to be chordal. It is shown in this paper that all strictly chordal graphs are leaf powers and a linear time algorithm is presented to compute a leaf root for any given strictly chordal graph. An intermediate root-and-power problem, the Steiner root problem, is also examined.

© 2005 Elsevier B.V. All rights reserved.

Keywords: Computational biology; Phylogeny reconstruction; Leaf root; Steiner root; Chordal; Strictly chordal

\ast Corresponding author.
E-mail addresses: kennedy@cs.ualberta.ca (W. Kennedy), ghlin@cs.ualberta.ca (G. Lin), yangy@mail.amt.ac.cn (G. Yan).

1 Supported partially by NSERC.
2 Supported by NSERC and CFI.
3 Supported partially by Chinese Academy of Sciences. Work done while visiting the Department of Computing Science, University of Alberta.

1570-8667/$ – see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.jda.2005.06.005
1. Introduction

A fundamental problem in computational biology is the reconstruction of the phylogeny, or evolutionary history tree, of a set of organisms [6] (the reader is referred to the journal *Evolution* through “http://www.sciencemag.org/cgi/collection/evolution” for more interesting discussions). In a phylogeny, each leaf is labeled by a distinct existing organism; the phylogeny is formed by positioning possible ancestors that might have led to this set of organisms. By viewing the correlations between leaves in a phylogeny as distances between vertices in a graph where adjacency describes the evolutionary closeness, the problem of forming a phylogeny can be cast as the problem of forming a tree from a graph. One such correlation between graphs and trees, or more generally between graphs and graphs, arises in the notion of graph powers.

Given a simple, undirected, and connected graph $G = (V, E)$, the $k$th power of $G$ is another simple graph on the same vertex set $V$ such that there is an edge between vertices $u$ and $v$ in the power graph if and only if a shortest path connecting $u$ and $v$ in $G$ has length at most $k$. The length of a path is measured by the number of edges on the path. The $k$th power of $G$ is denoted by $G^k$. Computing powers of a graph can be done straightforwardly and thus is an easy problem. However, the inverse problem, computing roots of a graph, turns out to be much more difficult. In fact, even recognizing if a graph is a square graph (of some other graph) had been proven to be hard [11]. An interesting special case is to recognize if a graph is a square of some tree. An $O(|V|^3)$ time algorithm has been developed in [10] to compute tree square roots of graphs. A generalization to compute $k$th tree roots of graphs, for any fixed $k$, has been done by Kearney and Corneil [7], which says that it can be determined in polynomial time whether a graph has a $k$th tree root or not, and if so, to construct such a tree root.

The notion of $k$th leaf power is introduced in [12,13], where a graph $G = (V, E)$ is the $k$th leaf power of a tree $T$ if vertices in $V$ one-to-one correspond to leaves in $T$ and there exists an edge in $E$ between vertices $u$ and $v$ if and only if the path connecting the leaves corresponding to $u$ and $v$ in $T$ is of length at most $k$. The problem of recognizing $k$th leaf powers is inspired by the problem of forming a phylogeny based on distance thresholds: Given a similarity graph $G = (V, E)$ in which there is an edge between each pair of organisms within certain evolutionary distance, the tree $T$ of which $G$ is the $k$th leaf power is a phylogeny in which the corresponding leaves are guaranteed to be at distance at most $k$ (measured by the number of edges on the path connecting them). It should be noted that in practice, phylogeny reconstruction problems involve edge-weighted trees where the weight measures the evolutionary distance.

A more restricted version of recognizing $k$th leaf powers is the recognition of $k$th phylogenetic powers, which is essentially the same as $k$th leaf powers except that the internal nodes in the desired phylogeny must have degree at least 3 whereby to represent the speciation events that are generally assumed to give to at least 2 new species. For simplicity, the internal nodes in the phylogenies, in either problem, are called Steiner nodes.

Given a (simple, undirected, connected) graph $G = (V, E)$ and a parameter $k$, to determine whether or not $G$ is the $k$th leaf power of some phylogeny $T$, and if so return a such phylogeny, is called the $k$th leaf root problem and denoted as $k$-LRP; To determine whether or not $G$ is the $k$th phylogenetic power of some phylogeny $T$, and if so return a
such phylogeny, is called the kth Phylogenetic Root problem and denoted as k-PRP. The k-LRP problem has been studied in [12,13] where numerous structural properties of a graph that is a kth leaf power are proven; Moreover, for k ≤ 4, O(|V|^3) time algorithms have been presented in [12,13] for k-LRP. The k-PRP problem has been studied in [9] where again numerous structural properties of a graph that is a kth phylogenetic power are proven and for k ≤ 4, O(|V| + |E|) time algorithms have been presented for k-PRP.

The main contribution of this paper is to take advantage of structural properties of graphs that are kth leaf powers, and to use the notion of critical clique defined in [9], to construct for any given strictly chordal graph a phylogeny that is a kth leaf root, to be defined in next section, for any k ≥ 4. Along the way, many structural properties are explored and construction techniques are developed, which are potentially useful for studies on general graphs. The rest of paper is organized as follows: In Section 2, we review some structural properties of graphs that are kth leaf powers. We also review some graph theoretical notions from [9]. Section 3 deals with an intermediate graph power problem called the kth Steiner Root problem, or k-SRP for short, and presents two linear time algorithms for constructing a kth Steiner root, if it exists, for a strictly chordal graph, for k ≥ 4 and k = 3 respectively. Section 4 discusses the k-LRP problem and concludes that a kth leaf root for a strictly chordal graph, for k ≥ 4, can always be constructed in linear time. Section 5 summarizes the results and points out some of our future work.

2. Preliminaries

We begin with some graph theoretical notions introduced in [9]. Given a graph G = (V, E), if it has a kth leaf root T, then vertices in V one-to-one correspond to leaves in T. Internal nodes in T are called Steiner nodes. Relaxing the constraint such that vertices in V could appear internally in the destination root, if such a root T exists, then G is the kth Steiner power of T and T is a kth Steiner root of G. Again, internal nodes in Steiner root T that are not vertices in V are called Steiner nodes. To determine whether or not G is a kth Steiner power, and if so return a kth Steiner root, is called the kth Steiner Root problem and denoted as k-SRP. It should be noted that in a kth leaf root or a kth Steiner root, the Steiner nodes have degree greater than or equal to 2.

Suppose tree T is a kth leaf/Steiner root of graph G = (V, E). Let s be a Steiner node in T. The number of Steiner nodes adjacent to s (excluding s) is the S-degree of s, and the number of vertices in V that are adjacent to s is the V-degree of s. We use d_T(u, v) to denote the distance between u and v in T, which is the length of the path connecting u and v in T; Likewise, d_G(u, v) denotes the distance between u and v in G, the length of a shortest path connecting u and v in G.

In graph G = (V, E), a maximal clique is a maximal subset of vertices that are pairwise adjacent; a maximal subset of vertices, which are pairwise adjacent and have a common set of neighbors outside of the subset, is called a critical clique [9]. As a concrete example, in the graph shown in Fig. 5(a), C_1 \cup C_2 is a maximal clique and both C_1 and C_2 are critical cliques. It is known that the collection of all maximal cliques doesn’t necessarily form a partition of vertex set V. Nonetheless, the collection of all critical cliques does form a partition of vertex set V. Look again at the graph in Fig. 5(a). Vertex set V contains 22
vertices and is partitioned into 13 critical cliques \((C_i, 1 \leq i \leq 13)\). For two critical cliques \(C_i\) and \(C_j\), if the vertices in \(C_i\) are neighbors of vertices in \(C_j\), then \(C_i\) and \(C_j\) are said adjacent. Using the fact that the collection of all critical cliques forms a partition of the vertex set and the definition of adjacency between critical cliques, a skeleton graph, called the Critical Clique Graph and denoted as \(CC(G)\), of \(G\) can be constructed by taking every critical clique of \(G\) as a node of \(CC(G)\), and two nodes in \(CC(G)\) are adjacent if and only if the corresponding critical cliques are adjacent [9]. Fig. 5(b) shows the critical clique graph for the graph in Fig. 5(a). To be distinguishable, we usually call a vertex in graph \(G\) a vertex, while a vertex in \(CC(G)\), which is a critical clique in \(G\), a node. Clearly, regarding \(CC(G)\) itself as a graph, every critical clique in \(CC(G)\) contains exactly one node. This is stated in the following lemma:

**Lemma 2.1.** For any graph \(G\), \(CC(G) = CC(CC(G))\).

**Definition 2.1.** [1] Let \(G = (V, E)\) be a graph and \((v_1, v_2, \ldots, v_\ell)\) be a simple cycle in \(G\), that is, \(v_1, v_2, \ldots, v_\ell\) are \(\ell\) distinct vertices, edge \((v_i, v_{i+1}) \in E\) for \(1 \leq i < \ell\), and edge \((v_\ell, v_1) \in E\).

A chord of the cycle is an edge in \(E\) between non-consecutive vertices in the cycle.

A graph \(G\) is chordal if each cycle in \(G\) of length at least 4 has at least one chord.

**Lemma 2.2.** [9,12,13] If graph \(G\) is connected and has a \(k\)th leaf root \((k \geq 3)\) or a \(k\)th Steiner root \((k \geq 1)\), then it must be chordal.

**Lemma 2.3.** There exists a linear time algorithm to recognize if graph \(G\) is chordal or not [1]; If graph \(G\) is chordal, its critical clique graph \(CC(G)\) can be constructed in linear time [9].

Hypergraphs can be regarded as generalizations of undirected graphs, where an edge of a hypergraph is not simply a pair but a non-empty subset of vertices, called a hyperedge. Formally, a hypergraph can be defined similarly by its vertex set \(V\) and its hyperedge set \(E = \{E_1, E_2, \ldots, E_m\}\) where \(E_i \subseteq V\) for every \(i\), denoted as \(H(V, E)\). Given a hypergraph \(H(V, E)\), a hyperedge \(E_i \in E\) is called a twig [5] if there exists another hyperedge \(E_b \in E\) such that

\[
E_i \cap \left( \bigcup_{E \in E \neq E_i} E \right) = E_i \cap E_b.
\]

Obviously, in an ordinary graph, a leaf edge, which is an edge incident with a leaf, is a twig if the graph is regarded as a hypergraph. In the definition of twig \(E_i\), hyperedge \(E_b\) is a branch for \(E_i\). One twig can have more than one branch. We will be using the notion of twig to define hypertrees. Before we move on, we want to remark that twig is very close to the notion of hinge [4,14] that is another important concept associated with hypergraphs, typically in the database applications. Given a hypergraph \(H(V, E)\), a join tree is a tree whose nodes are subsets of hyperedges in \(E\) such that whenever a vertex \(v \in V\) appears in two hyperedges \(E_1\) and \(E_2\), then \(E_1\) and \(E_2\) are connected (could be within a single node) and \(v\) appears in every node on the unique path connecting \(E_1\) and \(E_2\) in the join tree. If
every pair of adjacent nodes in the join tree share exactly one hyperedge, then the nodes, which are subsets of hyperedges, are called hinges, which together with the join tree form a hinge decomposition of hypergraph $\mathcal{H}$. Though closely related, one difference between twig and hinge is that not every hyperedge is a twig, similar to the notion of a leaf with respect to a given tree, but every node is a hinge in a hinge decomposition.

There are a few distinct but close definitions of hypertrees, or acyclic hypergraphs. Using the notion of twig, a hypergraph $\mathcal{H}(V, \mathcal{E})$ is a hypertree if its hyperedges can be ordered, say $(E_1, E_2, \ldots, E_m)$, such that $E_i$ is a twig in the sub-hypergraph $\mathcal{H}_i(V, \mathcal{E}_i)$, where $\mathcal{E}_i = \{E_1, E_2, \ldots, E_i\}$, for $i = 2, 3, \ldots, m$. Any such ordering is called a hypertree constructing ordering for $\mathcal{H}$ [5]. As a special case, the order of edges added by Prim’s algorithm to compute a minimum spanning tree is a hypertree constructing ordering for the Prim’s minimum spanning tree. This is the hypertree definition adopted in this paper. We note that in [1], a hypergraph $\mathcal{H}(V, \mathcal{E})$ becomes a hypertree if there is a tree $T$ on $V$ such that every hyperedge in $\mathcal{E}$ induces a subtree in $T$. In [3,14], associated with a hypergraph $\mathcal{H}(V, \mathcal{E})$ is a primal graph $G$ on $V$ in which two vertices are adjacent if they appear in a common hyperedge in $\mathcal{E}$. Hypergraph $\mathcal{H}$ is a hypertree if its primal graph $G$ is chordal.

We are interested in the connected hypertree, that is, for every pair of vertices $u$ and $v$, there is a sequence of hyperedges $E_{i_1}, E_{i_2}, \ldots, E_{i_k}$ such that $u \in E_{i_1}$, $v \in E_{i_k}$, and $E_{i_j} \cap E_{i_{j+1}} \neq \emptyset$ for $1 \leq j < k$. In this sense, $E_i \cap E_b \neq \emptyset$ for every twig $E_i$ and its associated branch $E_b$. Let $\mathcal{E}' = \{E_{i_1}, E_{i_2}, \ldots, E_{i_k}\}$ ($\ell \geq 2$) be a subset of hyperedges with non-empty intersection, that is, $I = \bigcap_{j=1}^\ell E_i_j \neq \emptyset$. $\mathcal{E}'$ is maximal if including any one more hyperedge will make the intersection an empty set. For simplicity, $I$ is called the intersection of $\mathcal{E}'$ (in fact, $I$ is the intersection of the hyperedges in $\mathcal{E}'$). $I$ is a strict intersection of $\mathcal{E}'$ if for every pair of hyperedges $E', E'' \in \mathcal{E}'$, $E' \cap E'' = I$, and for every other hyperedge $E''' \in \mathcal{E} - \mathcal{E}'$, $E''' \cap I = \emptyset$. A hypertree is strict if all its intersections are strict. As a special case, any tree is strict if it is regarded as a hypertree.

Let $G = (V, E)$ be a graph. Define the clique hypergraph of $G$, denoted as $\mathcal{H}(G)$, to be a hypergraph on $V$ with its hyperedge set being the collection of all maximal cliques of $G$.

**Definition 2.2.** A graph $G$ is strictly chordal if it is chordal and its clique hypergraph $\mathcal{H}(G)$ is a strict hypertree.

**Lemma 2.4.** For a chordal graph $G$, the clique hypergraph $\mathcal{H}(G)$ is a strict hypertree if and only if in the critical clique graph $CC(G)$, for every simple cycle, the nodes therein form a clique.

**Proof.** The “if” part is obvious from the definition of strict hypertree. To prove the “only if” part, suppose $\mathcal{H}(G)$ is a strict hypertree and assume to the contrary that $(C_1, C_2, \ldots, C_\ell)$, where $\ell \leq 4$, is a minimal simple cycle in $CC(G)$ such that $\{C_1, C_2, \ldots, C_\ell\}$ is not a clique. Without loss of generality, assume $C_1$ and $C_3$ are not adjacent. Let $K_1$ be a maximal clique in $G$ that includes $C_1$ and $C_2$, and $K_2$ be a maximal clique including $C_2$ and $C_3$ (cf. Fig. 1). Let $I = K_1 \cap K_2$, which includes $C_2$ at least. If $C_i \subset I$ for some $i \neq 2$, then we may conclude from the fact that $\mathcal{H}(G)$ is a strict hypertree that all the vertices in $C_2 \cup C_i$ have a common set of neighbors outside of $C_2 \cup C_i$. It follows that $C_2 \cup C_i$ must be a critical clique and thus contradicts the maximality in the definition of.
critical clique. Therefore, \( C_i \cap I = \emptyset \) for every \( i \neq 2 \). Let \( i \) be the largest index in the range \([3, \ell]\) such that \( C_i \subset K_2 \) (\( i = 5 \) in Fig. 1). Let \( K_3 \) be a maximal clique including \( C_i \) and \( C_{i+1} \) (define \( C_{\ell+1} = C_1 \)). Then, the intersection \( K_2 \cap K_3 \) includes \( C_i \), and it shouldn’t intersect \( K_1 \cap K_2 \). If \( K_3 \) includes \( C_1 \), we are done in obtaining a sequence of 3 maximal cliques; Otherwise we continue on to search for another maximal clique that intersects \( K_3 \) but the intersection doesn’t overlap \( K_2 \cap K_3 \), and so on. The result is a sequence of at least 3 maximal cliques \( K_1, K_2, \ldots, K_\ell \) such that every adjacent two maximal cliques in the sequence overlap while their intersections are distinct from each other. Suppose without loss of generality that \( K_\ell \) is the one among this sequence of maximal cliques that appears the last in a hypertree constructing ordering \((K_1, K_2, \ldots, K_m)\). Then we will find no branch for \( K_\ell \) since \( K_\ell \) intersects at least two other maximal cliques in the sub-ordering \((K_1, K_2, \ldots, K_\ell)\). This contradicts the assumption that the clique hypergraph \( \mathcal{H}(G) \) is a strict hypertree.

**Theorem 2.5.** There exists a linear time algorithm for recognizing whether or not a graph is strictly chordal, and if it is, returns its critical clique graph.

**Proof.** From Lemma 2.3, recognizing if a graph \( G \) is chordal can be done in linear time, and if so, to return its critical clique graph \( CC(G) \). To detect if there is a simple cycle in \( CC(G) \) such that the nodes on the cycle do not form a clique, we conduct the following: Pick any node in \( CC(G) \) as a root and start the breadth-first-search (BFS) [2]. It is well-known that for an undirected graph, there is no back edge with respect to the BFS tree. In other words, edges can be either tree edges or cross edges. If there is a cross edge connecting non-sibling nodes (i.e., these two nodes do not have the same parent node) in the BFS tree, then it indicates a cycle such that the nodes on the cycle do not form a clique in \( CC(G) \). For the set of all child nodes of any fixed node in the BFS tree, if its induced subgraph in \( CC(G) \) is not a collection of disjoint cliques, then there is a cycle whose nodes do not form a clique. In the other case, we claim that \( CC(G) \) does not contain any simple cycle whose nodes do not form a clique, that is, graph \( G \) is strictly chordal by Lemma 2.4. Note that BFS takes a linear time in the number of edges in \( CC(G) \) and checking for all child node lists again takes a linear time. Therefore, from the fact the number of edge in \( CC(G) \) is not greater than the number of edges in \( G \), we conclude that the overall recognition time is linear in the size of graph \( G \). 

![Fig. 1](image-url)
3. The Steiner root problem

In the $k$th Steiner Root problem ($k$-SRP), the input is a simple, undirected, connected graph $G = (V, E)$ where vertices represent organisms and edges indicate similarity, and the desired output is a Steiner tree $T$ on vertex set $V$ such that for every pair of vertices $u$ and $v$ in $V$, $(u, v) \in E$ if and only if $d_T(u, v) \leq k$. Note that vertices in $V$ could be internal in $T$. Tree $T$ might include nodes that are not vertices in $V$ and they are called Steiner nodes. When $k \leq 2$ and Steiner nodes in Steiner roots are required to have degree at least 3, to determine if $G$ is a $k$th Steiner power can be done in linear time [9]. It is not hard to see that those algorithms in [9] can be adapted for the case where Steiner nodes can have degree 2. We consider the latter case in this paper for $k \geq 3$. The main theorems in this section are: 1) If graph $G$ is strictly chordal, then it is a $k$th Steiner (and leaf) power for $k \geq 4$ and a $k$th Steiner root, which is also a $k$th leaf root, can be constructed in linear time; 2) If graph $G$ is strictly chordal, then it is not necessarily a 3rd Steiner power, but recognizing whether or not it is a 3rd Steiner power can be done in linear time, and if so, returning a 3rd Steiner root for $G$ at the same time.

Lemma 3.1. Suppose graph $G = (V, E)$ is a $k$th Steiner power and $C$ is a critical clique in $G$ of size $|C| > 2$. Then there is a $k$th Steiner root $T$ of $G$ such that vertices in $C$ appear as leaves in $T$ and they are adjacent to a common internal node, which could be a Steiner node or a vertex in $V - C$.

Proof. Suppose $T$ is a $k$th Steiner root of $G$. Let $T[C]$ denote the minimum subtree of $T$ that contains all vertices in $C$. Note from the minimality that all leaves in $T[C]$ must be vertices in $C$. Since $|C| > 2$, the diameter of $T[C]$, which is the maximum distance between a pair of vertices in $C$, must be at least 2. Suppose without loss of generality that the distance between $v_1$ and $v_2$ reaches the diameter. Let $u$ denote the node (which could be another vertex in $V$ or a Steiner node) that is adjacent to $v_1$ and it is on the $v_1$-to-$v_2$ path in $T$. If $u$ is a vertex in $C$, then place a Steiner node $s$ at the location and make $u$ a leaf attached to $s$; Otherwise, we don’t need to do the placement.

In either case, we will have a Steiner node or a vertex in $V - C$, denoted as $u$, adjacent to $v_1$ and $u$ is on the $v_1$-to-$v_2$ path. The next step is for every vertex $v \in C$ that is internal in $T$, place a distinct Steiner node at the location and make $v$ a leaf attached to Steiner node $u$, and for every vertex $v \in C$ that is already a leaf in $T$ but not attached to $u$, move it to be a leaf attached to $u$. This gives a new tree that is still a $k$th Steiner root, and thus proves the lemma. \[\Box\]

From Lemma 3.1, we can limit our efforts to search for a $k$th Steiner root for $G$, if it exists, in which for every critical clique of size greater than 2, the vertices therein appear as leaves in the root and they are adjacent to a common internal node. For simplicity, the internal node that vertices in such a critical clique are adjacent to is called the represent $\text{**ative}$ of the critical clique. The next two lemmas are straightforward.
Lemma 3.2. Suppose graph \( G = (V, E) \) is a \( k \)th Steiner power of \( T \), and \( C_1 \) and \( C_2 \) are two adjacent critical cliques in \( G \) both of size greater than 2. Then in \( T \) the distance between the representatives of \( C_1 \) and \( C_2 \) is at most \( k - 2 \).

Lemma 3.3. Suppose graph \( G = (V, E) \) is a \( k \)th Steiner power of \( T \), and \( C_1 \) and \( C_2 \) are two non-adjacent critical cliques in \( G \) both of size greater than 2. Then in \( T \) the distance between the representatives of \( C_1 \) and \( C_2 \) is at least \( k - 1 \).

Theorem 3.4. If graph \( G = (V, E) \) is strictly chordal, then it is a \( k \)th Steiner power for \( k \geq 4 \); Furthermore, a \( k \)th Steiner root, which is also a \( k \)th leaf root, can be computed in linear time.

Proof. Since graph \( G \) is strictly chordal, from the definition of strictly chordal and Lemma 2.4, we know that in \( CC(G) \), which can be constructed in linear time, two maximal cliques overlap by at most one node. Also it is true that there is no sequence of maximal cliques that form a simple cycle. Let \( K \) be a maximal clique in \( CC(G) \). If \( |K| = 2 \), then create a path to connect the two nodes such that on the path there are exactly \( k - 3 \) degree-2 Steiner nodes; If \( |K| \geq 3 \), then create a star to interconnect the nodes in \( K \) where the center of the star is a Steiner node of degree \( |K| \) and the path from the center to each node in \( K \) contains exactly \((\lfloor k/2 \rfloor - 1)\) degree-2 Steiner nodes. This process produces a Steiner tree \( T_0 \) on the node set of \( CC(G) \).

In tree \( T_0 \), it is guaranteed that adjacent nodes in \( CC(G) \) will be at distance at most \( k - 2 \), while non-adjacent nodes will be at distance at least \( 4 \times \lfloor k/2 \rfloor \geq 2k - 2 \geq k - 1 \), when \( k \geq 5 \). When \( k = 4 \), non-adjacent nodes will be at distance at least 4. It follows that, in tree \( T_0 \), replacing every node in \( CC(G) \) by a Steiner node and attaching vertices in the corresponding critical clique in \( G \) to that Steiner node, gives a \( k \)th Steiner root \( T \) for \( G \). The construction time is obviously linear.

Note that \( T \) is also a \( k \)th leaf root for \( G \), since every vertex in \( V \) appears as a leaf in \( T \). □

The construction process in the above proof of Theorem 3.4 fails for the case \( k = 3 \), at the place where \( |K| \geq 3 \). The rest of this section deals with this special case.

Lemma 3.5. Suppose graph \( G \) has a 3rd Steiner root \( T \). For two vertices \( v_1, v_2 \in V \), if \( d_G(v_1, v_2) = 2 \) (that is, \( v_1 \) and \( v_2 \) are non-adjacent but they have a common neighbor), then \( 4 \leq d_T(v_1, v_2) \leq 6 \).

Proof. The proof is trivial since there exists a vertex \( u \) that is adjacent to both \( v_1 \) and \( v_2 \) in \( G \) (i.e., \( u \) is a common neighbor), and therefore \( d_T(u, v_1) \leq 3 \) and \( d_T(u, v_2) \leq 3 \). □

Lemma 3.6. Suppose graph \( G \) has a 3rd Steiner root \( T \). For two vertices \( v_1, v_2 \in V \) such that \( d_G(v_1, v_2) = 2 \),

- if \( d_T(v_1, v_2) = 6 \), then \( |NG[v_1] \cap NG[v_2]| = 1 \);  
- if \( d_T(v_1, v_2) = 5 \), then \( |NG[v_1] \cap NG[v_2]| \leq 2 \);
• if $d_T(v_1, v_2) = 5$ and $|N_G[v_1] \cap N_G[v_2]| = 2$, then the two vertices in $N_G[v_1] \cap N_G[v_2]$ must be adjacent in both $G$ and $T$.

Proof. The proof is easy as we can see that because $5 \leq d_T(v_1, v_2) \leq 6$, only the center node(s) on the $v_1$-to-$v_2$ path in $T$ could be in $N_G[v_1] \cap N_G[v_2]$, which is non-empty. □

Lemma 3.7. Suppose graph $G$ has a 3rd Steiner root $T$. For two vertices $v_1, v_2 \in V$ such that $d_G(v_1, v_2) = 2$, if $|N_G[v_1] \cap N_G[v_2]| > 2$ then $d_T(v_1, v_2) = 4$.

Furthermore, all vertices in $N_G[v_1] \cap N_G[v_2]$ are adjacent to each other in $G$ and they must be adjacent to the center node on the $v_1$-to-$v_2$ path in $T$.

Proof. Since $|N_G[v_1] \cap N_G[v_2]| > 2$, it follows from Lemma 3.6 that $d_T(v_1, v_2) = 4$. Every common neighbor of $v_1$ and $v_2$ in graph $G$ must be at distance at most 3 from both $v_1$ and $v_2$ in $T$. It is trivial to see that if it is not adjacent to the center node on the $v_1$-to-$v_2$ path in $T$, then it must be farther than 3 away from either $v_1$ or $v_2$. This proves the lemma. □

Lemma 3.8. Suppose graph $G$ has a 3rd Steiner root $T$. Assume there exist in $G$ three maximal cliques $K_1, K_2, K_3$ such that $K_1 \cap K_2 = I_1 \neq \emptyset$, $K_2 \cap K_3 = I_3 \neq \emptyset$, and $K_1 \cap K_3 = \emptyset$. Let $I_2 = K_2 - I_1 - I_3$. If $|I_1| > 2$, then either $|I_3| = 1$ or $|I_2| = 0$.

Proof. Assume to the contrary that $|I_3| > 1$ and $|I_2| > 0$. Let $v_1 \in K_1 - I_1$, $v_2 \in I_2$, and $v_3, v'_3 \in I_3$ (see an adjacency scenario demonstrated in Fig. 2). Note that $|N_G[u_1] \cap N_G[u_2]| \geq |I_1| > 2$, for $u = v_2, v_3, v'_3$. From Lemma 3.7, $d_T(v_1, v_2) = d_T(v_1, v_3) = d_T(v_1, v'_3) = 4$. Since $d_T(v_2, v_3) \leq 3$, we conclude that $d_T(v_2, v_3) = 2$ and similarly $d_T(v_2, v'_3) = 2$. Let $v_4$ be a vertex in $K_3 - I_3$ (note from the proof of Theorem 3.4 that $v_4$ is not adjacent to $v_2$).

From Lemmas 3.6 and 3.7, no matter what size set $N_G[v_2] \cap N_G[v_4]$ is of, it is impossible for $v_2$ to be at distance exactly 2 from two vertices $v_3, v'_3$ from $I_3 \subset N_G[v_2] \cap N_G[v_4]$. Such a contradiction shows that either $|I_3| = 1$, or $I_2$ must be empty. □

Lemma 3.9. Suppose graph $G$ has a 3rd Steiner root $T$. Assume there exist in $G$ three maximal cliques $K_1, K_2, K_3$ such that $K_1 \cap K_2 = I_1 \neq \emptyset$, $K_2 \cap K_3 = I_3 \neq \emptyset$, and $K_1 \cap K_3 = \emptyset$. Let $I_2 = K_2 - I_1 - I_3$. If $I_1 = \{u_1, u'_1\}$, $I_3 = \{u_3, u'_3\}$, and $|I_2| > 0$, then $u_1-u'_1-u'_3-u_3$ is a path in $T$ and every vertex in $I_2$ is adjacent to either $u'_1$ or $u'_3$.

Fig. 2. An adjacency scenario for the proof of Lemma 3.8.
Proof. Let \( v_1 \in K_1 - I_1, u_2 \in I_2, \) and \( v_2 \in K_3 - I_3 \) (see an adjacency scenario demonstrated in Fig. 3). Since \( |N_G[v_i] \cap N_G[u_2]| \geq 2 \) for \( i = 1, 2, \) it follows from Lemmas 3.6 and 3.7 that \( 4 \leq d_T(v_1, u_2) \leq 5, d_T(u_1, u'_1) \leq 2, \) and \( d_T(u_3, u'_3) \leq 2. \) Since \( \{u_1, u'_1, u_3, u'_3\} \subseteq K_2, \) the maximum distance among these four vertices in \( T, \) denoted as \( D, \) must be at most 3.

If \( D = 2, \) that is, \( u_1, u'_1, u_3, \) and \( u'_3 \) form a star in \( T, \) then at most one of them can serve as the center of the star. Assume without loss of generality that none of \( u_1, u'_1, \) and \( u_3 \) is the center, in other words, \( d_T(u_1, u_3) = d_T(u'_1, u_3) = 2. \) Therefore, \( u_1 \) and \( u'_1 \) are not adjacent in \( T. \) It follows from Lemma 3.6 that \( d_T(v_1, u_3) = 4. \) It is easy to check now that \( v_1 \) must be at a distance 4 from at least one of \( u_1 \) and \( u'_1. \) This violates the fact that \( \{v_1, u_1, u'_1\} \subseteq K_1. \) Thus, we conclude that \( D = 3, \) and furthermore assume without loss of generality that \( d_T(u_1, u_3) = 3. \)

Let us assume without loss of generality that \( d_T(u_1, u_2) < d_T(u_3, u_2). \) There shouldn’t be \( d_T(u_1, u_2) = 1, \) as it implies that \( u_2 \) is on the \( u_1\)-to-\( u_3 \) path in \( T, \) which would further imply that \( d_T(u'_1, u_2) = 1, \) a similar contradiction as in the last paragraph (by replacing \( u_3 \) with \( u_2). \) Therefore, we may conclude that \( u'_1 \) should be on the \( u_1\)-to-\( u_3 \) path in \( T \) and \( d_T(u_1, u'_1) = 1. \) For the same reasons, by substituting \( v_1 \) with \( v_2, \) \( u'_2 \) should be on the \( u_1\)-to-\( u_3 \) path in \( T \) too and \( d_T(u_3, u'_3) = 1. \) That is, \( u_1-u'_1-u'_3-u_3 \) is a path in \( T. \) It follows easily now vertex \( u_2 \) is adjacent to either \( u'_1 \) or \( u'_3 \) in order to satisfy the distance constraints. \( \square \)

Theorem 3.10. If graph \( G = (V,E) \) is strictly chordal, then determining whether or not it is a 3rd Steiner power, and if so, constructing a 3rd Steiner root, can be done in linear time.

Proof. Since graph \( G \) is strictly chordal, from the definition of strictly chordal and Lemma 2.4, we know that the critical clique graph \( CC(G) \) can be constructed in linear time and in which two maximal cliques overlap by at most one node. Also it is true that there is no sequence of maximal cliques that form a cycle (i.e., every two adjacent ones have a non-empty intersection) but have an empty intersection. Let \( \mathcal{K} \) be a maximum clique in \( CC(G), \) which can be identified at the time \( CC(G) \) is constructed (refer to the proof of Theorem 2.5). Note that if \( |\mathcal{K}| = 2, \) then replacing every node in \( CC(G) \) with a Steiner node and attaching the vertices in the critical clique as leaves to the Steiner node produces a 3rd Steiner root for \( G. \) We consider in the following the non-trivial case where \( |\mathcal{K}| > 2. \) Notice that by the definition of critical clique there is at most one node in \( \mathcal{K} \) belonging to no other maximal clique than \( \mathcal{K} \) itself. If such a node exists, then it is called the leaf node.
in $\mathcal{K}$. Every other node belonging to more than one maximal clique is called a non-leaf node.

Let $C_1$ be the largest non-leaf node (in terms of the number of vertices it contains in $G$) in $\mathcal{K}$; Let $C_2$ be the second largest non-leaf node in $\mathcal{K}$. Recall that since $|\mathcal{K}| > 2$, both $C_1$ and $C_2$ are well defined.

Case 1. $|C_1| > 2$. In this case, we conclude from Lemma 3.8 that $|C_2| = 1$ (and consequently, every non-leaf node in $\mathcal{K}$ other than $C_1$ has size 1). To construct a 3rd Steiner root, create two Steiner nodes $s(\mathcal{K})$ and $s(C_1)$ and connect them via an edge, attach vertices in $C_1$ as leaves to $s(C_1)$, replace every non-leaf node with the vertex in the corresponding critical clique and connect it to $s(\mathcal{K})$, and connect all the vertices in the leaf node to $s(\mathcal{K})$ as leaves. An example of this case is shown in Figs. 5(c)–(d) where $\mathcal{K} = C_2 \cup C_3$ (there is no leaf node in this $\mathcal{K}$).

Case 2. $|C_1| = 2$ and $|C_2| = 2$. In this case, we conclude from Lemma 3.9 that there is no other non-leaf node in $\mathcal{K}$ of size 2. And accordingly as stated in Lemma 3.9, arrange the 4 vertices in $C_1 \cup C_2$ into a path and connect, for all other nodes in $\mathcal{K}$, their containing vertices to either center of the path. For this case, there is no Steiner node created for $\mathcal{K}$. Suppose $C_3$ is a non-leaf node in $\mathcal{K}$ and it is in another maximal clique $\mathcal{K}'$. We conclude from the path configuration that $|\mathcal{K}'| = 2$; moreover, that two adjacent Steiner nodes $s(C_3)$ and $s(\mathcal{K}')$ must be created in the 3rd Steiner root such that the vertex in $C_3$ is adjacent to $s(C_3)$ and the vertices in the other node in $\mathcal{K}'$ are adjacent to $s(\mathcal{K}')$. An example of this case is shown in Fig. 5(d) where $\mathcal{K} = C_3 \cup C_5 \cup C_6$, and $\mathcal{K}' = C_2 \cup C_3$.

Case 3. $|C_1| = 2$ and $|C_2| = 1$. In this case, there will be two possible ways to interconnect vertices in the nodes in $\mathcal{K}$. In one of them, we create a Steiner node $s(\mathcal{K})$, replace $C_1$ with its two vertices connected by an edge, connect one of them to $s(\mathcal{K})$, and connect, for all other nodes in $\mathcal{K}$, their containing vertices to $s(\mathcal{K})$ (an example of this case is shown in Fig. 5(e) where $\mathcal{K} = C_6 \cup C_7$). In the other way, we do the same as in Case 1. Nonetheless, it might be the case that the interconnection for vertices in $C_1$ has been determined when $C_1$ was considered in some other maximal clique (since $C_1$ is a non-leaf node). Therefore, there must be an order of considerations to resolve the conflict during the construction, if such a solution exists.

From the above three cases, we can specify an order to examine all the critical cliques in $G$ and determine how to interconnect all the vertices in $G$. We begin with a claim that if two maximal cliques $\mathcal{K}_1$ and $\mathcal{K}_2$ fall in Case 2 and they overlap at a critical clique $C$ with $|C| = 2$, then graph $G$ is not a 3rd Steiner power. This is easily seen as vertices in $C$ must be forming a length-3 path in for both $\mathcal{K}_1$ and $\mathcal{K}_2$, which would violate the distance constraint for the vertices in the other size-2 non-leaf nodes. Similarly, if there are two overlapping maximal cliques $\mathcal{K}_1$ and $\mathcal{K}_2$ both of which contain two size-2 non-leaf nodes and one of them falls in Case 2, then graph $G$ is not a 3rd Steiner power. To conclude, if one of the following three situations occurs, then $G$ is not a 3rd Steiner power; Otherwise, the algorithm with its high-level description depicted in Fig. 4 returns a 3rd Steiner root for $G$:

- (Situation 1) There is a maximal clique $\mathcal{K}$ of size greater than 2, of which $|C_1| > 2$ and $|C_2| > 1$;
INPUT: a connected strictly chordal graph $G$;
OUTPUT: if $G$ has a 3rd Steiner root, and if so, one such root.

1. Construct the critical clique graph $CC(G)$;
2. Check all maximal cliques to make sure:
   2.1 no Situation 1 occurs;
   2.2 no Situation 2 occurs;
   2.3 no Situation 3 occurs;
3. for every non-leaf node $C$ of size $> 2$
   3.1 create a Steiner node $s(C)$;
   3.2 make vertices in $C$ leaves attached to $s(C)$;
4. for every non-leaf node $C$ of size 2
   4.1 if in a maximal clique $K$ of size $> 2$ containing another size-2 non-leaf node
      4.1.1 create a path containing these 4 vertices;
      4.1.2 connect other vertices in nodes of $K$ to the centers of the path;
      4.1.3 for every non-leaf size-1 node $C$ in $K$ and $K'$
         4.1.3.1 create a Steiner node $s(C)$;
         4.1.3.2 create a Steiner node $s(K')$;
         4.1.3.3 connect $s(C)$ to $s(K')$;
         4.1.3.4 connect the vertex in $C$ to $s(C)$;
         4.1.3.5 connect vertices in the other node of $K'$ to $s(K')$;
   4.2 if in a maximal clique $K$ of size 2 containing another size-2 non-leaf node
      4.2.1 create a Steiner node for each node;
      4.2.2 connect these two Steiner nodes via an edge;
      4.2.3 connect vertices in a node to its Steiner node (as leaves);
   4.3 else
      4.3.1 create a Steiner node $s(C)$;
      4.3.2 make vertices in $C$ leaves attached to $s(C)$;
5. for every other maximal clique $K$ not considered in Step 4.
   5.1 create a Steiner node $s(K)$;
   5.2 for every non-leaf node $C$ in $K$
      5.2.1 if $|C| > 2$, then connect $s(C)$ to $s(K)$;
      5.2.2 if $|C| = 2$ as in step 4.1, then connect $s(C)$ to $s(K)$;
      5.2.3 if $|C| = 2$ as in step 4.2, then connect the vertex in $C$ at the end of
         the path to $s(K)$;
      5.2.4 if $|C| = 1$, then connect the vertex in $C$ to $s(K)$;
   5.3 if there is a leaf node $C$ in $K$
      5.3.1 connect vertices in $C$ to $s(K)$ (as leaves);
6. Output the generated tree.

Fig. 4. A high-level description of the algorithm to construct a 3rd Steiner root.

- **(Situation 2)** There is a size-2 non-leaf node belonging to two maximal cliques $K_1$ and $K_2$, each of which contains another size-2 non-leaf node and one of which is of size greater than 2.
- **(Situation 3)** There is a maximal clique $K$ of size greater than 2, $|C_1| = |C_2| = 2$, and $K$ overlaps at a size-1 node with another maximal clique of size greater than 2.
Fig. 5. An example graph $G$ shows the steps of operations for constructing a 3rd Steiner root. (a) Graph $G$: every $C_i$ (dashed circle), $i = 1, 2, \ldots, 13$, denotes a critical clique; every filled circle denotes a vertex. (b) $CC(G)$: every patterned circle denotes a critical clique in $G$. (c) Growing a Steiner root: after step 3; the empty circle denotes a Steiner node to replace critical clique $C_2$. (d) Growing a Steiner root: after step 4; the empty square denotes a Steiner node to replace maximal clique $C_2 \cup C_3$. (e) A complete 3rd Steiner root.

The linear running time follows directly from the steps of construction as depicted in Fig. 4 and thus proves the theorem. One example that is designed to show all the cases that should be treated differently by the algorithm is provided in Fig. 5.

4. The leaf root problem

In the $k$th Leaf Root problem ($k$-LRP), the input is a simple connected graph $G = (V, E)$ where vertices represent the organisms and edges indicate similarity, and the desired output is a phylogeny $T$ on vertex set $V$ such that for every pair of vertices $u$ and $v$ in $V$, $(u, v) \in E$ if and only if $d_T(u, v) \leq k$. Note that $V$ is the leaf set of phylogeny $T$. When $k \leq 4$, $O(|V|^3)$ time algorithms have been presented in [12,13] to determine whether or not
$G$ is a $k$th leaf power, and if so, return a $k$th leaf root. Although many structural properties of the leaf powers and the leaf roots have been examined, the algorithm design techniques in [12,13] failed to cover larger $k$'s. As has been proven in Theorem 3.4, strictly chordal graphs are $k$th leaf powers, for $k \geq 4$.

**Theorem 4.1.** If graph $G = (V, E)$ is strictly chordal, then it is a $k$th leaf power for $k \geq 4$; Furthermore, a $k$th leaf root for $G$ can be computed in $O(|V| + |E|)$ time.

5. Concluding remarks

Graphs that are Steiner powers or leaf powers are chordal. For $k \geq 4$, we have shown that strictly chordal graphs are $k$th Steiner powers and are $k$th leaf powers. It would be interesting to discover exactly which subclass of chordal graphs (yet a superclass of strictly chordal graphs) are $k$th Steiner powers and/or $k$th leaf powers.

**Theorem 3.10** tells that some strictly chordal graphs are not 3rd Steiner powers. Designing polynomial time algorithms to recognize 3rd Steiner powers that are not strictly chordal is one of our future research subjects.

We didn’t discuss the phylogenetic root problem on the strictly chordal graphs in this paper. We have obtained some preliminary results along the line and hopefully key structural properties examined in the above would help us in developing an efficient algorithm to construct a $k$th phylogenetic root for a strictly chordal graph, if it exists. We have noticed that in [8], a linear time algorithm has been designed to construct a 5th phylogenetic root for a graph whose critical clique graph is a tree, if it has one. Clearly, such a graph is strictly chordal, but the class of strictly chordal graphs is larger than the class considered in [8].

References


