Linear-Time Algorithms for Tree Root Problems

Maw-Shang Chang[1,⋆], Ming-Tat Ko[2], and Hsueh-I Lu[3,⋆⋆]

[1] Department of Computer Science and Information Engineering,
National Chung Cheng University, Ming-Shiun, Chiayi 621, Taiwan
mschang@cs.ccu.edu.tw
[2] Institute of Information Science,
Academia Sinica, Taipei 115, Taiwan
mtko@iis.sinica.edu.tw
[3] Department of Computer Science and Information Engineering,
National Taiwan University, Taipei 106, Taiwan
hil@csie.ntu.edu.tw
http://www.csie.ntu.edu.tw/~hil

Abstract. Let T be a tree on a set V of nodes. The *p-th power* T^p of T is the graph on V such that any two nodes u and w of V are adjacent in T^p if and only if the distance of u and w in T is at most p. Given an n-node m-edge graph G and a positive integer p, the *p-th tree root problem* asks for a tree T, if any, such that $G = T^p$. Given a graph G, the *tree root problem* asks for a positive integer p and a tree T, if any, such that $G = T^p$. Kearney and Corneil gave the best previously known algorithms for both problems. Their algorithm for the former (respectively, latter) problem runs in $O(n^3)$ (respectively, $O(n^4)$) time. In this paper, we give $O(n + m)$-time algorithms for both problems.

1 Introduction

Let H be a graph on a set V of nodes. The *p-th power* H^p of H is the graph on V such that any two nodes u and w of V are adjacent in H^p if and only if the distance of u and w in H is at most p. If $G = H^p$, then we say that graph H is a *p-th root* of graph G or, equivalently, G is the *p-th power* of H. Determining whether the input graph is a power of some other graph is the *graph root problem*. Graph roots and powers have been extensively studied in the literature. Motwani and Sudan [12] proved the NP-completeness of recognizing squares of graphs. Lau [9] showed that squares of bipartite graphs can be recognized in polynomial time and proved the NP-completeness of recognizing cubes of bipartite graphs. Lau and Corneil [10] also studied the tractability of recognizing powers of proper interval, split, and chordal graphs. Lin and Skiena [11] gave a linear-time algorithm to find square roots of planar graphs.

If $G = T^p$ for some tree T and number p, we call such a tree T a *p-th tree root* of G. Given a graph G and a positive integer p, the *p-th tree root problem*

⋆ The author thanks the Institute of Information Science of Academia Sinica of Taiwan for their hospitality and support where part of this research took place.

⋆⋆ Corresponding author.

L. Arge and R. Freivalds (Eds.): SWAT 2006, LNCS 4059, pp. 411–422, 2006.

asks for a tree T, if any, with $G = T^p$. Given a graph G, the *tree root problem* asks for a tree T and a number p, if any, with $G = T^p$. Ross and Harary [14] characterized squares of trees and showed that square tree roots, when they exist, are unique up to isomorphism. Lin and Skiena [11] gave a linear-time algorithm to recognize squares of trees. Kearney and Corneil [7] gave the best previously known algorithms for the p-th tree root problem and the tree root problem. Their algorithm for the p-th tree root problem runs in $O(n^3)$ time for any n-node graph, leading to an $O(n^4)$-time algorithm for the tree root problem. Gupta and Singh [4] gave a characterization of graphs which are the p-th powers of trees and proposed a heuristic algorithm to construct a p-th tree root. The running time of their algorithm is $O(n^3)$, but they did not prove the correctness of their algorithm. It was unknown whether the p-th tree root problem can be solved in $o(n^3)$ time [7, 9].

In this paper we improve Kearney and Corneil's result [7] by giving linear time algorithms, in the size of the input graph, for the tree root problem as well as the p-th tree root problem for any given p. For the p-th tree root problem, our linear-time algorithm processes the input graph in two different but analogous ways, depending on whether p is even. If p is even, our algorithm is based upon a new observation that the p-th power T^p of a tree T is a chordal graph admitting a unique clique tree which is isomorphic to the $\frac{p}{2}$-centroid $T(\frac{p}{2})$ of T. Since it takes linear time to compute a clique tree for a chordal graph, $T(\frac{p}{2})$ can thus be obtained efficiently. To determine the remaining topology of T, we resort to a linear-time computable *clique-position function* for all nodes. We also prove that the existence of clique-position function is a new characterization for graphs admitting tree roots. To be more specific, there is a one-to-one mapping between the nodes of $T(\frac{p}{2})$ and the maximal cliques of T^p. The clique-position for a node u of T can be used to determine the distance between u and the node w in $T(\frac{p}{2})$ that is closest to u in T. Having determined the clique positions of all nodes, we grow the remaining tree topology from the outermost layer to the innermost layer. As for the case that p is odd, our algorithm works in an analogous way. In particular, the minimal node separators of T^p plays the role of maximal cliques of T^p for the case that p is even. The separator tree of T^p is unique and isomorphic to the $\frac{p+1}{2}$-centroid $T(\frac{p+1}{2})$ of T. The remaining topology of T can be determined by a linear-time computable *separator-position function* of T^p.

Our linear-time algorithm for the p-th tree root problem immediately yields a quadratic-time algorithm for the tree root problem. Deriving from (1) the diameters of the clique tree and separator tree of the input graph and (2) the clique positions and separator positions of nodes in the input graph, we also show a linear-time algorithm for finding the minimum p, if any, such that the input graph admits a p-th tree root. As a result, we have a linear-time algorithm for the tree root problem.

Due to space limit, the case for odd p is omitted in this extended abstract. The rest of the paper is organized as follows. Section 2 gives the preliminaries. Section 3 gives our linear-time algorithm for the p-th tree root problem for the

case that p is even. Section 4 gives our linear-time algorithm for finding the smallest even number p such that the input graph admits a p-th tree root.

2 Preliminaries

For any set S, let $|S|$ denote the cardinality of S. All graphs in this paper are undirected, simple, and have no self-loops. Let $V(G)$ (respectively, $E(G)$) consist of the nodes (respectively, edges) of graph G. For any subset U of $V(G)$, let $G[U]$ denote the subgraph of G induced by U. A node is *dominating* in G if it is adjacent to all other nodes in G. Let $Dom(G)$ consist of the dominating nodes of G, which can be computed from G in linear time.

2.1 Notation for Trees

Let T be a tree. Let $Path_T(u, w)$ denote the path of T between u and w. Let $d_T(u, w)$ denote the distance of nodes u and w in T. Define $d_T(u) = \max_{w \in V(T)} d_T(u, w)$. The *diameter* d_T of a tree T is $\max_{u \in V(T)} d_T(u)$. Let $\Gamma_T(u, i)$ consist of the nodes w with $d_T(u, w) \leq i$.

Define the *i-centroid* $T(i)$ of T as follows. Let $T(0) = T$. For each i with $1 \leq i \leq \frac{d_T}{2}$, let $T(i)$ be the tree obtained by deleting the leaf nodes of $T(i-1)$. If $i > \frac{d_T}{2}$, then $T(i)$ is the empty graph. The *centroid* $Cent(T)$ of T is $T(\lfloor \frac{d_T}{2} \rfloor)$, which is either a single node or a single edge. The centroid of T can be computed from T in $O(|V(T)|)$ time.

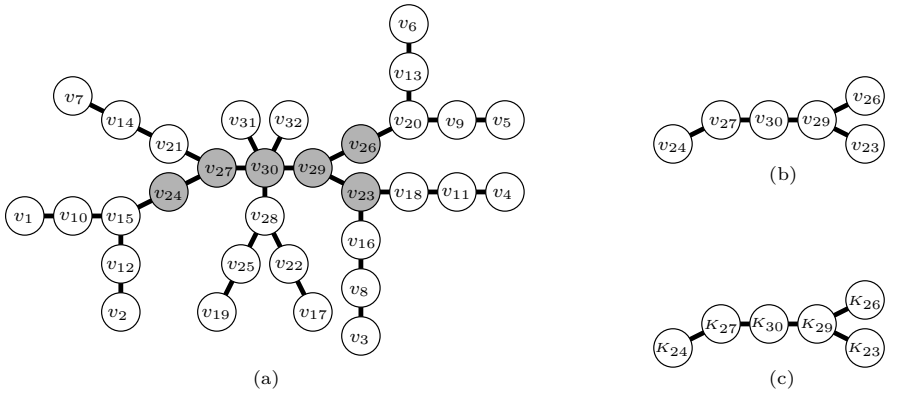

Fig. 1. (a) A tree T. (b) The 3-centroid $T(3)$ of T. (c) The clique tree of T^6, where $K_i = \Gamma_T(v_i, 3)$.

We use the tree T shown in Figure 1(a), which also appeared in [7], to illustrate the aforementioned notation: $\Gamma_T(v_{10}, 3) = \{v_1, v_2, v_{10}, v_{12}, v_{15}, v_{24}, v_{27}\}$. The 3-centroid $T(3)$ of T, as shown in Figure 1(b), is the subtree of T induced by $\{v_{24}, v_{27}, v_{30}, v_{29}, v_{26}, v_{23}\}$,

2.2 Chordal Graphs

A graph G is *chordal* if it contains no induced subgraph which is a cycle of size greater than three. Chordal graphs, which can be recognized in linear time [15, 13], have been extensively studied in the literature. It is well known that any tree power is chordal (see, e.g., [7]).

A subset S of $V(G)$ is a *separator* of a connected graph G if $G[V(G) \setminus S]$ has at least two connected components. A separator S of G is *minimal* if any proper subset of S is not a separator of G. A separator S of G is a (u, w)-*separator* of G if nodes u and w are in different connected components of $G[V(G) \setminus S]$. A (u, w)-separator S is *minimal* if any proper subset of S is not a (u, w)-separator of G. A *minimal node separator* is a minimal (u, w)-separator for some nodes u and w. Note that a minimal node separator is not necessarily a minimal separator, as the minimal (u, w)-separator may contain the minimal (x, y)-separator for some other nodes x and y. Graph G is chordal if and only if every minimal node separator of G induces a clique in G [2].

Let G be a chordal graph. Let $\mathcal{K}_G$ consist of the maximal cliques of G. For each node u of G, let $\mathcal{K}_G(u)$ consist of the maximal cliques of G containing u. A *clique tree* of a chordal graph G is a tree $\mathcal{T}$ with $V(\mathcal{T}) = \mathcal{K}_G$ such that each $\mathcal{K}_G(u)$ with $u \in V(G)$ induces a subtree of $\mathcal{T}$. Gavril [3] and Buneman [1] ensured that graph G is chordal if and only if G has a clique tree. Moreover, a clique tree of any chordal graph G can be computed in $O(|V(G)| + |E(G)|)$ time [6]. A chordal graph may have more than one clique tree [5]. A chordal graph is *uniquely representable* [8] if it admits a unique clique tree.

Lemma 1. *Let $\mathcal{T}$ be a clique tree of a chordal graph G. Then, S is a minimal node separator of G if and only if $S = K_1 \cap K_2$ for some edge (K_1, K_2) of $\mathcal{T}$.*

Proof. For each edge e of $\mathcal{T}$, let S_e consist of the nodes u of G such that e belongs to the subtree of $\mathcal{T}$ induced by $\mathcal{K}_G(u)$. That is, $S_e = \{u \in V(G) \mid e \in E(\mathcal{T}[\mathcal{K}_G(u)])\}$. Ho and Lee [5] ensured that S is a minimal node separator of G if and only if $S = S_e$ for some $e \in E(\mathcal{T})$. Therefore, it remains to ensure $S_{(K_1,K_2)} = K_1 \cap K_2$ by verifying that (K_1, K_2) is an edge of $\mathcal{T}[\mathcal{K}_G(u)]$ if and only if $\{K_1, K_2\} \subseteq \mathcal{K}_G(u)$ if and only if $u \in K_1 \cap K_2$. □

3 The p-th Tree Root Problem for any Given Even p

This section assumes that the given positive number $p \leq n$ is even. Let $h = \frac{p}{2}$.

3.1 Unique Representability

This subsection shows that if T is a tree, then T^p has a unique clique tree, which has to be isomorphic to $T(h)$.

Lemma 2. *For any tree T, we have that $G = T^p$ if and only if $\mathcal{K}_G = \{\Gamma_T(u, h) \mid u \in V(T(h))\}$.*

Proof. Observe that nodes u and w are adjacent in G if and only if there exists a maximal clique K of G that contains both u and w. Therefore, for any graphs G and H, we have that $G = H$ if and only if $\mathcal{K}_G = \mathcal{K}_H$. Gupta and Singh [4] proved that K is a maximal clique of T^p if and only if there exists a node u of $T(h)$ with $\Gamma_T(u,h) = K$. The lemma is proved. □

Theorem 1. *If T is a tree, then T^p has a unique clique tree. Moreover, the clique tree of T^p is isomorphic to $T(h)$.*

Proof. Let $\mathcal{T}$ be the tree with

$$V(\mathcal{T}) = \{\Gamma_T(u,h) \mid u \in V(T(h))\};$$
$$E(\mathcal{T}) = \{(\Gamma_T(u,h), \Gamma_T(w,h)) \mid (u,w) \in E(T(h))\}.$$

It is not hard to verify that $T(h)$ and $\mathcal{T}$ are isomorphic. We first show that $\mathcal{T}$ is a clique tree of T^p. Observe that by Lemma 2, we have

$$\begin{aligned}\mathcal{K}_{T^p}(u) &= \{\Gamma_T(w,h) \mid w \in V(T(h)), d_T(u,w) \le h\} \\ &= \{\Gamma_T(w,h) \mid w \in V(T(h)) \cap \Gamma_T(u,h)\}.\end{aligned}$$

Since $T[V(T(h)) \cap \Gamma_T(u,h)]$ is a subtree of $T(h)$, $\mathcal{T}[\mathcal{K}_{T^p}(u)]$ is a subtree of $\mathcal{T}$. Therefore, $\mathcal{T}$ is a clique tree of T^p.

To show that T^p has no other clique tree, we resort to an observation of Kumar and Madhavan [8] stating that a chordal graph G is uniquely representable if (and only if) every minimal node separator of G is contained in exactly two maximal cliques of G. By Lemmas 1 and 2, each minimal node separator of T^p has the form $\Gamma_T(u,h) \cap \Gamma_T(w,h)$ for some nodes u and w adjacent in $T(h)$. Observe that $\Gamma_T(u,h) \cap \Gamma_T(w,h) \not\subseteq \Gamma_T(x,h)$ for any node x of $T(h)$ other than u and w. The theorem is proved. □

3.2 Clique-Position Function

Let G be a uniquely representable chordal graph. Let $\mathcal{T}$ be the clique tree of G. We say that (K,i) is a *clique position* of u in G if $\mathcal{K}_G(u) = \Gamma_{\mathcal{T}}(K,i)$, i.e., the maximal cliques containing u are exactly those nodes in the clique tree $\mathcal{T}$ that are at a distance up to i from K. For notational brevity, we also write $u \in \Pi_G(K,i)$ to signify that (K,i) is a clique position of u in G, where the subscript G may be omitted when it is clear from the context. Let us use Figure 1 to explain this crucial concept of the paper. For each index i with $v_i \in V(T(3))$, let $K_i = \Gamma_T(v_i, 3)$. By Lemma 2, we know that these K_i are the maximal cliques of T^6. Observe that

$$\mathcal{K}_G(v_{14}) = \{K_{24}, K_{27}, K_{30}\} = \Gamma_{\mathcal{T}}(K_{27}, 1) = \Gamma_{\mathcal{T}}(K_{24}, 2).$$

Therefore, both $(K_{27}, 1)$ and $(K_{24}, 2)$ are clique positions of v_{14} in T^6. One can also verify that $v_{15} \in \Pi_{T^6}(K_{27}, 1) \cap \Pi_{T^6}(K_{24}, 2)$.

A *clique-position function of G with respect to p* is a function $\Phi : V(G) \to \mathcal{K}_G \times \{0, 1, \ldots, h\}$ that satisfies the following conditions.

Condition $C1$: For each $u \in V(G)$, $\Phi(u)$ is a clique position of u in G.
Condition $C2$: For each $K \in \mathcal{K}_G$, there exists a unique node u of G with $\Phi(u) = (K, h)$.
Condition $C3$: If $\Phi(u) = (K, i)$ for some $K \in \mathcal{K}_G$ and $i < h$, then there is a node w of G with $\Phi(w) = (K, i+1)$.

Given a chordal graph T^p, an integer p, and the clique tree $\mathcal{T}$ of T^p, the rest of the subsection shows how to compute a clique-position function of T^p with respect to p. Note that our algorithm does not know T.

For each node $u \in V(T) \setminus V(T(h))$, define $\ell(u)$ to be the largest index i with $i \leq h-1$ such that u belongs to $\Pi(K, i)$ for some maximal clique K of T^p. If $u \in V(T) \setminus V(T(h))$, then $u \in \Pi(\kappa(w), h - d_T(u, w))$, where w is the node of $T(h)$ that is closest to u in T. Therefore, $\ell(u)$ is well defined for each node $u \in V(T) \setminus V(T(h))$. To simplify the description of our algorithm, each node u of T^p is initially white, signifying that $\Phi(u)$ is still undefined. If $\Phi(u)$ is defined but may be changed later, then u is gray. If $\Phi(u)$ is defined and will not be changed later, then u is black. Our algorithm is as shown in Algorithm 1, whose correctness is ensured by the following lemma.

Input: A positive even number p, and a graph T^p and its clique tree $\mathcal{T}$.
Output: A clique-position function Φ of T^p with respect to p.

1: For each node u of T^p, compute the clique positions of u in T^p and let u be white.
2: For each $K \in \mathcal{K}_{T^p}$, choose a white node $u \in \Pi(K, h)$, let $\Phi(u) = (K, h)$, and let u be black.
3: Let K^* be a maximal clique of T^p in $V(\mathit{Cent}(\mathcal{T}))$.
4: For each white node $u \in \mathit{Dom}(T^p)$, let $\Phi(u) = (K^*, h-1)$ and let u be gray.
5: For each white node $u \in V(T^p) \setminus \mathit{Dom}(T^p)$, compute $\ell(u)$.
6: **while** there are still white nodes in $V(T^p) \setminus \mathit{Dom}(T^p)$ **do**
7: Let u be a white node in $V(T^p) \setminus \mathit{Dom}(T^p)$ with the smallest $\ell(u)$.
8: Let $\kappa(u)$ be a maximal clique of T^p with $u \in \Pi(\kappa(u), \ell(u))$.
9: Let $\Phi(u) = (\kappa(u), \ell(u))$ and let u be black.
10: For each white node $w \in \Pi(\kappa(u), \ell(u))$, let $\Phi(w) = (\kappa(u), \ell(u))$ and let w be gray.
11: **for** $j = \ell(u) + 1$ to $h - 1$ **do**
12: Choose a non-black node $w \in \Pi(\kappa(u), j)$. Let $\Phi(w) = (\kappa(u), j)$ and let w be black.
13: For each white node $w \in \Pi(\kappa(u), j)$, let $\Phi(w) = (\kappa(u), j)$ and let w be gray.
14: **end for**
15: **end while**

Algorithm 1: Computing a clique-position function of T^p with respect to p

Lemma 3. *Algorithm 1 correctly computes a clique-position function of T^p with respect to p.*

Proof. Our proof is based upon the facts that T is one of the p-th tree roots of T^p and the clique tree $\mathcal{T}$ of T^p is isomorphic to $T(h)$. As we will see, the challenge

of the proof lies in showing that the algorithm does not abort at Steps 2 and 12. Let us first assume that the algorithm does not abort at Steps 2 and 12, and show that the function Φ computed by the algorithm has to be a clique-position function of T^p.

- Condition C1. One can verify that the algorithm assigns $\Phi(u) = (K, i)$ only if $u \in \Pi(K, i)$. It is also easy to see that $\Phi(u)$ is defined for every node of T, i.e., no white nodes left, at the end of the algorithm. Thus, Condition C1 holds for the function Φ computed by the algorithm.
- Condition C2. Since the algorithm does not abort at Step 2, the algorithm successfully assigns clique positions for $|\mathcal{K}_{T^p}|$ nodes at the end of Step 2. Since the rest of the algorithm never assigns (K, h) to any $\Phi(u)$, Condition C2 holds for the function Φ computed by the algorithm.
- Condition C3. By Condition C2 of Φ, we know that Condition C3 holds for each node processed at Step 4. For each iteration of the while-loop (Steps 6–15), we first let $\Phi(u) = (\kappa(u), \ell(u))$ and turn u into black. Then, in the for-loop (Steps 11–14), for each index j with $\ell(u) < j < h$, the algorithm assigns $\Phi(w) = (\kappa(u), j)$ for some non-black node w and turns w into black. Therefore, as long as the algorithm does not abort at Step 12, one can see that Condition C3 holds for each node not processed at Step 4.

We then show that the algorithm does not abort at Step 2. Observe that each node u of $T(h)$ belongs to $\Pi(\Gamma_T(u, h), h)$. By Lemma 2, for each maximal clique K of T^p, the number of maximal cliques K' of T^p with $\Gamma_{\mathcal{T}}(K', h) = \Gamma_{\mathcal{T}}(K, h)$ is no more than $|\Pi(K, h)|$. Therefore, Step 2 successfully assigns clique positions for $|\mathcal{K}_{T^p}|$ nodes.

It remains to prove that the algorithm does not abort at Step 12. We first show that if an iteration of the while-loop enters the for-loop, then the node u of the iteration has a unique clique position in T^p. We can focus only on the case with $1 \leq \ell(u) \leq h-2$, because (1) if $\ell(u) = 0$, then u has a unique clique position in T^p, and (2) if $\ell(u) \geq h-1$, then the algorithm does not enter the for-loop. We also observe that $\kappa(u)$ cannot be a leaf of $\mathcal{T}$. The reason is that if $\kappa(u)$ is a leaf of $\mathcal{T}$, then T has at least one leaf whose unique clique position in T^p is $(\kappa(u), 0)$. Since the while-loop processes nodes u in non-decreasing order of $\ell(u)$, $\ell(u) \geq 1$ implies that u cannot be white at the beginning of the iteration of the while-loop. Let v be the node of $T(h)$ such that $\kappa(u) = \Gamma_T(v, h)$. Since $\kappa(u)$ is not a leaf of $\mathcal{T}$, node v is not a leaf of $T(h)$. See Figure 2 for an illustration for the proof. Let $S = \Gamma_{T(h)}(v, \ell(u))$. Since $\ell(u) \geq 1$ and u is not a dominating node of T^p, there is an edge (x, y) of $T(h)$ such that $y \in S - \{v\}$ and $x \notin S$. Since v is not a leaf of $T(h)$, there has to be a neighbor w of v in $T(h)$ such that $Path_{T(h)}(w, y)$ contains v. Since $\ell(u) \leq h-2$, we know $\Gamma_{T(h)}(w, \ell(u)+1) \neq S$. There has to be an edge (x', y') of $T(h)$ such that $y' \in S$, $x' \notin S$, and $Path_{T(h)}(x', x)$ contains y, y', w, and v. By the existence of edges (x, y) and (x', y') of $T(h)$, we know that $S = \Gamma_{T(h)}(z, j)$ implies $z = v$ and $j = \ell(u)$. Thus, $(\kappa(u), \ell(u))$ is the unique clique position of u in T^p.

Let u_i be the node u for the i-th iteration of the while-loop that enters the for-loop. We already showed that $(\kappa(u_i), \ell(u_i))$ is the unique clique position of u_i

in T^p. Let v_i be the node with $\kappa(u_i) = \Gamma_T(v_i, h)$. By definition of our algorithm, all the maximal cliques $\kappa(u_i)$ have to be distinct. Thus, all paths $Path_T(u_i, v_i)$ are disjoint. Therefore, the number of indices i such that $Path_T(u_i, v_i)$ contains nodes in any $\Pi(\kappa, \ell)$ is no more than the cardinality of $\Pi(\kappa, \ell)$. It follows that our algorithm does not abort at Step 12. □

3.3 A New Characterization for Tree Powers

Theorem 2. *A graph G has a p-th tree root if and only if G is a uniquely representable chordal graph admitting a clique-position function with respect to p.*

Proof. Lemma 3 ensures the only-if direction. The rest of the proof shows the other direction. Let $\mathcal{T}$ be the clique tree of G. Let Φ be a clique-position function of G with respect to p. We construct a tree T with $V(T) = V(G)$ as follows.

- Let S consist of the nodes u of G with $\Phi(u) = (K, h)$ for some maximal clique K of G. By Condition C2 of Φ, we know that Φ provides a one-to-one mapping between S and $\mathcal{K}_G$. Let $T[S]$ be the tree isomorphic to $\mathcal{T}$ via this isomorphism.
- As for each node u of G not in S, we know that $\Phi(u) = (K, i)$ for some maximal clique K of G and some index i with $0 \leq i < h$. We simply add an edge between u and an arbitrary node w with $\Phi(w) = (K, i + 1)$. By Condition C3 of Φ, such a node w always exists.

One can see that the resulting T is a tree. It is also clear that the path of T attached to $T[S]$ has length no more than h.

We first prove $T(h) = T[S]$ by showing that each leaf of $T[S]$ is attached by a length-h path in T. By definition of T, it suffices to ensure that for each leaf K of $\mathcal{T}$, there exists a node v of G with $\Phi(v) = (K, 0)$: Let K' be the maximal clique of G with $(K, K') \in E(\mathcal{T})$. By the maximality of K, there exists a node u in $K \setminus K'$. By definition of clique tree, $\mathcal{K}_G(u)$ induces a subtree of $\mathcal{T}$. Since K is a leaf of $\mathcal{T}$, it follows that $\mathcal{K}_G(u) = \{K\}$, i.e., $(K, 0)$ is the unique clique position of u in G. By Condition C1 of Φ, we have $\Phi(u) = (K, 0)$.

Next we show that $\Phi(v) = (K, h)$ implies $\Gamma_T(v, h) = K$.

- To show $K \subseteq \Gamma_T(v, h)$, let u be a node in K, where $\Phi(u) = (K', i)$. Observe that $d_T(K, K') \leq i$. Let w be the node of S with $\Phi(w) = (K', h)$. Since $T[S]$ is isomorphic to $\mathcal{T}$, we know $d_T(v, w) \leq i$. By Condition C1 of Φ, $u \in K'$. By the definition of T, $d_T(u, w) = h - i$. It follows that $d_T(v, u) = d_T(v, w) + d_T(w, u) \leq h$. Thus, $u \in \Gamma_T(v, h)$.

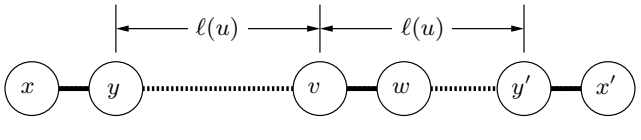

Fig. 2. An illustration for showing that u has only one clique position in T^p

– To show $\Gamma_T(v,h) \subseteq K$, let u be a node in $\Gamma_T(v,h)$, where $\Phi(u) = (K',i)$. Let w be the node with $\Phi(w) = (K',h)$. By the definition of T, $d_T(u,v) = d_T(u,w)+d_T(w,v) = h-i+d_T(w,v)$. Since $d_T(u,v) \leq h$, we have $d_T(w,v) \leq i$. Since $T[S]$ is isomorphic to $\mathcal{T}$, we have that $d_{\mathcal{T}}(K,K') \leq i$. By Condition C1 of Φ, $u \in \Pi(K',i)$. Hence $\mathcal{K}_G(u) = \Gamma_{\mathcal{T}}(K',i)$. Since $d_{\mathcal{T}}(K,K') \leq i$, we have $u \in K$.

Since $\Phi(v) = (K,h)$ implies $\Gamma_T(v,h) = K$, we have that $\mathcal{K}_G = \{\Gamma_T(v,h) \mid v \in S\}$ by Condition C2 of Φ. By $T[S] = T(h)$, we know that $\mathcal{K}_G = \{\Gamma_T(v,h) \mid v \in V(T(h))\}$. By (the "if" direction of) Lemma 2, $G = T^p$. □

3.4 A Linear-Time Algorithm

Theorem 3. *The p-th tree root problem for any n-node m-edge graph G and any even number p can be solved in $O(n+m)$ time.*

Proof. The constructive proof for the "if" direction of Theorem 2 can be implemented to run in $O(n)$ time. Since chordal graphs can be recognized in linear time [13, 15] and a clique tree $\mathcal{T}$ of the input chordal graph G can be obtained in linear time [6], the remaining task for solving the p-th tree root problem for G in $O(n+m)$ time is to compute a clique-position function, if any, of G with respect to p. Although Algorithm 1 is designed for computing a clique-position function for T^p, we can still run it on any chordal graph G. If the execution of the algorithm aborts at Step 2 or 12, Lemma 3 ensures that G does not admit any p-th tree root. If the algorithm does not abort at Step 12 and successfully outputs a function Φ, it takes $O(n)$ time to determine whether Φ is indeed a clique-position function of G, thereby figuring out whether G admits a p-th tree root.

It suffices to show that Step 1 of Algorithm 1, i.e., determining the clique positions of all nodes in G, can be implemented to run in $O(n+m)$ time. Observe that $\sum_{K \in \mathcal{K}_G} |V(K)| = O(n+m)$. All clique positions of all nodes in G can be computed in time linear in the size of G as follows. For each node $u \in V(G) \setminus Dom(G)$, let X_u consist of the nodes X of $\mathcal{K}_G(u)$ such that the degree of X in $\mathcal{T}[\mathcal{K}_G(u)]$ is less than the degree of X in $\mathcal{T}$. Observe that $u \in V(G) \setminus Dom(G)$ implies that X_u is non-empty. The sets $\mathcal{K}_G(u)$ and X_u for all nodes $u \in V(G) \setminus Dom(G)$ can be computed in $O(n+m)$ time. It remains to show how to solve the following problem.

> Let τ be a tree, some of whose leaves are marked. A node u of τ is a *center* of τ if u has the same distance δ to all marked leaves of τ and the distance between u and any node of τ is no more than δ. Under the assumption that τ has a center, the problem is to identify all centers of τ in $O(V(\tau))$ time.

Let y be an arbitrary marked leaf of τ. If $|V(Cent(\tau))| = 1$, then let x be the single node in $Cent(\tau)$. If $|V(Cent(\tau))| = 2$, then let x be the node of $Cent(\tau)$ whose distance to y in τ is larger. If τ has centers, then x has to be one of them. Let z be a node of $V(\tau) \setminus \{x\}$. Let Z be the connected component of $\tau - x$ that

contains z. One can verify that Z contains a marked leaf of τ if and only if z is not a center of τ. Therefore, the above problem can be solved in $O(|V(\tau)|)$ time. The theorem is proved. □

4 The p-th Tree Root Problem for Unknown Even p

Theorem 4. *Let G be an n-node m-edge graph. Then, it takes $O(n+m)$ time to determine the smallest even number p, if any, such that G admits a p-th tree root.*

Proof. Since tree powers are chordal and chordal graphs can be recognized in linear time, we may assume without loss of generality that G is chordal, thereby admitting a clique tree $\mathcal{T}$. Let K^* be a node in $Cent(\mathcal{T})$. Let J consist of the even numbers j such that $\Pi(K^*, j/2)$ is non-empty. Let

$$p^* = \begin{cases} \max J & \text{if } |Dom(G)| = 0; \\ d_{\mathcal{T}} + |Dom(G)| - 1 & \text{if } 1 \le |Dom(G)| \le 2; \\ 2\lceil d_{\mathcal{T}}/2 \rceil + 2 & \text{if } |Dom(G)| \ge 3. \end{cases}$$

Let P consist of the positive numbers p such that the input graph admits a p-tree root. We first show that if P is non-empty, then $p^* = \min P$. For brevity of proof, we regard $\mathcal{T}$ as being rooted at K^*: For any maximal clique K of G other than K^*, the *parent* of K in $\mathcal{T}$, denote $\pi(K)$, is the maximal clique K' of G such that (K, K') is an edge of $Path_{\mathcal{T}}(K, K^*)$. We say that K is a *child* of $\pi(K)$ in $\mathcal{T}$.

Case 1: $|Dom(G)| = 0$. By Condition C2 of any clique-position function of G with respect to p, we have $P \subseteq J$. We show $P = \{\max J\}$ by proving that that $j \le p$ holds for any even numbers $j \in J$ and $p \in P$. Assume for a contradiction that $j > p$ holds for some even numbers $j \in J$ and $p \in P$. Let T be a p-th tree root of G. Let $v(\cdot)$ be an isomorphism between $\mathcal{T}$ and $T(p/2)$. That is, $v(K)$ is the node in $V(T(p/2))$ such that $\Gamma_T(v(K), p/2) = K$.

Since $Dom(G) = \emptyset$, we know $d_T > 2p$. Therefore, $d_{\mathcal{T}} = d_{T(p/2)} > p$. It follows from $d_{\mathcal{T}} > p$ and $j > p$ that

$$\Gamma_{\mathcal{T}}(K^*, p/2) \subsetneq \Gamma_{\mathcal{T}}(K^*, j/2).$$

Let w be a node in $\Pi(K^*, j/2)$, implying $\mathcal{K}_G(w) = \Gamma_{\mathcal{T}}(K^*, j/2)$. Let $u^* = v(K^*)$. We know $u^* \in \Pi(K^*, p/2)$, implying $\mathcal{K}_G(u^*) = \Gamma_{\mathcal{T}}(K^*, p/2)$. It follows that

$$\mathcal{K}_G(u^*) \subsetneq \mathcal{K}_G(w),$$

thereby $u^* \neq w$. Since K^* is a centroid of $\mathcal{T}$, u^* is also a centroid of $T(p/2)$. By $d_{T(p/2)} > p$ and $u^* \neq w$, there is a node u in $V(T(p/2))$ such that $d_T(u, u^*) = p/2$ and $d_T(u, w) > p/2$. Let $K = \Gamma_T(u, h)$. We have $u^* \in V(K)$ and $w \notin V(K)$. Thus, K is a maximal clique in $\mathcal{K}_G(u^*) \setminus \mathcal{K}_G(w)$, contradicting $\mathcal{K}_G(u^*) \subsetneq \mathcal{K}_G(w)$.

Case 2: $1 \leq |Dom(G)| \leq 2$. We show that P consists of the single number $d_{\mathcal{T}} - |Dom(G)| + 1$ as follows. Let p be a number in P. Let T be a p-th tree root of G. By $1 \leq |Dom(G)| \leq 2$, it is not hard to see that $Dom(G)$ consists of the centroids of T. If $|Dom(G)| = 1$, then $d_T = 2p$. It follows that $d_{T(p/2)} = d_{\mathcal{T}} = p$. If $|Dom(G)| = 2$, then $d_T = 2p - 1$. It follows that $d_{T(p/2)} = d_{\mathcal{T}} = p - 1$. For either case, we have $p = d_{\mathcal{T}} + |Dom(G)| - 1$. Since G has exactly one clique tree $\mathcal{T}$, the diameter of $\mathcal{T}$ is fixed. We have $|P| = 1$.

Case 3: $|Dom(G)| \geq 3$. We show that $\min P = 2\lceil d_{\mathcal{T}}/2 \rceil + 2$. Let p be an index in P. Let T be a p-th tree root of G. By $|Dom(G)| \geq 3$, one can verify that $d_T \leq 2p - 2$. Therefore, $d_{\mathcal{T}} = d_{T(p/2)} \leq p - 2$, implying that $\min P \geq d_{\mathcal{T}} + 2$. Observe that $2\lceil d_{\mathcal{T}}/2 \rceil + 2$ is the smallest even number that is no less than $d_{\mathcal{T}} + 2$. We prove the equality by showing that if $p \geq d_{\mathcal{T}} + 4$, then G also admits a $(p-2)$-nd tree root.

The rest of the proof assumes $d_{\mathcal{T}} \leq p - 4$, which directly implies that

$$\Gamma_{\mathcal{T}}(K^*, p/2 - 2) = \mathcal{K}_G. \tag{1}$$

For any maximal clique K of G other than K^*, observe that $d_{\mathcal{T}} \leq p - 4$ also implies

$$\Pi(K, p/2) \subseteq \Pi(\pi(K), p/2 - 1); \tag{2}$$

$$\Pi(K, p/2 - 1) \subseteq \Pi(\pi(K), p/2 - 2). \tag{3}$$

Let Φ be a clique-position function of G with respect to p. Let $\Phi_1(u)$ (respectively, $\Phi_2(u)$) denote the first (respectively, second) component of $\Phi(u)$. Let Φ' be obtained from Φ by the following steps.

1. For each node u with $\Phi_2(u) \leq p/2 - 2$, we simply let $\Phi'(u) = \Phi(u)$.
2. For the node u with $\Phi(u) = (K^*, p/2)$, i.e., $u = v(K^*)$, let $\Phi'(u) = (K^*, p/2 - 1)$. For each node u of G with $\Phi(u) = (K^*, p/2 - 1)$, let $\Phi'(u) = (K^*, p/2 - 2)$.
3. For each maximal clique $K \neq K^*$ of G that is not a leaf of $\mathcal{T}$, we do the following:
   (a) Choose an arbitrary child K_1 of K in $\mathcal{T}$ and let $\Phi'(v(K_1)) = (K, p/2 - 1)$.
   (b) For each child K_2 of K other than K_1, let $\Phi'(v(K_2)) = (\pi(K), p/2 - 2)$.
   (c) For each node u of G with $\Phi(u) = (K, p/2 - 1)$, let $\Phi'(u) = (\pi(K), p/2 - 2)$.
4. For each maximal clique $K \neq K^*$ of G that is a leaf of $\mathcal{T}$, we do the following:
   (a) Choose an arbitrary node u with $\Phi(u) = (K, p/2 - 1)$ and let $\Phi'(u) = (K, p/2 - 1)$.
   (b) For each node $w \neq u$ with $\Phi(w) = (K, p/2 - 1)$, let $\Phi'(w) = (\pi(K), p/2 - 2)$.
5. Let u be the node with $\Phi(u) = (K^*, p/2)$. Let $\Phi'(u) = (K^*, p/2 - 1)$. For each node w with $\Phi(w) = (K, p/2)$ and $\pi(K) = K^*$, let $\Phi'(w) = (K^*, p/2 - 2)$.

We show that Φ' is a clique-position function of G with respect to $p - 2$.

- Condition C1: One can verify that $\Phi'(u)$ is well defined for each node u of G. Moreover, by Condition C1 of Φ and Equations (1), (2), and (3), one can also verify that $\Phi'(u)$ is a clique position of u in G.
- Condition C2: By Condition C2 of Φ and Steps 2, 3(a), 4(a), and 5, one can see that for each maximal clique K of G, there exists a unique node u of G with $\Phi'(u) = (K, p/2 - 1)$.
- Condition C3: Condition C2 of Φ' ensures that if u is a node with $\Phi'_2(u) = p/2 - 2$, then there is a node w with $\Phi'_1(w) = \Phi'_1(u)$ and $\Phi'_2(w) = \Phi'_2(u) + 1$. Condition C3 of Φ and Step 1 ensures that if u is a node with $\Phi'_2(u) < p/2-2$, then there is a node w with $\Phi'_1(w) = \Phi'_1(u)$ and $\Phi'_2(w) = \Phi'_2(u) + 1$.

Observe that whether P is empty or not, p^* can always be computed from G in linear time. (This includes the case that $J = Dom(G) = \emptyset$, i.e., p^* is undefined.) Thus, the above proof reduces the tree root problem G to the p^*-th tree root problem for G, which by Theorem 3 can be solved in linear time. □

References

1. P. Buneman. Characterization of rigid circuit graphs. *Discrete Mathematics*, 9:205–212, 1974.
2. G. A. Dirac. On rigid circuit graphs. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 25:71–76, 1961.
3. F. Gavril. The intersection graphs of subtrees in trees are exactly chordal graphs. *Journal of Combinatorial Theory, Series B*, 16:47–56, 1974.
4. S. K. Gupta and A. Singh. On tree roots of graphs. *International Journal of Computer Mathematics*, 73:157–166, 1999.
5. C.-W. Ho and R. C. T. Lee. Counting clique trees and computing perfect elimination schemes in parallel. *Information Processing Letters*, 31:61–68, 1989.
6. W.-L. Hsu and T.-H. Ma. Fast and simple algorithms for recognizing chordal comparability graphs and interval graphs. *SIAM Journal on Computing*, 28:1004–1020, 1999.
7. P. E. Kearney and D. G. Corneil. Tree powers. *Journal of Algorithms*, 29:111–131, 1998.
8. P. S. Kumar and C. E. V. Madhavan. Clique tree generalization and new subclasses of chordal graphs. *Discrete Applied Mathematics*, 117:109–131, 2002.
9. L. C. Lau. Bipartite roots of graphs. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 952–961, 2004.
10. L. C. Lau and D. G. Corneil. Recognizing powers of proper interval, split, and chordal graphs. *SIAM Journal on Computing*, 18(1):83–102, 2004.
11. Y. L. Lin and S. Skiena. Algorithms for square roots of graphs. *SIAM Journal on Discrete Mathematics*, 8:99–118, 1995.
12. R. Motwani and M. Sudan. Computing roots of graphs is hard. *Discrete Applied Mathematics*, 54:81–88, 1994.
13. D. Rose, R. Tarjan, and G. Lueker. Algorithmic aspects of vertex elimination of graph. *SIAM Journal on Computing*, 5(2):266–283, 1976.
14. I. C. Ross and F. Harary. The squares of a tree. *Bell System Technical Journal*, 39:641–647, 1960.
15. R. 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(3):566–576, 1984.