# On graph powers for leaf-labeled trees


Naomi Nishimura[*] Prabhakar Ragde[†]

*Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, N2L 3G1*
E-mail: {nishi,plragde}@uwaterloo.ca

and

Dimitrios M. Thilikos[‡]

*Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Campus Nord, Mòdul C5, c/Jordi Girona Salgado, 1-3. E-08034, Barcelona, Spain*
E-mail: sedthilk@lsi.upc.es



We extend the well-studied concept of a graph power to that of a ***k*-leaf power** *G* of a tree *T*: *G* is formed by creating a node for each leaf in the tree and an edge between a pair of nodes if and only if the associated leaves are connected by a path of length at most $k$. By discovering hidden combinatorial structure of cliques and neighbourhoods, we have developed polynomial-time algorithms that, for $k = 3$ and $k = 4$, identify whether or not a given graph $G$ is a $k$-leaf power of a tree $T$, and if so, produce a tree $T$ for which $G$ is a $k$-leaf power. We believe that our structural results will form the basis of a solution for more general $k$. The general problem of inferring hidden tree structure on the basis of leaf relationships shows up in several areas of application.



*Key Words:* graph power, tree power, leaf power, phylogenetic tree


## 1. INTRODUCTION

The results in this paper are derived from two abundant areas of research: graph powers and leaf-labeled trees. Both areas contain results of a purely theoretical nature as well as applications to such diverse areas as distributed computing [8], computational biology, and mathematical psychology [1].

Trees are versatile in their ability to represent relations between data items stored in their nodes. In many instances, data items are stored in a subset of the nodes (typically leaves); the structure of internal nodes is dictated by measures of distance or similarity among leaves. For example, a Steiner tree is a tree of minimal length containing every point in a set of inputs; a more general formulation is known as an $X$-tree [1]. A fundamental problem in computational biology is the reconstruction of the *phylogeny*, or evolutionary history, of a set of species or genes, typically represented as a *phylogenetic tree* (the reader is referred to papers that review research in the area of evolutionary history [4, 5, 7]). In a phylogenetic tree, each leaf is labeled by a distinct known species; a tree is then formed by positing possible ancestors that might have led to this set of species.

By viewing the correlations between leaves as distances between nodes in a graph, we can frame the problem of forming a phylogenetic tree as the problem of forming a tree from a graph. One such


[*] Supported by the Natural Sciences and Engineering Research Council of Canada.
[†] Supported by the Natural Sciences and Engineering Research Council of Canada.
[‡] Supported by the Ministry of Education and Culture of Spain, Grant number MEC-DGES SB98 0K148809.

correlation between graphs and trees, or more generally between graphs and graphs, arises in the notion of *graph powers*. A graph $G$ is the *kth power* of a graph $H$ if nodes $x$ and $y$ are adjacent in $G$ if and only if the length of the shortest path from $x$ to $y$ in $H$ is at most $k$. Although in general it is NP-complete to recognize a graph power [9], it is possible to determine if a graph is the power of a tree in time $O(n^3)$, where $n$ is the number of vertices in the input graph [2].

In this paper we introduce the notion of a *$k$-leaf power* of an unlabeled tree $T$, where a graph $G$ is the *$k$-leaf power* of a tree $T$ if there exists a vertex in $V(G)$ for each leaf in $T$ and an edge in $E(G)$ between vertices $u$ and $v$ if and only if there is a path of length at most $k$ between the leaves associated with $u$ and $v$ in $T$. The problem of recognizing $k$-leaf powers is inspired by the problem of forming a phylogenetic tree based on distance thresholds: given a graph $G$ in which there is an edge for each pair of species at distance at most $k$, the tree $T$ of which $G$ is a $k$-leaf power is a phylogenetic tree in which the associated leaves are guaranteed to be at distance at most $k$. In practice, phylogenetic problems involve edge-weighted trees and errors in distance estimators; our algorithms should be seen as a first step towards handling this more general situation. A related concept is that of the *threshold graph* formed on a set of $n$ nodes and a weighting function on edges by including only edges less than a set threshold; there exist algorithms to extract a tree from the graph by first finding connected components [1].

We derive polynomial-time algorithms for recognizing $k$-leaf powers for $k = 3$ and $k = 4$. Our algorithms are based on the hidden structure of $k$-leaf powers, of independent combinatorial interest; as leaf labels do not play a part in our algorithms, our work is applicable to arbitrary trees. The complex characterizations of $k$-leaf powers are based on the structural properties of cliques and neighbourhoods. The properties are particularly tricky to derive in the presence of internal nodes that are not the neighbours of leaves: such nodes serve as invisible entities that subtly alter the structure of the relationships of neighbourhoods. The lemmas we prove may be helpful not only in generalizing our results to $k > 4$, but also in unrelated problems on graphs and trees.

We first establish properties of neighbourhoods in trees in Section 3. Next, we present a representation of the original graph as a *clique graph*, defined in Section 4. Section 5 contains polynomial-time algorithms which determine whether or not a graph $G$ is a 3-leaf power or a 4-leaf power of a tree $T$, and if so, demonstrate one such $T$. Finally, directions for future research are discussed in Section 6.

## 2. PRELIMINARIES

Our algorithms identify tree structure by examining cliques of leaves in the tree. As a consequence, we need to distinguish between internal vertices that are neighbours of leaves (*visible vertices*) and internal vertices that are not neighbours of leaves (*invisible vertices*). A tree that does not contain any invisible vertices is an *ideal tree*. To help avoid confusion, we will refer to *vertices* in a tree $T$ and *nodes* in its $k$-leaf power $G$.

The case of 2-leaf powers is not interesting; a 2-leaf power $G$ is a set of disjoint cliques, each clique corresponding to the leaves of $T$ adjacent to one internal vertex. Any tree formed by connecting the internal vertices yields the same 2-leaf power.

When $k > 2$, the set of leaves of $T$ adjacent to an internal vertex also forms a clique in $G$, but these cliques may overlap and will not, in general, be maximal. It is the maximal cliques of $G$ that hold the key to reconstructing $T$, and we must find elements in $T$ that correspond to these cliques.

Since a $k$-leaf power is an induced subgraph of a power of trees, and powers of trees are chordal [6], clearly the graphs we are trying to recognize are chordal. We can check for chordality and find all maximal cliques in linear time [3, 10].

A few easy observations will simplify our task. Given a graph $G$ which is a $k$-leaf power, we can treat each connected component separately, and connect the resulting trees by paths of length $k$. Consequently we can assume (and will do so for the rest of the paper) that $G$ is connected. Any tree $T$ whose 3-leaf power is connected has no invisible vertices (as these would disconnect the 3-leaf power) and similarly, any tree $T$ whose 4-leaf power is connected cannot have two adjacent invisible vertices.

The *distance* $d(u, v)$ between two nodes $u, v$ in a tree $T$ is the number of edges in the unique path between them. We will find it convenient to define the distance $d(u, e)$ between a node $u$ and an edge $e = (v, w)$ as $(d(u, v) + d(u, w))/2$. Note that, in a tree, this is always of the form $j + \frac{1}{2}$ for an integer $j$; intuitively, the extra half is the amount needed to "get to the centre of the edge". Similarly, we define the distance $d(e_1, e_2)$ between two edges $e_1$ and $e_2$ in a tree as one more than the number of edges in the unique path between them. Intuitively, the addition is due to the two extra halves needed to "get to the centre" of each edge. It is not hard to verify that the triangle inequality holds for this extended notion of distance.

DEFINITION 2.1. The $2k$-*neighbourhood* with *center* vertex $v$ in a tree $T$ is the set $S$ of all leaves of distance at most $k$ from internal vertex $v$. The $(2k+1)$-*neighbourhood* with *center* edge $e$ in a tree $T$ is the set $S$ of all leaves of distance at most $k + \frac{1}{2}$ from edge $e = (u, v)$, where $u$ and $v$ are both internal vertices.

LEMMA 2.1. *In a $2k$-leaf power $G$ of a tree $T$, the vertices of any maximal clique $M$ of $G$ form a $2k$-neighbourhood in $T$ of some internal vertex $v$.*

*Proof.* We let $P$ be a longest path in $T$ between two points of $M$, and let $u$ and $w$ be the endpoints of $P$. Clearly, $P$ has length at most $2k$. We define $v$ to be the midpoint of $P$ (if $P$ contains an even number of vertices, we break the tie arbitrarily) and let $N$ be the $2k$-neighbourhood of $v$.

To see that $M \subseteq N$, we suppose instead that there exists a vertex $x \in M \backslash N$, and hence $d(x, v) > k$. The path from $x$ to $v$ must contain an edge from a vertex outside of $P$ to a vertex $z$ in $P$; without loss of generality, we assume that $z$ appears on the path from $u$ to $v$ in $P$. Since $P$ is a longest path, $d(u, w) \geq d(x, w)$, and hence we can conclude that $d(u, v) \geq d(x, v)$. By assumption $d(x, v) > k$, which implies that $d(u, v) > k$, and hence the length of $P$ is greater than $2k$, a contradiction.

Next, we show that $N \subseteq M$. Let $y$ be an arbitrary vertex in $N$ and $x$ an arbitrary vertex of $M$. Since $M \subseteq N$, $d(x, v) \leq k$. Thus $d(x, y) \leq d(x, v) + d(v, y) \leq 2k$, and $x$ and $y$ are connected in $G$. Since $x$ was arbitrary, $y$ is connected to every vertex in $M$, and by maximality of $M$, $y \in M$. ▌

LEMMA 2.2. *In a $(2k+1)$-leaf power $G$ of a tree $T$, the vertices of any maximal clique $M$ of $G$ form a $(2k+1)$-neighbourhood in $T$ of some edge $e$ between two internal vertices.*

*Proof.* Similar to that of the previous lemma. ▌

The converse of Lemmas 2.1 and 2.2 fails to hold. For example, we can construct a path $u$, $v$, $w$, and $x$ of internal vertices such that both $v$ and $x$ are invisible, and all other neighbours of $w$ are leaves. Then, the 4-neighbourhood with center $w$ (the leaf neighbours of $w$) is a proper subset of the 4-neighbourhood with center $v$ (the leaf neighbours of $u$ and $w$). Even in an ideal tree, vertices "close to the edge" can have nonmaximal 4-neighbourhoods, in a way quantified in the next section, where we look at the structure underlying neighbourhoods and maximal cliques.

## 3. PROPERTIES OF NEIGHBOURHOODS

### 3.1. Properties of neighbourhoods in arbitrary trees

We will discover the hidden structure of the underlying tree of a $k$-leaf power by intersecting maximal cliques, which are neighbourhoods. The following technical lemma aids in characterizing the structure of intersections of neighbourhoods.

Lemma 3.1. *For $v_1$ and $v_2$ internal vertices or edges in a tree $T$ such that $d(v_1, v_2) = r$, and $S_i$ the $k_i$-neighbourhood of $v_i$ for $k_i \geq 2$, $i \in \{1, 2\}$, $k_1 \geq k_2$, the following conditions hold:*

*(a) if $\frac{k_1+k_2}{2} - 2 < r$, then $S_1 \cap S_2 = \emptyset$,*

*(b) if $r \leq \frac{k_1-k_2}{2}$, then $S_2 \subseteq S_1$.*

*(c) if $\frac{k_1-k_2}{2} < r \leq \frac{k_1+k_2}{2} - 2$, then $S_1 \cap S_2$ is the $(\frac{k_1+k_2}{2} - r)$-neighbourhood of the unique vertex/edge whose distance from $v_1$ is $\frac{k_1-k_2}{4} + \frac{r}{2}$ and whose distance from $v_2$ is $\frac{k_2-k_1}{4} + \frac{r}{2}$.*

*Proof.* We will examine the case where $k_1, k_2$, and $r$ are all even; the analysis for the other cases is very similar. First, we prove the contrapositive of (a). Suppose that $S_1 \cap S_2 \neq \emptyset$, $w$ is an arbitrary leaf in $S_1 \cap S_2$, and $v$ is the unique vertex of $T$ that is adjacent to $w$. Clearly, $d(v, v_i) \leq \frac{k_i}{2} - 1$, for $i = 1, 2$, and hence $r = d(v_1, v_2) \leq d(v_1, v) + d(v, v_2) \leq \frac{k_1+k_2}{2} - 2$.

Suppose now that $r \leq \frac{k_1-k_2}{2}$. For any $x \in S_2$, $d(v_2, x) \leq \frac{k_2}{2}$. As $d(v_1, x) \leq d(v_1, v_2) + d(v_2, x)$, we can conclude that $d(v_1, x) \leq r + \frac{k_2}{2} \leq \frac{k_1}{2}$ and thus $x \in S_1$. Therefore, $S_2 \subseteq S_1$ and (b) follows.

To prove (c), let $u$ be the unique vertex of $T$ that is at distance $\frac{k_1-k_2}{4} + \frac{r}{2}$ from $v_1$ and distance $\frac{k_2-k_1}{4} + \frac{r}{2}$ from $v_2$, and $S$ be the $(\frac{k_1+k_2}{2} - r)$-neighbourhood of $u$. We must show that $S = S_1 \cap S_2$. For $x \in S$, $d(x, u) \leq \frac{k_1+k_2}{4} - \frac{r}{2}$. As $d(u, v_1) = \frac{k_1-k_2}{4} + \frac{r}{2}$, we conclude that $d(x, v_1) \leq d(x, u) + d(u, v_1) \leq \frac{k_1+k_2}{4} - \frac{r}{2} + \frac{k_1-k_2}{4} + \frac{r}{2} = \frac{k_1}{2}$, and hence $x \in S_1$. Similarly, we can show that $x \in S_2$ and thus $S \subseteq S_1 \cap S_2$.

For $x \notin S$, $d(x, u) > \frac{k_1+k_2}{4} - \frac{r}{2}$. Deleting $u$ divides $T$ into connected components; at least one of the $v_i$'s, say $v_1$, is not in the connected component that contains $x$. As $v_1$ and $x$ are in different connected components, clearly $d(x, v_1) = d(x, u) + d(u, v_1) > \frac{k_1+k_2}{4} - \frac{r}{2} + \frac{k_1-k_2}{4} + \frac{r}{2} = \frac{k_1}{2}$ and so $x$ cannot be in the $k_1$-neighbourhood of $v_1$. This implies that $x \notin S_1 \cap S_2$, and we conclude that $S = S_1 \cap S_2$. ∎

Using Lemma 3.1 we can easily prove the following results concerning the structure of neighbourhoods. Although stated in a general form, in this paper we apply these primarily in the case $j = 4$.

Lemma 3.2. *The following conditions hold for any tree $T$ and $j \geq 4$:*

*1. The intersection of two distinct $j$-neighbourhoods is either empty or a $j'$-neighbourhood for $2 \leq j' \leq j - 1$.*

*2. No $(j-1)$-neighbourhood is a subset of more than two distinct $j$-neighborhoods for $j$ even.*

*3. If a $(j-2)$-neighbourhood is a subset of a $j$-neighbourhood, then their centers are either identical or adjacent.*

*4. The $(j-2)$-neighbourhood of a vertex of degree at least two is the intersection of the $j$-neighbourhoods of any two of its neighbours for $j$ even.*

*5. The $(j-1)$-neighborhood of an edge is the intersection (union) of the $j$-neighbourhoods ($(j-2)$-neighbourhoods) of its endpoints.*

*Proof.* Condition (1) follows by applying $k_1 = k_2 = j$, $r \geq 1$ (cases (a) and (c)). For condition (2), observe that, among three distinct $j$-neighbourhoods of vertices, there are two with centers of distance at least 2; applying Lemma 3.1 for $k_1 = k_2 = j$ and $r \geq 2$ (cases (a) and (c)) shows that their intersection cannot be a $(j-1)$-neighbourhood. Conditions (3), (4), and (5) follow from Lemma 3.1 with $k_1 = j, k_2 = j - 2$, $r < 2$ (case (b)), $k_1 = k_2 = j$, $r = 2$ (case (c)), and $k_1 = k_2 = j$, $r = 1$ (case (c)) respectively. The parenthesized version of condition (5) is obvious. ∎

In addition to the above general properties, the following specific properties are useful for the case $k = 4$.

LEMMA 3.3. *The following conditions hold for any tree $T$:*

*1. All the 2-neighbourhoods of the internal vertices of $T$ are vertex disjoint.*

*2. The intersection of the 4-neighbourhood of $v$ and the 3-neighbourhood of $e$ is either the 3-neighbourhood of $e$, a 2-neighbourhood of one of the endpoints of $e$, or empty.*

*3. The intersection of the 4-neighbourhood of $v_1$ and the 2-neighbourhood of $v_2$ is either the 2-neighbourhood of $v_2$, or empty.*

*4. Let $S$ be the intersection of two 3-neighbourhoods of two edges $e_1, e_2$. If $e_1$ and $e_2$ are adjacent then $S$ is the 2-neighbourhood of their common endpoint; otherwise $S$ is empty.*

*Proof.* Condition (1) follows by applying Lemma 3.1 with $k_1 = k_2 = 2$, $r \geq 1$ (case (a)), condition (2) by applying Lemma 3.1 with $k_1 = 4$, $k_2 = 3$, $r = \frac{1}{2}$ (case (b)), $r = \frac{3}{2}$ (case (c)), and $r > \frac{3}{2}$ (case (a)), and condition (3) by applying Lemma 3.1 with $k_1 = 4$, $k_2 = 2$, $r = 1$ (case (b)) and $r > 1$ (case (a)). Finally, condition (4) follows for $k_1 = k_2 = 3$ and $r > 1$ (case (a)), or $r = 1$ (case (c)). ▌

### 3.2. Properties of neighbourhoods in ideal trees

We make use of terminology that distinguishes between types of vertices in a tree. For $T'$ the tree obtained from $T$ after two successive leaf prunings, we partition the internal vertices of $T$ into those which are not in $T'$ (*marginal vertices*), those which are leaves in $T'$ (*peripheral vertices*) and those which are internal vertices in $T'$ (*central vertices*). Any edge incident on a central vertex is a *central edge*. An edge in $T$ is *pendant* if one of its endpoints is a leaf.

We now derive some properties of ideal trees.

LEMMA 3.4. *The following conditions hold for any ideal tree $T$:*

*1. No 4-neighbourhood (3-neighbourhood) of a nonmarginal vertex (central edge) is a subset of the 4-neighbourhood (3-neighbourhood) of another nonmarginal vertex (central edge).*

*2. The 4-neighbourhood of a peripheral vertex $v$ contains only one 3-neighbourhood of a central edge, the one centered on the unique central edge that contains $v$ as endpoint.*

*3. Let $\{S_1, \ldots, S_\ell\}$ be a set of 3-neighbourhoods of nonpendant edges, and $S$ a 4-neighbourhood (2-neighbourhood) of a nonmarginal vertex. Then $S_i \subset S$ ($S_i \supset S$), $i = 1, \ldots, \ell$ if and only if the centers of $S_i$, $i = 1, \ldots, \ell$ contain the center of $S$ as an endpoint.*

*4. Let $S_1$ and $S_2$ be two distinct 4-neighbourhoods (2-neighbourhoods) of nonmarginal vertices and $S$ a 3-neighbourhood. Then $S_i \supset S$ ($S_i \subset S$), $i = 1, 2$ if and only if the center of $S$ has the centers of $S_1$ and $S_2$ as endpoints.*

*5. Let $S_2$ and $S_4$ be the 2-neighbourhood and the 4-neighbourhood of a nonmarginal vertex and let $S_3$ be a 3-neighbourhood. Then $S_2 \subset S_3$ if and only if $S_3 \subset S_4$.*

*Proof of Lemma 3.4, point (1).* We let $v_1$ and $v_2$ be two nonmarginal vertices and $S_1$ and $S_2$ their 4-neighbourhoods. By the definition of nonmarginal, there are two leaves $u_1$ and $u_2$ such that the unique path $P$ connecting $v_1$ and $v_2$ is a subpath of the path connecting $u_1$ and $u_2$, and $d(v_i, u_i) = 2$ for $i = 1, 2$. Clearly $d(v_1, v_2) \geq 1$, and hence $u_1 \in S_1$ and $d(v_2, u_1) \geq 3$. We can conclude that $u_1 \notin S_2$ and $S_1 \not\subseteq S_2$. Symmetrically, $S_2 \not\subseteq S_1$. The parenthesized case can be proved using similar reasoning. ▌

*Proof of Lemma 3.4, point (2).* By the definitions of a central edge and a peripheral vertex, there is only one central edge $e$ with a given peripheral vertex $v$ as its endpoint. It is a direct consequence of Lemma 3.2, point 5 that the 3-neighbourhood of $e$ is a subset of the 4-neighbourhood of $v$. For any other central edge $e'$, at least one endpoint $u$ of $e'$ is at distance at least two from $v$; the result follows from the fact that the leaf neighbours of $u$ are in the 3-neighbourhood of $e$ but not in the 4-neighbourhood of $v$. ▮

*Proof of Lemma 3.4, point (3).* If the centers of the $S_i$'s contain the center of $S$, $S_i \subseteq S$ is a consequence of Lemma 3.1, case (b), for $k_1 = 4, k_2 = 3$, and $r = \frac{1}{2}$. For each $i$, to show that $S_i \subset S$, it suffices to find an element of $S$ that does not belong to $S_i$. As $S$ is the 4-neighbourhood of a nonmarginal vertex $v$, there exist at least two nonpendant edges sharing $v$ as an endpoint. Of these edges, at most one is the center of $S_i$, and hence there exists a nonpendant edge $(u, v)$ which is not the center of $S_i$. Since $T$ is ideal, there exists a leaf $w$ adjacent to $u$ in $T$. Clearly, $w \in S$ and $w \notin S_i$.

To prove the parenthesized version of this direction, we apply Lemma 3.1, case (b), for $k_1 = 3, k_2 = 2$ and $r = \frac{1}{2}$ to conclude that $S_i \supseteq S$. Let $v$ and $(v, u)$ be the centers of $S$ and $S_i$ respectively. As $T$ is ideal and $(v, u)$ nonpendant, there exists a leaf $w$ adjacent to $u$ in $T$. Clearly, $w \in S_i$ and $w \notin S$. Therefore, $S_i \subset S$.

Assume now that $S_i \subset S$, $i = 1, \ldots, \ell$, and let $v$ be the center of $S$. Suppose to the contrary that $e_j$, the center of $S_j$, does not contain $v$ as an endpoint. This means that $d(v, e_j) \geq \frac{3}{2}$. By applying Lemma 3.1 for $k_1 = 4, k_2 = 3$ and $r \geq \frac{3}{2}$ we know that the intersection of $S_j$ and $S$ (which is simply $S_j$, since $S_j \subset S$) is either empty (case (a), $r > \frac{3}{2}$) or is the 2-neighbourhood $S'_j$ of a vertex, say $v_j$, adjacent to $v$ (case (c), $r = \frac{3}{2}$). Notice that $S_j \cap S$ cannot be empty as it is equal to $S_j$, which in turn cannot be empty because $T$ is ideal. By Lemma 3.2, point 5, $S_j$ is the union of the 2-neighbourhoods of the endpoints of its center, namely $v_j$ and some $v'$, which are disjoint due to Lemma 3.2, point 1. Since we have shown $S_j = S'_j$, the 2-neighbourhood of $v'$ must be empty, again a contradiction to $T$ being ideal (recall that $v'$ is an endpoint of a nonpendant edge and therefore cannot be a leaf). The proof of the parenthesized version is similar and easier. ▮

*Proof of Lemma 3.4, point (4).* If the center of $S$ has the centers of $S_1$ and $S_2$ as endpoints, then $S \subset S_i$ follows from Lemma 3.1, case (b), for $k_1 = 4, k_2 = 3$, and $r = \frac{1}{2}$ (or case (b), $k_1 = 3, k_2 = 2$ and $r = \frac{1}{2}$ for its parenthesized version) and an argument similar to the one we used in (3) above.

Assume now that $S_i \supset S$, $i = 1, 2$, and let $e$ be the center of $S$. By applying (3), we can conclude that the centers of $S_1$ and $S_2$ are distinct endpoints of the center of $S$. The proof of the parenthesized version is similar. ▮

*Proof of Lemma 3.4, point (5).* For convenience, we let $v$ be the center of $S_2$ and $S_4$. If $S_2 \subset S_3$, we can apply the parenthesized version of (3) to conclude that $v$ is an endpoint of the center of $S_3$. By applying (3) again in the opposite direction, we obtain the desired result.

Similarly, if $S_3 \subset S_4$, then by (3), $v$ is an endpoint of the center of $S_3$; using the parenthesized version in the opposite direction yields the result. ▮

## 4. CLIQUE GRAPHS AND THEIR PROPERTIES

Our algorithms rely on the representation of graphs as directed acyclic graphs of maximal cliques and their intersections; here we introduce the notion of a *clique graph* and establish properties that prove useful algorithmically. Since in subsequent sections we will discuss clique graphs along with $k$-leaf powers, we will use *nodes* to refer to clique graphs (with names drawn from the first half of the alphabet) as well as to nodes in the original graph. In contrast, *vertices* in the tree will be named from the second half of the alphabet.

### 4.1. Clique graphs

The *clique graph* $C_G$ of a chordal graph $G$ is a directed acyclic graph, whose nodes are labeled by cliques of vertices in $G$. The definition of $C_G$ is given algorithmically as described in the *clique graph algorithm* below. We build $C_G$ by first computing all node labels, and then creating edges. The node labels are computed by first determining all maximal cliques and then iteratively finding intersections of existing node labels. A node is created for each label, and an edge added from a node $a$ to a node $b$ if the label of $a$ is a subset of the label of $b$. In Lemma 4.19 we prove linear bounds on the numbers of nodes and edges of clique graphs of leaf powers; the algorithm halts if at any point these bounds are exceeded. This is necessary because the clique graph of an arbitrary chordal graph could have exponential size (consider a graph on vertices $u_1, \ldots, u_{n/2}, v_1, \ldots, v_{n/2}$ in which there is an edge $(u_i, u_j)$ and $(u_i, v_j)$ for all $i \neq j$). Finally, we construct the transitive reduction of the graph in a naive fashion, by checking triples of nodes $(a, b, c)$, and removing the edge $(a, c)$ if edges $(a, b)$ and $(b, c)$ exist (this is unambiguous since our graph is a directed acyclic graph); as this step is not the bottleneck, our method suffices.

```
** clique graph algorithm **
1     form the set 𝒮 of node sets of all maximal cliques of G
2     form the set 𝒮′ of all intersections of sets in 𝒮
3     let 𝒮 ← 𝒮 ∪ 𝒮″
4     (if at any point above the number of sets exceeds c_s n,
         stop and answer NO)
5     for each set S in 𝒮 create a node a(S) labeled S
6     for each pair of sets (S, S′) in 𝒮
7        create edge (a(S), a(S′)) if S is a proper subset of S′
8     (if at any point above the number of edges exceeds c_e n,
         stop and answer NO)
9     for every triple of sets S, S′, S″ in 𝒮
10       if S ⊆ S′ ⊆ S″, mark edge (a(S), a(S″))
11    delete all marked edges
```

Beyond ensuring a polynomial running time, we have not attempted to optimize this construction; further investigation of the properties of chordal graphs may improve the lemma below.

Lemma 4.1. *Given a chordal graph $G = (V, E)$, $C_G$ can be computed in time $O(|V|^3)$.*

*Proof.* We first find all $O(|V|)$ maximal cliques in time $O(|V|^2)$ (the number of cliques and running time are a consequence of the linear-time recognition of chordal graphs by means of a perfect elimination ordering [10, 11]). Finding intersections naively in line 2 takes cubic time given that the number of nodes is linear. Finally, forming the graph and its transitive reduction can be accomplished naively in cubic time. ∎

We use tree/DAG terminology to describe relationships between nodes of a clique graph. If $(a, b) \in E(C_G)$, we say that *$a$ is a parent of $b$* (or, alternatively, that *$b$ is a child of $a$*). A node of the clique graph is a *border node* if it has a unique parent. We say that $b$ is a *descendant* of $a$ (or that $a$ is an *ancestor* of $b$) if there exists in $C_G$ a directed path starting from $a$ and finishing at $b$. If such a path has length 2, then we call $b$ a *grandchild* of $a$ or, alternatively, we say that $a$ is a *grandparent* of $b$.

Each node has associated with it a label and a level. The label of a node $c$ in a clique graph is its *clique graph label*, denoted cglabel($c$). Clearly, the union of the sets of nodes appearing in labels is equal to the set of nodes of $G$ and there is a one-to-one correspondence between labels of sinks of

$C_G$ and maximal cliques in $G$. For $k-2$ the length of the longest directed path in $C_G$, sinks are at *level* $k-1$, and any other node is at a level one less than the minimum level of its children. If a node is at level $j$ we call it a *level-$j$ node*. We use $C_{j,j+1}$ to denote the underlying undirected subgraph of $C_G$ induced on edges between nodes at levels $j$ and $j+1$. Levels of nodes in a clique graph can easily be found by depth-first search, and we prove in subsequent sections that the clique graph of a $k$-level power has at most $k-1$ levels (for $k=3,4$). The internal vertices of a sample input $T$ are illustrated in Figure 1; for convenience, sets of leaf neighbours, omitted from the figure, are indicated by letter labels and invisible vertices are indicated by squares. Figure 2 depicts the clique graph $G$ (for $k=4$) generated from the tree $T$. In this example, the nodes with clique graph labels `mop`, `nmo`, `cde`, `jk`, and `gh` are all border nodes.

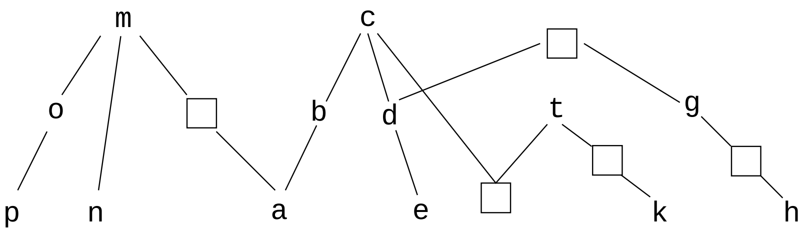


**FIG. 1.** Internal vertices of tree $T$. Letter labels denote sets of leaf neighbours; squares denote invisible vertices.

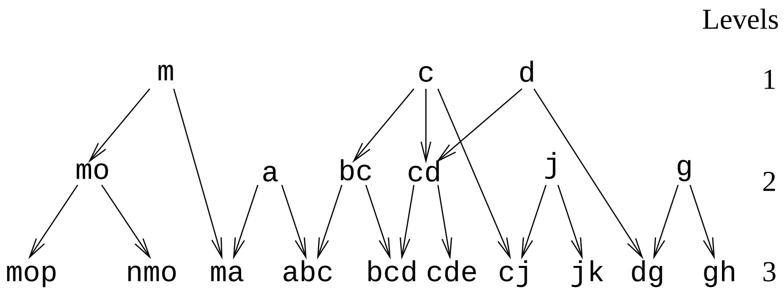


**FIG. 2.** Clique graph $G$ ($k=4$) of tree $T$ in Figure 1.

### 4.2. Ideal clique graphs

A clique graph is an *ideal clique graph* if it can be generated from a $k$-leaf power of an ideal tree $T$. Ideal clique graphs have elegant properties, as demonstrated in Lemma 4.2, which are absent from general clique graphs. We can view a general clique graph as having been generated from an ideal clique graph, with subsequent "collapsing" occurring at the invisible nodes.

LEMMA 4.2. *The following conditions hold for the clique graph $C$ of the 4-leaf power of an ideal tree $T$:*

1. *the labels of the level-3 nodes are the 4-neighbourhoods of the nonmarginal vertices of $T$;*
2. *the labels of the level-2 nodes are the 3-neighbourhoods of the central edges of $T$; and*
3. *the labels of the level-1 nodes are the 2-neighbourhoods of the central vertices of $T$.*

*Proof.* By Lemma 2.1, the label of any sink node of $C$ (i.e., any maximal clique of the 4-leaf power of $T$) is the 4-neighbourhood of some internal vertex of $T$. Lemma 3.4, point 1 shows that any 4-neighbourhood of a nonmarginal vertex of $T$ is maximal, yielding property 1. To see that 4-neighbourhoods of marginal vertices are not maximal, we observe that they are proper subsets of the 4-neighbourhoods of their peripheral neighbours.

The clique graph algorithm will generate all possible nonempty intersections of 4-neighbourhoods of nonmarginal vertices in $T$. By Lemma 3.2, point 1, these intersections are either 2-neighbourhoods or 3-neighbourhoods of vertices of $T$. Moreover, by Lemma 3.2, points 4 and 5, any 3-neighbourhood (2-neighbourhood) of a central edge (of a central vertex) is the intersection of two 4-neighbourhoods of nonmarginal vertices. Therefore, the level-2 nodes and level-1 nodes mentioned in properties 2 and 3 are all created during the first application of line 2 of the clique graph algorithm.

To prove properties 2 and 3, it will suffice to show that each 3-neighbourhood is at level 2 and each 2-neighbourhood is at level 1. As a consequence of Lemma 3.4, point 1, $C$ contains no directed path of length four. Each 3-neighbourhood is the parent of 4-neighbourhoods (Lemma 3.2, point 5) and each 2-neighbourhood is the parent of 3-neighbourhoods (Lemma 3.3, point 4), as needed. ∎

The notion of a center is well defined, due to Lemma 4.2, Lemma 3.3, point 1, and Lemma 3.4, point 1.

DEFINITION 4.1. Given a level-$j$ node $b$ in a clique graph of the 4-leaf power of an ideal tree $T$, center(cglabel($b$)) is the unique nonmarginal vertex or central edge $v$ such that cglabel($b$) is the $(j+1)$-neighbourhood of $v$.

The following lemmas determine when a clique graph is ideal; the proof of Lemma 4.3 is a simplification of that of Lemma 4.4 and hence omitted.

LEMMA 4.3. *The clique graph $C$ of the 3-leaf power of an ideal tree $T$ has the following properties:*

*1. level-2 nodes have one or two parents and no children; and*

*2. level-1 nodes have at least two children and no parents.*

LEMMA 4.4. *The clique graph $C$ of the 4-leaf power of an ideal tree $T$ has the following properties:*

*1. children of level-1 nodes are level-2 nodes, and children of level-2 nodes are level-3 nodes;*

*2. level-2 nodes have exactly two children and one or two parents; level-1 nodes have at least two children and no parents;*

*3. if $a$ is a level-1 node, then all its children share a common level-3 node as a child and no other 3-node is the child of more than one of $a$'s children; if $c$ is a level-3 node, then all its parents share a common level-1 node as a parent, and no other level-1 node is the parent of more than one of $c$'s parents; and*

*4. $C_{1,2}$ is a tree, and $C_{2,3}$ is a subtree of $C_{1,2}$.*

*Proof of Lemma 4.4, point (1).* We first observe that by Lemma 3.2, point 1, and Lemma 3.4, point 1, there are no edges in $C$ connecting nodes at the same level. Suppose that there is a level-1 node $a$ that has a level-3 node $c$ as a child, with $v_a = \text{center}(\text{cglabel}(a))$ and $v_c = \text{center}(\text{cglabel}(c))$. By Lemma 3.2, point 3, there exists a central edge $e$ in $T$ that contains both of $v_a$ and $v_c$ as endpoints (possibly $v_a = v_c$). By Lemma 3.2, point 5, for the level-2 node $b$ labeled by the 3-neighbourhood of $e$, $\text{cglabel}(a) \subset \text{cglabel}(b) \subset \text{cglabel}(c)$ and consequently edge $(a, b)$ and $(b, c)$ exist in $C$, so that edge $(a, c)$ should not exist. The fact that the children of level-2 nodes are all level-3 nodes follows from the parenthesized version of Lemma 3.4, point 1. ∎

*Proof of Lemma 4.4, point (2).* The first statement follows directly from Lemma 3.2, points 2 and 5. The second follows from the fact that any central vertex is adjacent to at least two central edges, and therefore its 2-neighbourhood is contained in at least two 3-neighbourhoods of central edges. ∎

*Proof of Lemma 4.4, point (3).* To prove the first statement, we let $a$ be a level-1 node of $C$ with children $b_1, \ldots, b_\ell, \ell \geq 2$, such that $v_a = \text{center}(\text{cglabel}(a))$ and $e_i = \text{center}(\text{cglabel}(b_i))$ for all $1 \leq i \leq \ell$. Since each cglabel$(b_i)$, for $i = 1, \ldots, \ell$, is a 3-neighbourhood of a central edge that properly contains cglabel$(a)$ (Lemma 4.2, points 2 and 3), and $v_a$ is the common endpoint of the $e_i$'s (Lemma 3.4, point 3). We now show that the $b_i$'s have a common child $c$, where $c$ is the level-3 node whose label is the 4-neighbourhood of $v_a$ (which exists by Lemma 4.2, point 1). By Lemma 3.4, point 3, cglabel$(b_i) \subset$ cglabel$(c)$, for $i = 1, \ldots, \ell$, and hence $b_1, \ldots, b_\ell$ are all parents of $c$.

Suppose now that a level-3 node $c'$ is a child of at least two of the children of $a$, say $b_1$ and $b_2$. By Lemma 3.4, point 3, center(cglabel$(c')$) is the common endpoint of the centers of $e_1$ and $e_2$. However, in the above paragraph, we showed that the common endpoint of the centers $e_1$ and $e_2$ is $v_a$. Therefore, center(cglabel$(c')$) $= v_a =$ center(cglabel$(c)$), which means that $c' = c$. The proof of the second statement is very similar, following the observation that center(cglabel$(c)$) is either peripheral or central; if $c$ has two or more parents, Lemma 3.4, point 2 rules out its being peripheral. ∎

*Proof of Lemma 4.4, point (4).* We prove that $C_{2,3}$ is a tree by showing that it is a subgraph of a tree $T'$ formed by subdividing each edge in $T$. We define $f$ to be the mapping of nodes of $C_{2,3}$ to vertices of $T'$ such that for a level-3 node $a$, $f(a) = v$ where center(cglabel$(a)$) $= v$, and for a level-2 node $b$ with children $c$ and $c'$, $f(b)$ is the subdivision vertex of the edge $(f(c), f(c'))$. It will suffice to show that $f$ is well-defined, since if it is, clearly $(a, a')$ is an edge in $C_{2,3}$ if and only if $(f(a), f(a'))$ is an edge in $T'$.

The fact that $f$ is well-defined on level-3 nodes follows from Lemma 4.2, point 1, and (2) implies that a level-2 node $b$ has two children $c$ and $c'$. As a consequence of Lemma 3.4, point 4, the center of cglabel$(b)$ has the centers of cglabel$(c)$ and cglabel$(c')$ as endpoints, and so (center(cglabel$(c)$), center(cglabel$(c')$)) $= (f(c), f(c'))$ is an edge in $T$.

To show $C_{1,2}$ is a subtree of $C_{2,3}$, it suffices to demonstrate a mapping $f$ of nodes in $C_{1,2}$ to nodes in $C_{2,3}$ such that $(a, b)$ is an edge in $C_{1,2}$ if and only if $(f(a), f(b))$ is an edge in $C_{2,3}$. We define $f$ to map each level-2 node to itself (since each such node is in both graphs) and to map each level-3 node $c$ to the level-1 node $f(c)$ such that center(cglabel$(c)$) $=$ center(cglabel$(f(c))$).

If $(a, b)$ is an edge in $C_{1,2}$, for $a$ at level 1 and $b$ at level 2, then cglabel$(a) \subset$ cglabel$(b)$, which implies that cglabel$(b) \subset$ cglabel$(f(a))$ (Lemma 3.4, point 5) and hence $(b, f(a)) = (f(b), f(a))$ is an edge in $C_{2,3}$. If $(b, c)$ is an edge in $C_{2,3}$, for $b$ at level 2 and $c$ at level 3, then cglabel$(b) \subset$ cglabel$(c)$, and again by Lemma 3.4, point 5, we can conclude that cglabel$(f^{-1}(c), b) = (f^{-1}(c), f^{-1}(b))$ is an edge in $C_{1,2}$. ∎

The next two lemmas prove useful properties needed in the proof of correctness of the algorithm for $k = 4$.

LEMMA 4.5. *In a three-level ideal clique graph, if a level-2 node $b$ has two parents $a_1$ and $a_2$, then cglabel$(b)$ = cglabel$(a_1) \cup$ cglabel$(a_2)$.*

*Proof.* Since cglabel$(a_i) \subset$ cglabel$(b)$, $i = 1, 2$, clearly cglabel$(a_1) \cup$ cglabel$(a_2) \subseteq$ cglabel$(b)$. Any node in the 3-neighbourhood of an edge $e$ belongs to one of the 2-neighbourhoods of the endpoints of $e$ (Lemma 3.2, point 5). Therefore, cglabel$(b) \subseteq$ cglabel$(a_1) \cup$ cglabel$(a_2)$, and the lemma follows. ∎

LEMMA 4.6. *In the clique graph of an ideal tree, the parent of a border level-3 node is a border level-2 node.*

*Proof.* Suppose instead that $c$ were a level-3 border node with nonborder parent $b$. Since $b$ is not a border node it has at least two parents $a_1$ and $a_2$, and by Lemma 4.4, point 2, $a_1$ has a child $b_1$ and $a_2$ has a child $b_2$ such that $b$, $b_1$, and $b_2$ are all distinct.

By Lemma 4.4, point 3, $b$ and $b_1$ must share a common child $c_1$ and $b$ and $b_2$ must share a common child $c_2$. Since $c$ has only a single parent, clearly $c_1$ and $c_2$ are distinct from $c$, violating Lemma 4.4, point 2 for $b$. ▌

### 4.3. General clique graphs

The presence of invisible nodes complicates the characterization of general clique graphs of 4-leaf powers, for which the correlation between neighbourhoods and levels is no longer so clean. For example, if $u$, $v$, $w$, and $x$ form a path in $T$ such that $v$ and $x$ are invisible and all other neighbours of $w$ are leaves, then the 4-neighbourhood of $w$ is equal to the 2-neighbourhood of $w$ and the 3-neighbourhoods of $(v, w)$ and $(w, x)$, and is a subset of the 4-neighbourhood of $v$. As a consequence of the blurring of distinctions between types of neighbourhoods, intuitively, general clique graphs of 4-leaf powers have the following structure: the sections of height 3 look like ideal clique graphs, but the sections of height 2 can be arbitrary bipartite trees. Theorem 4.1 confirms this intuition.

The following lemma characterizes a few constraints on the correlations between levels and neighbourhoods:

LEMMA 4.7. *In any three-level clique graph, the following conditions hold:*

*1. In any path of length three, the labels of the nodes are, in order from source to sink, a 2-neighbourhood, a 3-neighbourhood, and a 4-neighbourhood.*

*2. A level-3 node is always a 4-neighbourhood, sometimes a 3-neighbourhood, and never a 2-neighbourhood.*

*3. A level-2 node can be a 2-neighbourhood and/or a 3-neighbourhood, but never a 4-neighbourhood.*

*Proof.* Condition 1, the first part of condition 3, and the first part of condition 2 follow directly from the role of proper subsets in the definition of a clique graph and the fact that each level-3 node is a maximal clique and hence a 4-neighbourhood (Lemma 2.1). A level-3 node cannot be a 2-neighbourhood, since if a 2-neighbourhood is a maximal clique in the graph, the graph is not connected. The remaining statements result from the presence of invisible vertices. ▌

LEMMA 4.8. *The clique graph of a $k$-leaf power, for $k \geq 3$, has at most $k-1$ levels.*

*Proof.* Suppose instead that there is a path $(a_{h-k+1}, a_{h-k+2}, \ldots, a_{h-1}, a_h)$ of length $k$ such that for each $i$, node $a_i$ is at level $i$, and level $h$ is the highest level in the clique graph for some $h \geq k$. By Lemmas 2.1 and 2.2, cglabel$(a_h)$ is the $k$-neighbourhood of some vertex in $T$. Moreover, as the label of any node of level $i < h$ is the intersection of two nodes at levels greater than $i$, and any intersection of two neighbourhoods is also a neighbourhood (by Lemma 3.1) it is easy to prove that any label in $C$ is a neighbourhood of some vertex/edge in $T$. Observe now that cglabel$(a_{h-1})$ is the intersection of cglabel$(a_h)$ and the labels of a nonempty set $A_h$ of level-$h$ nodes. As the labels in $A_h \cup \{a_h\}$ are pairwise disjoint and all contain cglabel$(a_{h-1})$ as a proper subset, we can use Lemma 3.1 to conclude that cglabel$(a_{h-1})$ is a $k_{h-1}$-neighbourhood for some $k_{h-1} < k$. Iterating this reasoning, we can show that cglabel$(a_{h-i})$ is a $k_{h-i}$-neighbourhood for some $k_{h-i} < k - i + 1$, for $i = 1, 2 \ldots k-1$. Thus cglabel$(a_{h-k+1})$ is a $k_{h-k+1}$-neighbourhood for some $k_{h-k+1} < 2$, contradicting the fact that cglabel$(a_{h-k+1}) \subset$ cglabel$(a_{h-k+2})$ and both are nonempty. ▌

Given a three-level clique graph $C$, we decompose $C$ into a set of subgraphs and linking edges. We identify two types of three-level subgraphs, namely nondegenerate and degenerate three-level subgraphs, as well as two-level subgraphs. In Theorem 4.1 we stipulate additional conditions which ensure that $C$ is the clique graph of a 4-leaf power.

The decomposition algorithm starts by creating a partition $\mathcal{N}$ of level-1 nodes that have at least two level-2 children, where two nodes are in the same set of the partition if they share a level-2 child. For each set $P$ of the partition, it forms the subgraph $N_P$ of $C$ induced by $P$, the level-2 children of nodes in $P$, and the grandchildren of nodes in $P$. Removing every $N_P$ from $C$ temporarily, the algorithm starts to form the set $A$ of roots of additional three-level subgraphs. While there exists in $C$ a level-1 node $a$ with a level-2 child in $C$, it forms the subgraph $D_a$ of $C$ induced by $a$, its level-2 children and its grandchildren. $D_a$ is temporarily removed from $C$ and $a$ is added to $A$. Next, the algorithm forms the subgraph $F$ induced on vertices in $C$, renaming each level-1 vertex to be a level-2 vertex, and forms the set $E$ of linking edges, namely all edges of $C$ not in any $N_P$, $D_a$, or $F$. All removed components are restored, and the partitioning is done. We must now reason about its effects, and discover enough structure to justify the reconstruction algorithm.

`** clique graph partitioning **`

1 create a partition $\mathcal{N}$ of level-1 nodes that have at least two level-2 children, where two nodes are in the same set of the partition if they share a level-2 child

2 for each set $P$ of the partition form the subgraph $N_P$ of $C$ induced by $P$, the level-2 children of nodes in $P$, and the grandchildren of nodes in $P$

3 form $C'$ by removing each $N_P$ from $C$ and initialize $A = \emptyset$

4 while there exists in $C'$ a level-1 node $a$ with a level-2 child in $C'$

5     form the subgraph $D_a$ of $C$ induced by $a$, its level-2 children and its grandchildren

6     remove $D_a$ from $C'$ and add $a$ to $A$

7 form the subgraph $F$ induced on vertices in $C$, renaming each level-1 vertex to be a level-2 vertex

8 form the set $E$ of all edges of $C$ not in any $N_P$, $D_a$, or $F$

Since it is possible for the label of a node $a$ of the clique graph $C$ of a 4-leaf power of a fixed tree $T$ to be both an $i$-neighbourhood and a $j$-neighbourhood for $i \neq j$ (due to invisible vertices), we introduce the notion of a *range* (intuitively, the size of the visible part of the neighbourhood) and a *middle* (the center of the visible part of the neighbourhood). More formally, the *range* of $a$ is the length of the longest path in $T$ connecting leaves in cglabel($a$). The *middle* of $a$, denoted middle($a$), is the vertex/edge of $T$ that is in the middle of such a path (its uniqueness is a homework exercise in most graph theory textbooks).

LEMMA 4.9. *If $c$ is a node of range $k$, then cglabel($c$) is the $k$-neighbourhood of its middle.*

*Proof.* We first show that the $k$-neighbourhood of the middle $v$ of cglabel($c$) contains every node in cglabel($c$). If instead there were a vertex at distance greater than $k/2$ from $v$ in cglabel($c$), there would exist a path of length greater than $k$ connecting vertices in cglabel($c$), contradicting the assumption that the range of $a$ is $k$.

To see that the neighbourhood must be a $k$-neighbourhood we consider leaves $v_1$ and $v_2$ connected by a longest path. If cglabel($c$) were an $i$-neighbourhood for $i < k$, then each of $v_1$ and $v_2$ could be a distance of at most $i/2$ from $v$, and hence the distance from $v_1$ to $v_2$ would be at most $i < k$, contradicting the assumption that the range of $c$ is $k$. ∎

We can extract structure by examining middles in conjunction with levels of nodes.

LEMMA 4.10. *For $c$ a node in a clique graph $C$ of the 4-leaf power of a tree $T$,*

*1. if middle($c$) is a vertex $v$ for a level-3 node $c$, then cglabel($c$) is the 4-neighbourhood of $v$ and $v$ has at least two visible neighbours;*

*2. if middle($c$) is a vertex $v$ for a level-2 node $c$, then cglabel($c$) is the 2-neighbourhood of $v$ and $v$ is visible; and*

*3. if middle($c$) is an edge $e$, then cglabel($c$) is the 3-neighbourhood of $e$ and both endpoints of $e$ are visible.*

*Proof.* For $c$ a level-3 node, having a vertex as a middle forces cglabel($c$) to be either a 2-neighbourhood or a 4-neighbourhood, and by Lemma 4.7, point 2, cglabel($c$) cannot be a 2-neighbourhood. Moreover, since $v$ is the middle of cglabel($c$), $v$ is the midpoint of path of length four between two leaves and hence has at least two visible neighbouring vertices, proving condition 1.

To see that condition 2 holds, having a vertex as a middle and Lemma 4.7, point 3 force cglabel($c$) to be a 2-neighbourhood of $v$; by the definition of a middle, $v$ must be the midpoint in a path of length two between two leaves and hence visible.

Finally, condition 3 follows from the fact that $e$ is the middle and on a path of length three between two vertices. ∎

When the additional conditions specified below are satisfied, we call the subgraphs $N_P$ the *nondegenerate three-level subgraphs*, the subgraphs $D_a$ the *degenerate three-level subgraphs*, and the trees in the forest $F$ the *two-level subgraphs*. Figure 3 illustrates the decomposition of the clique graph $G$ from Figure 2, with linking edges (as defined in Theorem 4.1) appearing as dashed lines.

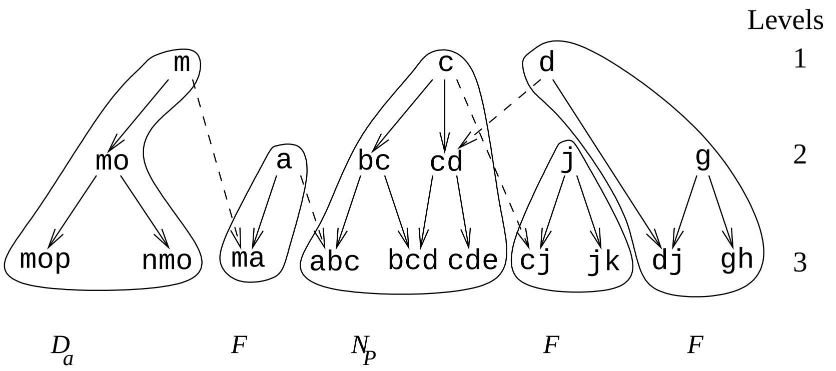


**FIG. 3.** Partitioned clique graph $G$.

To make precise the correspondence between a clique graph and $T$, it is helpful to define some additional terminology. A *visible component* of $T$ is a component of the forest induced by the visible vertices of $T$. The terms central, peripheral, and marginal defined on ideal trees can be extended in the obvious fashion to visible components. A visible component is an *ideal subtree* if it contains at least one central vertex.

THEOREM 4.1. *The following conditions are true of the clique graph $C$ of the connected 4-leaf power $G$ of a tree $T$:*

*1. For $P_1$ and $P_2$ distinct sets in the partition $\mathcal{N}$, $N_{P_1}$ and $N_{P_2}$ do not intersect.*

*2. For $a_1$ and $a_2$ distinct vertices in $A$, $D_{a_1}$ and $D_{a_2}$ do not intersect.*
*3. For any $P \in \mathcal{N}$ and $a \in A$, $N_P$ and $D_a$ do not intersect.*
*4. Each $N_P$ is isomorphic to a (necessarily ideal) clique graph of an ideal subtree.*
*5. In each $D_a$, $a$ has only one child and exactly two grandchildren.*
*6. $F$ is a forest without edges connecting nodes of the same level.*
*7. A linking edge can connect either*

*(i) a level-*1 *node of a three-level subgraph and a level-*3 *node of two-level subgraph (*central linking edge*), or*

*(ii) a level-*2 *node of a three-level subgraph and a level-*2 *node (formerly level-*1*) of a two-level subgraph (*peripheral linking edge*), or*

*(iii) a level-3 node of a three-level subgraph and a level-*2 *node of a two-level subgraph (*marginal linking edge*).*

*Proof of Theorem 4.1, point (1).* If $N_{P_1}$ and $N_{P_2}$ intersect (for distinct $P_1$ and $P_2$), they share a level-3 vertex $c$, child of level-2 nodes $b_1$ and $b_2$ in $N_{P_1}$ and $N_{P_2}$, respectively. By Lemma 4.7, point 1, $b_1$ and $b_2$ are 3-neighbourhoods, say of edges $e_1$ and $e_2$, and by Lemma 3.2, point 5, cglabel($c$) must be the 4-neighbourhood of endpoints of both $e_1$ and $e_2$. Since $e_1$ and $e_2$ are adjacent, by Lemma 3.3, point 4 $S = \text{cglabel}(b_1) \cap \text{cglabel}(b_2)$ is the 2-neighbourhood of the common endpoint of $e_1$ and $e_2$. The fact that $b_1$ and $b_2$ have parents in $C$ imply that the common endpoint has adjacent leaves; the 2-neighbourhood is thus a common parent of $b_1$ and $b_2$, contradicting the assumption that $P_1$ and $P_2$ are distinct. ∎

*Proofs of Theorem 4.1, points (2) and (3).* These results are direct consequences of line 4 and line 3 of the algorithm, respectively. ∎

*Proof of Theorem 4.1, point (4).* We prove the statement by defining, for any $N_P$, a subgraph $T_P^*$ of $T$, and then showing that $T_P^*$ is an ideal subtree of $T$ (Lemma 4.11) and that the clique graph $C^*$ of $T_P^*$ is isomorphic to $N_P$ (Lemma 4.13).

We define $T_P^*$ in terms of level-1 nodes of $N_P$ and their neighbours. For $a$ any level-1 node of $N_P$, let $(a, b, c)$ be a path of length three in $N_P$ (by the construction of $N_P$, at least one such path always exists). By Lemma 4.7, point 1, the labels of $a, b$, and $c$ are the 2-neighbourhood of a vertex $v_a$ of $T$, the 3-neighbourhood of an edge $e_b$ of $T$, and the 4-neighbourhood of a vertex $v_c$ of $T$, respectively. We take the union of all vertices $v_c$ for all paths of length three emanating from all level-1 nodes in $N_P$ to form the set $V_P \subseteq V(T)$. Furthermore, we define $V_P'$ to contain each visible vertex that is adjacent to at least one vertex in $V_P$. Finally, we set $V_P^* = V_P \cup V_P' \cup \{v \mid v \text{ is a leaf of } T\text{, adjacent to a vertex in } V_P \cup V_P'\}$. $T_P^*$ is then the subgraph of $T$ induced by $V_P^*$.

LEMMA 4.11. *$T_P^*$ is an ideal subtree of $T$.*

*Proof.* The proof of the lemma can be partitioned into three main claims, namely that all the internal vertices of $T_P^*$ are visible (Claim 4.1), $T_P^*$ is connected (Claim 4.2), and if $v$ is a visible vertex of $T$ and is adjacent to a vertex in $T_P^*$ then $v \in V_P \cup V_P'$ (Claim 4.3). Claim 4.3 shows that $T_P^*$ is a visible component of $T$ (since it is maximal), and Fact (ix) below shows that it has at least one central vertex, as required.

In what follows, we will always assume the correspondence, defined before, between paths $(a, b, c)$ of nodes of $N_P$ and the centers $v_a, e_b$, and $v_c$ of cglabel($a$), cglabel($b$), and cglabel($c$) respectively. We proceed first with facts and definitions which facilitate the classification of vertices of $T_P^*$; proofs of facts are clustered after statements for readability. Several of the facts hold for both nondegenerate and degenerate three-level subgraphs, and hence will be stated in the more general form. The first five facts lead to the first claim.

**Fact (i)** *If $a$ is a level-1 node of a three-level subgraph, then $v_a$ is visible.*
**Fact (ii)** *If $a$ is a level-1 node of a three-level subgraph and $b$ its child, $v_a$ is an endpoint of $e_b$.*
**Fact (iii)** *If $b$ is a level-2 node of a three-level subgraph, then both endpoints of $e_b$ are visible.*
**Fact (iv)** *If $b$ is a level-2 node of a three-level subgraph, then it has exactly two children in $C$, and, therefore, exactly two children in the subgraph.*
**Fact (v)** *If $c$ is the child of a level-2 node $b$ in a three-level subgraph, then $v_c$ is one of the endpoints of $e_b$.*

*Proof of Facts (i)–(v).* Fact (i) is a consequence of the facts that cglabel$(a) \neq \emptyset$ and cglabel$(a)$ is the 2-neighbourhood of $v_a$. Fact (ii) follows from Lemma 3.1 and the observations, by construction, that cglabel$(a) \subset$ cglabel$(b)$ and that cglabel$(b)$ is the 3-neighbourhood of $e_b$.

To see that Fact (iii) holds, we observe that $b$ is the child of some level-1 node $a$, $a$ is an endpoint of $e_b$ (by Fact (ii)) and $a$ is visible (by Fact (i)). Since cglabel$(a) \neq$ cglabel$(b)$, the other endpoint of $e_b$ must also be visible, as needed.

Fact (iv) follows from the observation that by Lemma 3.2, point 2 (for $j = 4$), the 3-neighbourhood cglabel$(b)$ cannot be a proper subset of more than two 4-neighbourhoods.

We prove Fact (v) by contradiction. If $e_b = (x, y)$ and $v_c \notin \{x, y\}$, then the distance between $v_c$ and $e_b$ is at least $\frac{3}{2}$. By applying Lemma 3.1, for $k_1 = 4$ and $k_2 = 3$, we conclude that either cglabel$(b) \cap$cglabel$(c)$ is empty (point (a) for $r > \frac{3}{2}$) or cglabel$(b) \cap$cglabel$(c)$ is the 2-neighbourhood of an endpoint of $e_b$, say $x$ (point (c) for $r = \frac{3}{2}$). Because cglabel$(c) \supset$ cglabel$(b)$, cglabel$(b) \cap$ cglabel$(c) =$ cglabel$(b)$, which implies that $y$ is invisible, contradicting Fact (iii) for $e_b = (x, y)$. ∎

*Claim 4.1.* All internal vertices of $T_P^*$ are visible.

*Proof.* We consider a path $(a, b, c)$ used to construct the vertex set of $T_P^*$. Each $v_c$ is visible, since it is an endpoint of $e_b$ (Fact (v)) and hence visible (Fact (iii)); all vertices in $V_P$ are thus visible. By definition, the vertices in $V'_P$ are visible, and hence we can conclude that all the internal vertices of $T_P^*$ are visible, as needed to complete the proof. ∎

To prove that $T_P^*$ is connected (Claim 4.2), we establish correlations between level-1 and level-3 vertices (Fact (vii)), which in turn allows us to classify centers of level-3 nodes. Given a level-1 node $a$ of $N_P$ and a level-3 node such that $v_c = v_a$, we call $c$ the *projection* of $a$. We classify centers of the level-3 nodes in $N_P$ as follows: if a vertex $v_c$ is the center of the projection of a level-1 node, $v_c$ is a *$P$-inner* vertex of $T$; otherwise, it is a *$P$-side* vertex of $T$. Facts (viii) and (x) suffice to prove Claim 4.2.

**Fact (vi)** *For $(a, b, c)$ a path in a three-level subgraph, either $v_c = v_a$ or $e_b = (v_c, v_a)$.*
**Fact (vii)** *For any level-1 node $a$ in a $N_P$, there exists a (clearly unique) level-3 node $c$ such that $v_a = v_c$.*
**Fact (viii)** *Each $P$-side vertex $v_c$ in $T$ is adjacent to a $P$-inner vertex of $T$.*

*Proofs of Facts (vi)–(viii).* Fact (vi) follows as a corollary of Facts (ii) and (v).

The proof of Fact (vii) is as follows. Suppose $b$ is a child of $a$ and $c$ and $c'$ are the children of $b$, as determined by Fact (iv). By Fact (v), we conclude that $e_b = (v_c, v_{c'})$. As a consequence of Fact (ii), either $v_c$ or $v_{c'}$ is identical to $v_a$ completing the proof.

To see that Fact (viii) holds, we first observe that $c$ is the level-3 node of a path $(a, b, c)$. Since $v_c$ is a $P$-side vertex, $v_c \neq v_a$. By Fact (vi), we can conclude $e_b = (v_c, v_a)$. As a consequence of Fact (vii) there exists a level-3 node $c'$ such that $v_a = v_{c'}$, and hence $v_a$ is a $P$-inner vertex. ▮

**Fact (ix)** *Any $P$-inner vertex of $T$ is the center of a path in $T$ of five visible vertices.*

*Proof of Fact (ix).* We let $a$ be a level-1 node in $N_P$ and let $b$ and $b'$ be two of its children, whose existence is guaranteed by the construction of $N_P$; we will show that $v_a$ is the center of a path of five visible vertices. The vertex $v_a$ is the common endpoint of $e_b$ and $e_{b'}$ (Fact (ii)). By Fact (iv), $b$ and $b'$ each have exactly two children in $N_P$; by Fact (vi) we can assume that $b$ has children $c$ and $c''$, $b'$ has children $c'$ and $c''$, $v_{c''} = v_a$, $e_b = (v_c, v_a)$ $e_{b'} = (v_a, v_{c'})$ and (by Fact (iii)) $v_c, v_a$ and $v_{c'}$ are all visible in $T$.

To extend the path $(v_c, v_a, v_{c'})$ to a path of five vertices, we observe that since $c$, $c''$,and $c'$ are level-3 nodes of $N_P$, the 4-neighbourhoods of $v_c, v_{c''} = v_a$, and $v_{c'}$ induce maximal cliques in $G$. Since the 4-neighbourhoods are distinct, each of $v_c$ and $v_{c'}$ must have a visible neighbour other than $v_a$, completing the proof of Fact (ix). ▮

**Fact (x)** *Two level-1 nodes $a$ and $a'$ in $N_P$ share a level-2 child if and only if the centers $v_c$ and $v_{c'}$ of their projections are adjacent in $T$.*

*Proof of Fact (x).* If $a$ and $a'$ are two level-1 nodes of $N_P$ with common child $b$, then $v_a$ and $v_{a'}$ are both endpoints of $e_b$ (Fact (ii)) and hence $v_c = v_a$ is adjacent to $v_{c'} = v_{a'}$.

To prove the converse, suppose that $v_c$ and $v_{c'}$ are the adjacent centers of the labels of the projections of two level-1 nodes $a$ and $a'$ of $N_P$. Both $v_c$ and $v_{c'}$ are $P$-inner vertices and therefore each is the center of a path of five visible vertices in $T$ (Fact (ix)). These paths may or may not each use the edge $(v_c, v_{c'})$, but in each case the edge $(v_c, v_{c'})$ is the center of a path $(y, z, v_c, v_{c'}, z', y')$ of six visible vertices in $T$. It is now easy to see that the 4-neighbourhoods of $z$, $v_c$, $v_{c'}$, and $z'$ are maximal cliques in $C$ and therefore correspond to labels of level-3 nodes of $C$.

During the formation of the clique graph, first the level-2 nodes $w$, $b$, and $w'$ (labelled by the 3-neighbourhoods of $(z, v_c)$, $(v_c, v_{c'})$, and $(v_{c'}, z')$, respectively) will be created. Next, the procedure will form the level-1 nodes whose labels are the 2-neighbourhoods of $v_c$ and $v_{c'}$, i.e. the level-1 nodes $a$ and $a'$. In addition, the edges $(a, b)$ and $(a', b)$ will be formed in $C$. Finally, the clique graph partitioning procedure will construct $C_P$ so that $(a, b)$ and $(a', b)$ will be edges of $N_P$, as claimed. ▮

*Claim 4.2.* $T_P^*$ is connected.

*Proof.* Fact (x) proves that the subgraph of $T_P^*$ induced by $P$-inner vertices is connected; Fact (viii) proves that the subgraph of $T_P^*$ induced by $V_P$ is connected. The connectivity of $T_P^*$ follows by the construction of $V_P^*$. ▮

*Claim 4.3.* If $v$ is a visible vertex of $T$ and is adjacent to a vertex in $T_P^*$, then $v \in V_P \cup V_P'$.

*Proof.* We let $y$ be a visible node of $T$ that is adjacent to a vertex $z$ in $V_P^* = V_P \cup V_P'$ and show that $y \in V_P^*$. If $z \in V_P$ then, by definition, $y \in V_P'$ and thus $y \in V_P^*$. Suppose now that $y$ is adjacent to a vertex $z \in V_P'$ but not to any vertex in $V_P$. By definition, $z$ is adjacent to a vertex $w$ in $V_P$. We examine two cases:

**Case 1:** $w$ is a $P$-inner vertex of $T$. In this case, there exists a level-1 node $a$ in $N_P$ whose label has $w$ as center, and by Fact (ix) $w$ is the center of a path of five visible vertices in $T$. We can use all or part of this path to create a path $(y, z, w, z', y')$ of five visible vertices in $T$. By reasoning about the formation of the clique graph (as in the proof of Fact (x)), we observe that $N_P$ will contain a level-2 node $b$ whose label is the 3-neighbourhood of $(z, w)$. The node $b$ is a child of $a$ in $N_P$, and $b$ will have a child $c$ whose label is the 4-neighbourhood of $z$. The existence of the path $(a, b, c)$ implies that $z$ is a vertex in $V_P$, a contradiction.

**Case 2:** $w$ is a $P$-side vertex of $T$. Clearly $w$ is adjacent to a $P$-inner vertex $w'$ (Fact (viii)) which is the center of the label of a level-1 node $a'$ in $N_P$. Fact (ix) proves that $w'$ is the center of a path of five visible vertices in $T$; using all or part of this path, we can form a path $(y, z, w, w', z', y')$ of six visible vertices in $T$. As in the previous case we can demonstrate the existence of a path $(a, b, c)$ which implies that $w$ is a $P$-inner vertex of $T$, a contradiction. ∎

This completes the proof of Lemma 4.11. ∎

The following lemma strengthens the correspondence between $T$ and $T_P^*$, and is useful in proving the isomorphism between $N_P$ and the clique graph $C^*$ of $T_P^*$.

Lemma 4.12. *A vertex $x$ is a $P$-inner vertex of $T$ if any only if it is a central vertex of $T_P^*$, and $x$ is a $P$-side vertex of $T$ if and only if it is a peripheral vertex of $T_P^*$.*

*Proof.* If $x$ is a $P$-inner vertex of $T$, then by Fact (ix) it is a central vertex of $T_P^*$. If $x$ is a $P$-side vertex of $T$, then by Lemma 4.7, point 1 and Lemma 4.10, point 1, it cannot be marginal.

We show that any central vertex of $T_P^*$ is a $P$-inner vertex of $T$ by supposing instead that there exists a central vertex that is not a $P$-inner vertex. Let $z$ be a $P$-inner vertex and let $y$ be the central but not $P$-inner vertex at minimum distance from $z$. By minimality, there is a path of central vertices from $z$ to $y$ such that each point on the path except $y$ is central and $P$-inner. We let $y'$ be the neighbour of $y$ on this path. As both $y$ and $y'$ are central, by Fact (ix) there must exist a path of six visible vertices, say $(u, w, y, y', w', u')$ in $T_P^*$.

By using the argument in the proof of Fact (x), in the clique graph formation procedure there must be level-3 nodes labeled by 4-neighbourhoods of $w$, $y$, $y'$, and $w'$, level-2 nodes labeled by 3-neighbourhoods of $(w, y)$, $(y, y')$ and $(y', w')$, and level-1 nodes labeled by 2-neighbourhoods of $y$ and $y'$. By definition, $y$ is a $P$-inner vertex, contradicting the assumption.

To complete the proof, we note that a peripheral vertex in $T_P^*$ cannot be in $V_P'$ (as it would then be marginal), so it must be in $V_P$ and hence the center of a level-3 node. Since it cannot be a $P$-inner vertex, it must be a $P$-side vertex. ∎

Lemma 4.13. *$N_P$ is isomorphic to the clique graph $C^*$ of $T_P^*$.*

*Proof.* Since $C^*$ is an ideal clique graph (Lemma 4.11), we can make use of the bijection between neighbourhoods of $T_P^*$ and nodes of $C^*$ as established in Lemma 4.2. We define $\phi = \phi_1 \cup \phi_2 \cup \phi_3$ mapping the nodes of $N_P$ to the nodes of $C^*$ as follows.

For each level-3 node $c$ of $N_P$, $\phi_3(c)$ is the level-3 node of $C^*$ whose label is identical to the 4-neighbourhood of the nonmarginal vertex $v_c$ of $T_P^*$.

For each level-2 node $b$ of $N_P$, $\phi_2(b)$ is the level-2 node of $C^*$ whose label is identical to the 3-neighbourhood of the central edge $e_b$ of $T_P^*$.

For each level-1 node $a$ of $N_P$, $\phi_1(a)$ is the level-1 node of $C^*$ whose label is identical to the 2-neighbourhood of the central vertex $v_a$ of $T_P^*$ (notice that $v_a$ is identical to the center $v_c$ of the projection of $a$).

To see that $\phi$ is a bijection, we first observe that the images of two distinct nodes in $N_P$ are distinct. Next we consider the nodes in $C^*$, level by level, and show that each has a preimage in $N_P$.

Suppose $c^*$ is a level-3 node of $C^*$. The label of $c^*$ is the 4-neighbourhood of a nonmarginal vertex $x$ of $T_P^*$ that is either a $P$-inner or a $P$-side vertex of $T$ (Lemma 4.12). The 4-neighbourhood of $x$ is the label of a node $c$ in $N_P$, and $\phi(c) = c^*$, as needed.

For $b^*$ a level-2 node of $C^*$, the label of $b^*$ is the 3-neighbourhood of a central edge $(x, x')$ of $T_P^*$. As $(x, x')$ is a central edge in $T_P^*$, one, say $x$, of its endpoints will be a central vertex in $T_P^*$ and therefore a $P$-inner vertex in $N_P$ (Lemma 4.12). Clearly, $x'$ can be either a $P$-inner vertex or a $P$-side vertex of $T$. We distinguish two cases.

**Case 1**: $x'$ is a $P$-side vertex of $T$. Let $(a, b, c)$ be a path in $N_P$ such that $v_c = x'$. By Facts (ii) and (vi), $e_b = (x', v_a)$. Since $x'$ is peripheral (Lemma 4.12), it cannot be adjacent to two distinct central vertices $v_a$ and $x$, and hence $x = v_a$. As $e_b = (x, x')$, we can conclude that $\phi(b) = b^*$.

**Case 2**: $x'$ is a $P$-inner vertex of $T$. Let $c$ and $c'$ be the level-3 nodes of $N_P$ whose labels have $x$ and $x'$ as centers. Clearly $c$ and $c'$ are projections of two level-1 nodes $a$ and $a'$ respectively. Since $x$ and $x'$ are adjacent, $a$ and $a'$ have a common child $b$ in $N_P$ (Fact (x)). Since the centers of the labels of $a$ and $a'$ are the endpoints of $e_b$ (Fact (ii)), $e_b = (x, x')$ and thus $\phi(b) = b^*$.

Finally, suppose $a^*$ is a level-1 node of $C^*$. The label of $a^*$ is the 2-neighbourhood of a central vertex $x$ of $T_P^*$ which, in turn, is a $P$-inner vertex of $T$ (Lemma 4.12). Since there exists a level-1 node $a$ in $N_P$ whose label is the 2-neighbourhood of $x$, $v_a = x$ and $\phi(a) = a^*$. This demonstrates that $\phi$ is a bijection.

To complete the proof of the lemma, it will suffice to show that $N_P$ and $T_P^*$ are isomorphic under $\phi$. By the construction of $N_P$, we observe that $N_P$ has no edges connecting level-1 vertices and level-3 vertices, which is also the case for ideal clique graphs (Lemma 4.4, point 1). In addition, in a clique graph the nodes at a given level form an independent set. Thus, to prove that $N_P$ and $T_P^*$ are isomorphic, it will suffice to prove that there exists an edge connecting a level-$i$ node and a level-$(i+1)$ node $(i = 1, 2)$ in $N_P$ if and only if there exists an edge connecting their images through $\phi$ in $T_P^*$. It is not difficult to see that that nodes in two consecutive levels of $N_P$ or $T_P^*$ are adjacent if and only if the label of the smaller level node is a subset of the label of the higher level node. By the construction of $\phi$, we can conclude that for any $d \in V(N_P)$, cglabel$(d)$ = cglabel$(\phi(d))$, as needed to complete the proof. ▮

This completes the proof of Theorem 4.1, point (4). ▮

*Proof of Theorem 4.1, point (5).* The fact that $a$ in $D_a$ has only one child $b$ follows from line 1 of the algorithm. Any level-1 node with two children is in a nondegenerate three-level subgraph. The fact that $b$ has only two children follows from the facts that cglabel$(b)$ is a 3-neighbourhood, that the labels of the children of $b$ are all 4-neighbourhoods (Lemma 4.7), and from Lemma 3.2, point 2. ▮

*Proof of Theorem 4.1, point (6).* In order to prove that $F$ is a forest, we need to show that there exists no cycle among nodes in $F$. By construction, the neighbours of any level-1 node in $F$ are all at level 3. Since there are no edges between nodes at the same level, each edge in $F$ is between a level-3 node and a node not at level 3. It thus suffices to prove Lemma 4.18 below, which claims that $C$ does not contain any even cycle in which each edge in the cycle connects a level-3 node to a node not at level 3.

In order to prove Lemma 4.18, we need to establish properties concerning nodes which might appear in such a cycle. We call two level-3 nodes *close* if they have a common neighbour; Lemma 4.14 (a direct consequence of Lemma 3.1 and Lemma 4.7, point 2) and Lemma 4.15 establish properties of middles of close nodes.

LEMMA 4.14. *If two level-3 nodes $c_1$ and $c_2$ are close and their middles are $x_1$ and $x_2$ respectively, then $1 \leq d(x_1, x_2) \leq 2$.*

LEMMA 4.15. *It is not possible for both of the middles of two close nodes to be edges.*

*Proof.* Suppose instead that $c_1$ and $c_2$ are close nodes with middles $(x, y)$ and $(y, z)$ (in order for the clique graph labels of $c_1$ and $c_2$ to have a nonempty intersection, their middles must share an endpoint). For $b$ the common neighbour of $c_1$ and $c_2$, cglabel$(b)$ = cglabel$(c_1) \cap$ cglabel$(c_2)$ is nonempty, and hence $y$ is a visible vertex. This, in turn, implies that $x$ and $z$ are visible as well (otherwise cglabel$(c_1)$ = cglabel$(b)$ or cglabel$(c_2)$ = cglabel$(b)$). As a consequence, the 4-neighbourhood of $y$ is maximum clique containing the 3-neighbourhoods of $(x, y)$ and $(y, z)$ and hence the labels of $c_1$ and $c_2$, contradicting the assumption that $c_1$ and $c_2$ are level-3 nodes. ▌

The following two lemmas establish properties of paths involving close nodes, which prove useful in the proof of Lemma 4.18. Lemma 4.17 is a consequence of applying Lemma 4.16 iteratively.

LEMMA 4.16. *For $c_i$ a level-3 node with middle $x_i$, $i = 1, 2, 3$, if $c_1$ is close to $c_2$, $c_2$ is close to $c_3$, and $c_1$ is not close to $c_3$, then $x_2$ is a vertex on the path in $T$ connecting $x_1$ and $x_3$.*

*Proof.* We suppose instead that $x_2$ is not on the path connecting $x_1$ and $x_3$ in $T$, and without loss of generality assume that $x_3$ is on the path connecting $x_1$ and $x_2$ in $T$. The three middles must be distinct as they are middles of level-3 nodes, and hence either 4-neighbourhoods or 3-neighbourhoods. By Lemma 4.14, $1 \leq d(x_1, x_2) \leq 2$, and hence it suffices to consider the following cases:

**Case 1**: $d(x_1, x_2) = 1$. By Lemma 4.15, $x_1$ and $x_2$ are both vertices, and thus the labels of $c_1$ and $c_2$ are 4-neighbourhoods of $x_1$ and $x_2$ respectively. Since $x_3$ is on the path from $x_1$ to $x_2$, cglabel$(c_3)$ is the 3-neighbourhood of $(x_1, x_2)$, properly contained in the labels of both $c_1$ and $c_2$, and contradicting the assumption that $c_3$ is a level-3 node.

**Case 2**: $d(x_1, x_2) = \frac{3}{2}$. Without loss of generality we assume that $x_1$ is a vertex and $x_2$ is an edge $(y, z)$. Therefore, the label of $c_1$ is a 4-neighbourhood, the label of $c_2$ is a 3-neighbourhood, and $x_3$ is either the edge $(x_1, y)$ or the vertex $y$. In the first case cglabel$(c_3) \subset$ cglabel$(c_1)$, contradicting the assumption that $c_3$ is a level-3 node and in the second case cglabel$(c_2) \subset$ cglabel$(c_3)$, contradicting the assumption that $c_2$ is a level-3 node.

**Case 3**: $d(x_1, x_2) = 2$. By Lemma 4.15, $c_1$ and $c_2$ are vertices, their labels are 4-neighbourhoods, and for $d$ their common neighbour, cglabel$(d)$ is the 2-neighbourhood of the unique vertex $x$ between $x_1$ and $x_2$ in $T$. Applying Lemma 4.14 for $c_2, c_3$, we observe that $d(x_2, x_3) \geq 1$ and therefore $x_3$ cannot be the edge $(x, x_2)$. By symmetry, $x_3$ cannot be the edge $(x_1, x)$, either, so $x_3 = x$.

Since $x_3 = x$, cglabel$(c_3)$ is the 4-neighbourhood of $x$. As $c_1$ and $c_2$ are close, cglabel$(c_1) \cap$ cglabel$(c_2)$ is nonempty and consists of the 2-neighbourhood of $x = x_3$, or cglabel$(d)$. As $c_3$ and $c_2$ are close, cglabel$(c_3) \cap$cglabel$(c_2)$ is nonempty and consists of the 3-neighbourhood of $(x_3, x_2)$ which is the label of $d'$, where $d'$ is the common neighbour of $c_2$ and $c_3$. If $x_2$ is invisible, cglabel$(d')$ = cglabel$(d)$, a contradiction to the fact that $c_1$ is not close to $c_3$ (which implies that $d'$ and $d$ are distinct). If $x_2$ is visible, cglabel$(d) \subset$ cglabel$(d') \subset$ cglabel$(c_2)$, a contradiction to the existence of the edge $(d, c_2)$. ▌

LEMMA 4.17. *For $c_i$ a level-3 node for all $1 \leq i \leq r$, $r \geq 3$, such that two nodes $c_i$, $c_j$ are close if and only if $|i - j| = 1$, the middle of $c_j$ is on the path connecting $c_i$ and $c_h$ in $T$ for any $i$ and $j$ such that $1 \leq i < j < h \leq r$.*

LEMMA 4.18. *$C$ does not contain any even cycle in which each edge in the cycle connects a level-3 node to a node not at level 3.*

*Proof.* We suppose instead that $(d_1, \ldots, d_r)(r \geq 4)$ is an even chordless cycle (that is, a minimal counterexample) in which each edge in the cycle connects a level-3 node to a node not at level 3. Without loss of generality we assume that the level-3 nodes are the odd nodes.

If $r = 4$, then the existence of edges $(d_1, d_2)$ and $(d_2, d_3)$ implies that cglabel$(d_2)$ = cglabel$(d_1) \cap$ cglabel$(d_3)$ and the existence of edges $(d_3, d_4)$ and $(d_4, d_1)$ implies that cglabel$(d_4)$ = cglabel$(d_1) \cap$ cglabel$(d_3)$. We can then obtain a contradiction by concluding that $d_2 = d_4$.

If instead $r \geq 6$, we consider the path $(d_1, \ldots, d_{r-1})$ of $(d_1, \ldots, d_r)$ that avoids $d_r$. For $x_i$ the middle of $d_i$, by Lemmas 4.14 and 4.17, we observe that the path connecting $x_{d_1}$ and $x_{d_{r-1}}$ in $T$ has length at least $\frac{r-2}{2} \geq 2$. Since $d_1$ and $d_{r-1}$ are close nodes, Lemma 4.14 implies that their distance will be at most two, which is only possible if $r = 6$ and the following conditions hold:

$$d(x_1, x_3) = 1; \quad (1)$$

$$d(x_3, x_5) = 1; \text{ and} \quad (2)$$

$$d(x_1, x_5) = 2. \quad (3)$$

Since by Lemma 4.15 $x_1$, $x_3$, and $x_5$ cannot all be edges, in order to satisfy conditions (1)–(3), they must all be vertices.

We now show that both $x_3$ and $x_5$ are visible. The node $x_3$ must be visible since cglabel$(d_6)$ = cglabel$(d_1) \cap$cglabel$(d_5)$ (by (3) and the fact that $d_6$ is adjacent to both $d_1$ and $d_5$), where cglabel$(d_6)$ is nonempty and equal to the 2-neighbourhood of $x_3$. We can conclude that $x_5$ is visible, since if it were not, then cglabel$(d_4)$ = cglabel$(d_3) \cap$ cglabel$(d_5)$ would be the 2-neighbourhood of $x_3$, and hence the label of $d_6$, yielding $d_4 = d_6$, a contradiction.

Using the fact that $x_3$ and $x_5$ are visible, we are able to complete the proof. Since cglabel$(d_4)$ is the 3-neighbourhood of $(x_3, x_5)$, cglabel$(d_6) \subset$ cglabel$(d_4)$ (recall that cglabel$(d_6)$ is the 2-neighbourhood of $x_3$). It is not difficult to see that cglabel$(d_4) \subset$ cglabel$(d_5)$. Combining these observations, we conclude that cglabel$(d_6) \subset$ cglabel$(d_4) \subset$ cglabel$(d_5)$, a contradiction to the existence of edge $(d_5, d_6)$. ∎

This concludes the proof of Theorem 4.1, point (6). ∎

*Proof of Theorem 4.1, point (7).* To prove this point, we consider all ways that subgraphs can be joined and prove the impossibility of all edges other than those mentioned in the statement of the lemma. As a consequence of the way the clique graph is formed, there are no edges joining nodes at the same level. In addition, by the formation of three-level subgraphs, there can be no edge from a level-$i$ node of a three-level subgraph to a level-$(i+1)$ node of any other subgraph. The only case that remains to be considered is an edge connecting a level-3 node $c$ of a three-level subgraph and a (formerly level-1) level-2 node $a$ of a two-level subgraph.

We consider a path $(a', b', c)$ in the three-level subgraph containing $c$. As $a$ was a level-1 node in $C$ there must exist in $C$ a path $(a, b'', c'')$, where $b' \neq b''$ and $c \neq c''$ (since otherwise $a$ would be part of the same three-level subgraph as $a'$). Mimicking the beginning of the proof of point 4, we can use Lemma 4.7, point 1, to define $v_{a'}$, $e_{b'}$, and $v_c$ such that their 2, 3, and 4-neighbourhoods are cglabel$(a')$, cglabel$(b')$ and cglabel$(c)$ respectively. We define $v_a, e_{b''}, v_{c''}$ in a similar manner.

To complete the proof, we make use of facts from the proof of point 4, all of which hold for any three-level subgraph, and in particular for the path $(a', b', c)$. The node $b'$ has exactly two children (Fact (iv)), $c$ and some $c'$, such that $v_c$ and $v_{c'}$ are both endpoints of $e_{b'}$ (Fact (v)). Both $v_c$ and $v_{c'}$ are visible (Fact (iii)). We claim that $v_c$ is adjacent to a visible vertex $z$ of $T$ different from $v_{c'}$. If not, then cglabel$(c)$ will be equal to the 3-neighbourhood of $e_{b'} = (v_c, v_{c'})$ contradicting the fact that cglabel$(b') \subset$ cglabel$(c)$.

We now show that $v_a$ and $v_c$ must be adjacent. By Fact (i), cglabel$(a)$ is the 2-neighbourhood of a visible vertex $v_a$. As cglabel$(a) \subset$ cglabel$(c)$ and cglabel$(c)$ is a 4-neighbourhood, either $v_a$ and $v_c$ are the same vertex, or they are adjacent. To see that $v_a$ and $v_c$ cannot be the same vertex, we observe that $e_{b''}$ has two visible endpoints (Fact (iii)), one of which is $v_a$ (Fact (ii)). If $v_a = v_c$, then since $v_a$ is an endpoint of $e_{b''}$, the 3-neighbourhood of $e_{b''}$ is a subset of the 4-neighbourhood of $v_a$

(and hence $v_c$) (Lemma 3.2, point 5). There must then be an edge $(b'', c)$ in $C$, which is incompatible with the existence of edge $(a, c)$. We can thus conclude that $v_a$ and $v_c$ are adjacent.

We wish to show that $v_a$ has at least two visible neighbours, $v_c$ (demonstrated above), and $y$, where $e_{b''} = (v_a, y)$. Clearly $v_c \neq y$, since if $v_c = y$, there would be an edge $(b'', c)$, making $(a, c)$ impossible.

Since $v_a$ has at least two visible neighbours, the 4-neighbourhood of $v_a$ is a maximal clique of $G$ and hence the label of some level-3 node $c^*$ in $C$. Clearly, the intersection of cglabel($c$) and cglabel($c^*$) results in another node $b^*$ of $C$, where cglabel($b^*$) = cglabel($c$) $\cap$ cglabel($c^*$). Moreover, cglabel($b^*$) is the 3-neighbourhood of $(v_c, v_a)$ which contains cglabel($a$) as a proper subgraph. Therefore, cglabel($a$) $\subset$ cglabel($b^*$) $\subset$ cglabel($c$), a contradiction to the initial assumption that $(a, c)$ is an edge in $C$, completing the proof. ▌

Finally, we can quantify the constants in the linear bounds on the number of vertices and edges in the clique graph of a 4-leaf power.

LEMMA 4.19. *If $G$ is the connected 4-leaf power of a tree $T$, then we can set $c_s = 5$ in line 4 of the clique graph algorithm and $c_e = 16$ in line 8.*

*Proof.* We first determine the number of internal vertices in $T$. For $n = |V(G)|$, there are $n$ leaves in $T$, each of which is associated with a visible internal vertex, for a total of at most $n$ visible internal vertices. The number of invisible internal vertices is also at most $n$, since each invisible vertex can be paired with a distinct visible vertex, as no two invisible vertices are adjacent.

To determine a bound $c_s$ on the total number of nodes in the clique graph, we use the fact that each node in the clique graph is a 2-, 3-, or 4-neighbourhood. In particular, there are at most $n$ 2-neighbourhoods (one for each visible vertex), at most $2n$ 3-neighbourhoods (one for each nonpendant edge) and at most $2n$ 4-neighbourhoods (one for each internal vertex). Thus, $c_s = 5$.

We count the edges between levels to determine $c_e = 16$. Each edge between a level-1 node and a level-2 node is either part of a three-level subgraph or is a peripheral linking edge, so by Lemma 4.4, point 4, these form a tree and hence number at most $5n$. Similarly, the edges between level-2 and level-3 nodes number at most $5n$.

We finally determine the number of edges that may be generated between level-1 and level-3 nodes (some of which will be subsequently deleted in the transitive reduction). There is an edge from a level-1 node (a 2-neighbourhood) to each 4-neighbourhood containing it. Since the 2-neighbourhood of a vertex $v$ is contained in exactly the 4-neighbourhoods of $v$ itself and all of $v$'s neighbours, the total number of edges is the sum over all internal vertices of the degree of the vertex plus one. This sum equals the number of internal vertices plus twice the number of nonpendant edges in $T$, and the total is at most $6n$, yielding $16n$ overall, as claimed. ▌

## 5. ALGORITHMS

We will briefly sketch the intuition behind our algorithms before giving details. For $k = 3$, our assumption that $G$ is connected makes its clique graph ideal. As a result, simple local replacement in the clique graph will construct a suitable tree. For $k = 4$, the clique graph is partitioned as in Theorem 4.1, each subgraph is transformed into a subtree by local replacement, and the subtrees are joined to form a single tree.

### 5.1. Algorithm for $k = 3$

The following simple algorithm and its justification are a good warmup for the more complicated $k = 4$ case.

```
1      form a clique graph C
2      if C has more than 2 levels, stop and answer NO
3      for each level-2 node a
4         if a does not have 1 or 2 parents
5            stop and answer NO
6      for each level-1 node a
7         if a has fewer than 2 children
8            stop and answer NO
9      for each level-1 node a
10        create a vertex t(a) labelled cglabel(a)
11     for each pair of level-1 nodes a and b with a common
12        child form the edge (t(a), t(b))
13     for each level-2 border node a with parent b
14        create a vertex t(a) labelled cglabel(a) \ cglabel(b)
15        form the edge (t(a), t(b))
16     form T by replacing each label by leaves
```

The correctness of the algorithm follows Lemma 4.3 and the following two lemmas.

LEMMA 5.1. *If leaves $u$ and $v$ are of distance at most three in $T$, then there exists an edge $(u, v)$ in $G$.*

*Proof.* Let $p$ and $q$ be the internal vertices associated with $u$ and $v$, respectively. Clearly, either $p = q$ or $p$ and $q$ are adjacent in $T$. Looking at how edges are created in $T$, there are two possibilities.

If $t^{-1}(p)$ and $t^{-1}(q)$ are both level-1 nodes, then they must have a common level-2 child, and both $u$ and $v$ are in the label of this child, meaning they are in the same maximal clique, ensuring the existence of $(u, v)$.

If they are not both level-1 nodes, then without loss of generality $t^{-1}(p)$ is a level-1 node and $t^{-1}(q)$ is a level-2 border node which is its child. Both $u$ and $v$ must be in the label of $t^{-1}(q)$, and the result follows. ▮

LEMMA 5.2. *If $(u, v)$ is an edge in $G$, then $u$ and $v$ are of distance at most three in $T$.*

*Proof.* The edge $(u, v)$ must appear in some maximal clique of $G$, and so both $u$ and $v$ appear in the label of a level-2 node $a$ in $C$.

If $a$ is a border node, then either both $u$ and $v$ are in the label of $t(a)$, they are both in the label of the parent $t(b)$, or one is in the label of $t(a)$ and one in the label of $t(b)$. In all cases, they are of distance at most three in $T$.

If $a$ is not a border node, by Lemma 4.3 it has exactly two parents $b$ and $c$, the union of whose labels is the label of $a$. Thus, $u$ and $v$ appear in one of the following: the same label, labels of a parent and a child, or labels of $t(b)$ and $t(c)$. Since there is an edge between $t(b)$ and $t(c)$ (created in line 12), in each case $u$ and $v$ must be of distance at most three. ▮

THEOREM 5.1. *Given a graph $G$ with $n$ vertices, it is possible in time $O(n^3)$ to determine whether or not $G$ is a 3-leaf power of a tree $T$, and if so, to determine such a $T$.*

### 5.2. Algorithm for $k = 4$

Rather than present the algorithm all at once, we will present it in sections, with justification following each section.

```
1      form a clique graph C
2      if C has more than 3 levels, stop and answer NO
```

The correctness of line 2 follows from Lemma 4.8. We now partition the clique graph into two-level components and three-level nondegenerate and degenerate components. Each nondegenerate three-level component is checked to see that it satisfies the necessary properties.

```
3      partition C into non. and deg. three-level and two-level
4      components for each nondegenerate three-level component N
5         for each level-1 node a in N
6            if a has only one child
                or any two children of a have more than one child
                   in common in N
                or if there is no level-3 node that is a child of
                   all children of a in N
7            stop and answer NO
8         for each level-2 node b in N
9            if b does not have exactly two children in N
                or if b has more than two parents
             stop and answer NO
10           if b has two parents a1 and a1 and
                cglabel(b) ≠ cglabel(a1) ∪ cglabel(a2)
             stop and answer NO
11        for each level-3 node c in N
12           if any two parents of c have more than one parent
                   in common in N
                or if there is no level-1 node that is a parent
                   of all parents of c in N
13           stop and answer NO
14        if the subgraph induced on levels 1 and 2 of N does
                not form a tree
             or the subgraph induced on levels 2 and 3 does
                not form a tree
             stop and answer NO
```

The correctness of line 7 follows from Lemma 4.4, points 2 and 3; line 9 from Lemma 4.4, point 2; line 10 from Lemma 4.5; line 13 from Lemma 4.4, point 3; and line 14 from Lemma 4.4, point 4.

We now create a subtree $T_N$ for each nondegenerate three-level component $N$.

```
16     for each N
17        create a subtree T_N (initially empty)
18        for each level-1 node a in N, create a vertex t(a)
             labeled cglabel(a)
19        if level-1 nodes a and b share a child, create
             the edge (t(a), t(b))
20        for each border node a with parent b
21           create a vertex t(a) in T_N labeled
                cglabel(a)\cglabel(b)
```

22 create the edge $(t(a), t(b))$
23 for each level-3 node $a$ with parents $b_1, \ldots, b_k$, $k \geq 2$ such that $A = \texttt{cglabel}(a) \backslash \{\cup_{i=1}^{k} \texttt{cglabel}(b_i)\}$ is nonempty
24 create a vertex $t(a)$ labeled $A$
25 for $d$ the common parent of $b_1, \ldots, b_k$, create edge $(t(a), t(d))$

To see that the graph formed is a tree, it suffices to observe that in $N$ the parent of each border node at level 3 is a border node at level 2, as shown in Lemma 4.6.

Next each degenerate three-level component $D$ is checked to ensure that it is a degenerate ideal clique graph, and from it a tree $T_D$ is formed. Figure 4 illustrates the subtrees derived from the degenerate and nondegenerate three-level components of Figure 3.

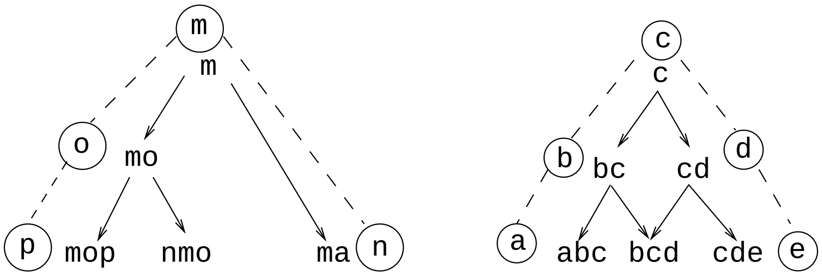


**FIG. 4.** Subtree derived from three-level components of Figure 3.

26 for each degenerate three-level component $D$
27 if there are more than two level-3 nodes, stop and answer NO
28 for the level-1 node $a$, create $t(a)$ labeled $\texttt{cglabel}(a)$
29 for the level-2 node $b$, create $t(b)$ labeled $\texttt{cglabel}(b) \backslash \texttt{cglabel}(a)$
30 for level-3 nodes $c$ and $d$, create $t(c)$ labeled $\texttt{cglabel}(c) \backslash \texttt{cglabel}(b)$ and create $t(d)$ labeled $\texttt{cglabel}(d) \backslash \texttt{cglabel}(b)$
31 create edges $(t(d), t(a))$, $(t(a), t(b))$, $(t(b), t(c))$
32 for $F$ the subgraph induced on nodes in $C$ not in any $N$ or $D$ in which each level 1 node is relabelled to be at level 2
if $F$ is not a forest of nodes at levels 2 and 3, stop and answer NO

The correctness of line 27 is a consequence of Theorem 4.1, point 5 and the correctness of line 32 is a consequence of Theorem 4.1, point 6.

33 for each tree $S$ in $F$
34 create a subtree $T_S$ (initially empty)
35 for each level-2 node $a$ in $S$, create a

```
              vertex t(a) labeled cglabel(a)
           for each level-3 node a such
           that A = cglabel(a)\ ⋃_b parent of a in S cglabel(b)
                 is empty
              create a vertex t(a) with the empty label
              for each parent b of a create an edge (t(a), t(b))
           for each level-3 node a such
           that A = cglabel(a)\ ∪_b parent of a in S cglabel(b)
                 is nonempty
              create a vertex t(a) labeled A
              create a vertex v_a with the empty label
              create the edge (t(a), v_a)
              for each parent b of a create an edge (t(b), v_a)
     form a labeled forest L from all T_N , T_D , and T_S
```

The subtrees derived from two-level components of the example clique graph $G$ can be seen in Figure 5.

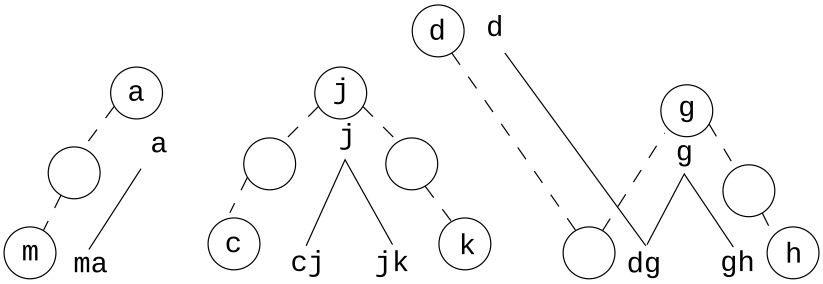


**FIG. 5.** Subtrees derived from two-level components of Figure 2.

```
     for each central linking edge (a, b), a in T_N or
           T_D , b in T_S
        add the edge (t(a), v_b)
        remove the label of t(a) from the label of t(b)
     for each peripheral or marginal linking edge (a, b),
           a in T_N or T_D , b in T_S
        identify t(a) and t(b)
     form T by replacing each label in L by leaves
```

Finally, we must show that the vertex $v_b$ exists whenever $(a, b)$ is a central linking edge, namely an edge between a level-1 node $a$ of a three-level subgraph and a level-3 node $b$ of a two-level subgraph. The vertex $v_b$ is created when cglabel$(b)$ contains nodes not found in its parent in $T_S$; the presence of the linking edge implies that such nodes exist (namely the label of cglabel$(a)$). Figure 6 shows the reconstructed tree for the running example. Although it is not identical to Figure 1 (as a clique graph can represent more than one possible tree), it is differs only in the absence of invisible vertices between $g$ and $h$ and between $j$ and $k$.

The correctness of the algorithm follows from Theorem 4.1 and the two lemmas below. The first shows that the 4-leaf power of the constructed tree $T$ is a subgraph of $G$. The second shows that $G$ is a subgraph of the 4-leaf power of $T$.

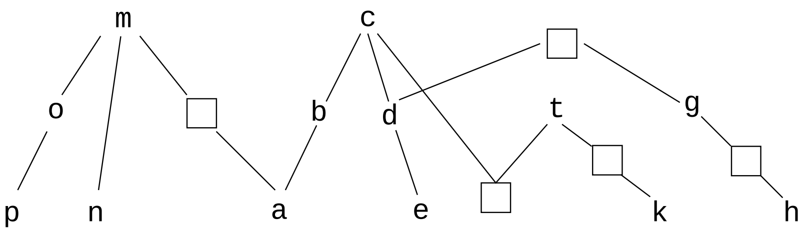


**FIG. 6.** Tree generated from clique graph of Figure 2 by reconstruction algorithm. As before, sets of leaf neighbours are indicated by letter labels, and invisible vertices by squares.

LEMMA 5.3. *If leaves $u$ and $v$ are of distance at most four in $T$, then there exists an edge $(u, v)$ in $G$.*

*Proof.* In the tree $T$ formed by the algorithm, we consider all possible parents $p$ and $q$ of $u$ and $v$ such that the distance between $p$ and $q$ is at most two, and show that in each case $(u, v)$ must be an edge in $G$. To show that $(u, v)$ is an edge in $G$, it suffices to find a node in the clique graph $C$ containing both $u$ and $v$ in its label.

We first observe that if $p$ and $q$ are at distance one in $T$, then $(u, v)$ is an edge in $G$. Edges between visible vertices are formed in lines 19, 22, 25, and 31. In these cases $t^{-1}(p)$ is either the parent or grandparent of $t^{-1}(q)$ (and hence the label of $t^{-1}(q)$ in the clique graph contained both $u$ and $v$) or $t^{-1}(p)$ and $t^{-1}(q)$ share a common child (which in turn contains both $u$ and $v$).

In the remainder of the proof we consider $p$ and $q$ at distance exactly two. For ease of exposition, we consider three cases, exhaustive up to the naming of $p$ and $q$: first, $p$ and $q$ both in some $T_N$ or $T_D$; next, both in some $T_S$; and finally, $p$ in some $T_N$ or $T_D$ and $q$ in some $T_S$. In each case, we use the fact that $p$ and $q$ are at distance at most two to find a node in the clique graph labeled with both $u$ and $v$, showing that $(u, v)$ is an edge in $G$.

**Case 1**: $p$ and $q$ in $T_N$ or $T_D$

The property is easy to verify for $T_D$; we now concentrate on $T_N$. In $T_N$, there are three types of vertices, *level*-1 *top vertices* (line 18), *border vertices* (line 21), and *grandchild vertices* (line 24).

**Case 1A**: $p$ and $q$ are both level-1 top vertices

If $p$ and $q$ are level-1 top vertices at distance two, they share a common neighbour $r$, where $r$ is also a level-1 top vertex. The edges $(p, r)$ and $(r, q)$ exist since $t^{-1}(p)$ and $t^{-1}(r)$ share a child $b$ and $t^{-1}(r)$ and $t^{-1}(q)$ share a child $b'$. By Lemma 4.4, point 3, the children of $t^{-1}(r)$ (in particular $b$ and $b'$) must share a child $c$, which contains both $u$ and $v$ as labels in the clique graph.

**Case 1B**: $p$ is a grandchild vertex

By Lemma 4.4, point 3 and line 24, a level-1 top vertex can be the neighbour of at most one grandchild. Thus, if $p$ is a grandchild vertex, then $q$ is either a level-1 top vertex or a border vertex.

When $q$ is a level-1 top vertex at distance 2 from $p$, there is a path in $T_N$ from $p$ to $q$ through $r$ where $t^{-1}(r)$ is the common grandparent of $t^{-1}(p)$'s parents. Since $r$ and $q$ are connected by an edge, clearly $t^{-1}(r)$ and $t^{-1}(q)$ have a common child $b$ (line 19). Since $t^{-1}(p)$ is the common grandchild of all of the children of $t^{-1}(r)$ (Lemma 4.4, point 3), $t^{-1}(q)$ is the grandparent of $t^{-1}(p)$, and hence $u$ and $v$ both appear in the label of guaranteeing that $t^{-1}(p)$.

If $q$ is a border vertex, there is path from $t^{-1}(p)$ to $t^{-1}(q)$ via $t^{-1}(p)$'s grandparent $a$, which is the parent of $t^{-1}(q)$. Since all children of $a$ share a common child $t^{-1}(p)$, $t^{-1}(q)$ is also a parent of $t^{-1}(p)$. Thus $u$ and $v$ are both in the label of $t^{-1}(p)$ in the clique graph, completing this case.

**Case 1C**: $p$ is a border vertex

The case in which $q$ is a grandchild vertex is handled in Case 1B above.

If both $p$ and $q$ are level-2 border vertices, $t^{-1}(p)$ and $t^{-1}(q)$ share a common parent and hence a common child containing both $u$ and $v$ is its label. We observe that $p$ and $q$ cannot both be level-3 border vertices, for the parent of $t^{-1}(p)$ and $t^{-1}(q)$, having exactly these two children (Lemma 4.4, point 2) would not share any child with its siblings, violating Lemma 4.4, point 3.

If $p$ is a level-3 border vertex and $q$ is a level-1 top vertex, then $q$ is the parent of $p$'s parent, a level-2 border vertex (Lemma 4.6). In this case, $u$ and $v$ are both in the label of $t^{-1}(p)$.

Finally, if $q$ is a level-1 top vertex that is a neighbour of $p$'s level-1 top vertex parent $r$, then $t^{-1}(q)$ and $t^{-1}(r)$ share a child $b$ and $t^{-1}(p)$ and $b$ share a child $c$. In the clique graph, $c$ is labeled with both $u$ and $v$, as needed.

**Case 2**: $p$ and $q$ in $T_S$

In $T_S$, the vertices with nonempty labels include *top vertices* (line 35) and *bottom vertices* (line 40). As before, we consider pairs of vertices at distance exactly two. Since each bottom vertex has a distinct parent, one of the vertices, say $p$, must be a top vertex.

If $q$ is a top vertex at distance exactly two from $p$, there exists a level-3 node containing the union of cglabel($p$) and cglabel($q$) (line 36 or 39), and hence $u$ and $v$ are in the label of a node in the clique graph, as needed. If instead $q$ is a bottom vertex at distance two, $t^{-1}(q)$ is a child of $t^{-1}(p)$ and hence $u$ and $v$ are in cglabel($q$).

**Case 3**: $p$ in $T_N$ or $T_D$ and $q$ in $T_S$

It suffices to consider $p$ in $T_N$ in the remainder of the discussion below, since the argument is identical for $p$ in $T_D$. We first suppose that $T_N$ and $T_S$ are joined by a marginal or peripheral edge. If $p$ or $q$ is one of the endpoints of the linking edge, then both $p$ and $q$ were in either $T_N$ or $T_S$, and hence covered by cases 1 and 2 above. If instead neither $p$ nor $q$ is an endpoint of the linking edge, then one, say $q$, must be a node in $T_S$ adjacent to the endpoint $r$ in $T_S$. By definition of marginal and peripheral, $t^{-1}(r)$ is at level 2, and by construction (lines 37–38 and 40–43), $t^{-1}(q)$ has an empty label, contradicting the assumption that $q$ is the parent of $u$.

We now suppose that $T_N$ and $T_S$ are joined by a central edge, in which case both $p$ and $q$ must both be neighbours of $v_b$ with $u$ and $v$ both being in cglabel($b$). This concludes the demonstration that any two leaves of distance four in $T$ have an edge between them in $G$. ∎

We next show that for each edge $(u, v)$ in $G$ there is a path from $u$ to $v$ in $T$ of distance at most four. If $(u, v)$ is an edge in $G$, then there exists a maximal clique containing $(u, v)$; we consider the assignment of labels of maximal cliques to vertices in $T$.

LEMMA 5.4. *If $(u, v)$ is an edge in $G$, then $u$ and $v$ are of distance at most four in $T$.*

*Proof.* Since $u$ and $v$ are leaves of $T$, it suffices to show that their associated internal vertices are of distance at most two in $T$. The edge $(u, v)$ must be in some maximal clique of $G$, and so $u$ and $v$ appear together in some label of a level-3 node $a$ in $C$. We omit the trivial cases in which $u$ and $v$ end up in the same label at line 50, and hence share an associated internal vertex. The proof proceeds by looking at what happens to node $a$ during the algorithm. In some cases, a vertex $t(a)$ is created; in others, it is not.

If $t(a)$ does not exist, then $a$ must be a level-3 node in a three-level ideal subgraph $N$ of $C$ with parents $b_1, \ldots, b_k$, $k \geq 2$ such that the union of the labels of $b_1, \ldots, b_k$ is exactly the label of $a$ (this is the negation of the conditions at lines 20 and 23). The nodes $b_1, \ldots, b_k$ share a common parent $d$ (Lemma 4.4, point 3); each $b_i$ may have another parent $c_i$ as well. The vertices $u$ and $v$ must occur somewhere in the labels of the $b_i$'s and therefore somewhere in the labels of the $t(c_i)$ vertices (created at line 18), say $u$ in $t(c_j)$ and $v$ in $t(c_{j'})$ or $t(d)$. Since the edges $(t(d), t(c_j))$ and $(t(d), t(c_{j'}))$ are created at line 19, the associated internal vertices of $u$ and $v$ are of distance at most two in $T$.

We now suppose that $t(a)$ exists and consider its roles, as a level-3 vertex, in $T_N$, $T_D$, and $T_S$. We observe that we can ignore marginal linking edges, since if there were one incident to $a$, the node $d$ on the other end would result in a vertex $t(d)$ being created which would have the same label as $t(a)$ and be identified with it.

If $t(a)$ is a border vertex (created at line 21 from $T_N$), then $u$ is in the label of $t(a)$ and $v$ is in the label of $t(b)$, for $b$ either the parent or grandparent of $a$ in the same component $N$. If $b$ is the parent of $a$, $t(a)$ and $t(b)$ (the latter created at line 18) are adjacent due to the edge created at line 22; if $b$ is the grandparent, $t(a)$ and $t(b)$ are at distance two due to edges created at line 22.

If $t(a)$ is a grandchild vertex (created at line 24 from $T_N$), then at most one of $u$ and $v$ is in the label of $t(a)$. We suppose $v$ is in the label of some parent $b_1$ of $a$ and $u$ is in the label of $t(a)$, in the label of $b_1$, or in the label of a parent $b_2 \neq b_1$ of $a$. For each $b_i$ we consider two cases, depending on the number of parents it has, which is either one or two, by Lemma 4.4, point 2. For ease of exposition, we let $d$ be the common parent of $b_1$ and $b_2$ (Lemma 4.4, point 3) and $c_i$ be another parent of $b_i$, if it exists.

If $b_i$ has one parent, then a node $t(b_i)$ is created at line 21, and since the parent $d$ of $b$ is a level-1 node, $t(d)$ is created at line 18. There are edges $(t(b_i), t(d))$ (created at line 22) and $(t(b_i), t(d))$ (created at line 25). If $b_i$ has two parents, then $t(c_i)$ exists and since $t(c_i)$ and $t(d)$ are adjacent, there is an edge $(t(c_i), t(d))$. Taking the two cases together, $u$ is in $t(a)$, $t(b_1)$, $t(c_1)$, $t(b_2)$, $t(c_2)$, or $t(d)$, and $v$ is in $t(b_1)$, $t(c_1)$, or $t(d)$. There is a path of length at most two between any of the two sets of vertices (via $t(d)$), and hence $u$ and $v$ are at distance at most four.

The case in which $t(a)$ is in $T_D$ follows from the observation that the only vertices in $T_D$ at distance more than two are $t(c)$ and $t(d)$, derived from two different level-3 nodes. Clearly $u$ and $v$ cannot be found in these two locations.

If $t(a)$ is an empty bottom vertex (created at line 37 from $T_S$), then $a$ cannot have an incident central linking edge, as the label of the vertex at the other end of that edge would falsify the condition of line 36. We can thus find $u$ in the label $t(b)$ and $v$ in the label $t(c)$ for $b$ and $c$ both parents of $a$. Since $t(b)$ and $t(c)$ are both adjacent to $t(a)$ by edges created at line 38, they are at distance at most two, as needed.

Finally, if $t(a)$ is a nonempty bottom vertex (created at line 40 from $T_S$), $u$ and $v$ can each be in any of three places: in the label of $t(a)$, in the label of $t(b)$ for some parent $b$ of $a$, or possibly in the label of $t(d)$ for some central linking edge $(d, a)$. Each of these vertices is adjacent to the vertex $v_a$ created at line 41, by edges created at lines 42, 43, and 46 respectively. Hence they are at distance two from each other, as needed to complete the proof. ∎

THEOREM 5.2. *Given a graph $G$ with $n$ vertices and $e$ edges, it is possible in time $O(n^3)$ to determine whether or not $G$ is a 4-leaf power of a tree $T$, and if so, to determine such a $T$.*

*Proof.* We have seen that clique graph generation takes $O(n^3)$ time, and produces a clique graph with $O(n)$ vertices and edges; partitioning and local replacement then clearly can be completed in time $O(n)$. ∎

## 6. CONCLUSIONS AND FURTHER WORK

Reconstructing the $k$-leaf powers of ideal trees for $k > 4$ would be easy (by generalizing the part of the reconstruction algorithm that deals with nondegenerate three-level subgraphs) but less interesting than handling more general trees. We believe the clique graph approach offers promise for the general case, though more work is needed to quantify exactly how collapses occur as a result of invisible vertices. The algorithm to create the clique graph works for all $k$, as do the general technical lemmas on neighbourhoods and some of the characterizations of clique graph structure. The main stumbling block appears to be the combinatorial explosion in the number of cases in the analysis of the extension of results like Theorem 4.1, which may be controlled by discovery of further general structure. It might also be possible to extend these techniques to consider the case of weighted edges in the tree $T$.

Among the objections to practical use of the algorithms is that the number of trees that correspond to a particular $k$-leaf power could be very large. It might be interesting to determine all corresponding trees, or perhaps all trees that satisfy a given set of additional constraints.

## ACKNOWLEDGMENT

We would like to thank Paul Kearney for suggesting this problem to us, and the anonymous referees for SWAT 2000 and this journal.

## REFERENCES

1. J.-P. Barthélemy and A. Guénoche. *Trees and Proximity Representations.* John Wiley and Sons, 1991.
2. D. G. Corneil and P. Kearney. Tree powers. *Journal of Algorithms*, 29:111–131, 1998.
3. R. Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set for chordal graphs. *SIAM Journal on Computing*, 1:180–187, 1972.
4. M. R. Henzinger, V. King, and T. Warnow. Constructing a tree from homeomorphic subtrees, with applications to computational evolutionary biology. *Algorithmica*, 24(1):1–13, 1999.
5. D. H. Huson, K. A. Smith, and T. Warnow. Estimating large distances in phylogenetic reconstruction. In *Proceedings of the 3rd International Workshop on Algorithm Engineering*, volume 1668 of *Lecture Notes in Computer Science*, pages 270–285, 1999.
6. P. E. Kearney. The relationship between a phylogeny and its ordinal assertions. Technical Report 308/97, University of Toronto, Department of Computer Science, 1997.
7. J. Kim and T. Warnow. Tutorial on phylogenetic tree estimation. manuscript, Department of Ecology and Evolutionary Biology, Yale University, `http://ismb99.gmd.de/TUTORIALS/Kim/4KimTutorial.ps`, 1999.
8. N. Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21:193–201, 1992.
9. R. Motwani and M. Sudan. Computing roots of graphs is hard. *Discrete Applied Mathematics*, 54:81–88, 1994.
10. D. J. Rose, R. E. Tarjan, and G. S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing*, 5(2):266–283, 1976.
11. R. E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 13:566–579, 1984.