Recognizing $k$-Leaf Powers in Polynomial Time, for Constant $k$

MANUEL LAFOND, Université de Sherbrooke, Canada

A graph $G$ is a $k$-leaf power if there exists a tree $T$ whose leaf set is $V(G)$, and such that $uv \in E(G)$ if and only if the distance between $u$ and $v$ in $T$ is at most $k$ (and $u \neq v$). The graph classes of $k$-leaf powers have several applications in computational biology, but recognizing them has remained a challenging algorithmic problem for the past two decades. The best known result is that 6-leaf powers can be recognized in polynomial time. In this article, we present an algorithm that decides whether a graph $G$ is a $k$-leaf power in time $O(n f(k))$ for some function $f$ that depends only on $k$ (but has the growth rate of a power tower function).

Our techniques are based on the fact that either a $k$-leaf power has a corresponding tree of low maximum degree, in which case finding it is easy, or every corresponding tree has large maximum degree. In the latter case, large-degree vertices in the tree imply that $G$ has redundant substructures which can be pruned from the graph. In addition to solving a long-standing open problem, it is our hope that the structural results presented in this work can lead to further results on $k$-leaf powers and related classes.

CCS Concepts: • Theory of computation → Graph algorithms analysis; Design and analysis of algorithms;

Additional Key Words and Phrases: Algorithms, graph theory, leaf powers

ACM Reference format:
Manuel Lafond. 2023. Recognizing $k$-Leaf Powers in Polynomial Time, for Constant $k$. ACM Trans. Algor. 19, 4, Article 35 (September 2023), 35 pages. https://doi.org/10.1145/3614094

1 INTRODUCTION

In computational biology, it is commonplace to use dissimilarity information between species to reconstruct a phylogenetic tree, in which the leaves are the species and the internal nodes represent common ancestors. Pairwise distances between sequences can be used for this task, but they are known to be unreliable because of, for instance, sequencing errors or heterogeneous evolutionary rates [20, 28]. In 2002, Nishimura et al. [27] proposed that each pair of species should simply be considered as either close or far. There should then be a threshold $k$ such that in the phylogeny, close species are at distance at most $k$ and far species at distance more than $k$. The $k$-leaf power problem arises when we model species as the vertices of a graph $G$ in which edges represent closeness. More specifically, we say that a graph $G$ is a $k$-leaf power if there exists a tree $T$ such that the set of leaves of $T$ is $V(G)$, and such that $uv \in E(G)$ if and only if $dist_T(u,v) \leq k$, where
$\text{dist}_T(u, v)$ is the distance between $u$ and $v$ in $T$ and $u, v$ are distinct. The tree $T$ is called a $k$-leaf root of $G$. The $G$ graph is a leaf power if it is a $k$-leaf power for some $k$.

Since their introduction, these graph classes have attracted the attention of both algorithm designers and graph theoreticians. Two important problems have remained open for the past two decades. The first is to obtain a precise graph-theoretical characterization of $k$-leaf powers in terms of $k$. A fundamental question asks whether, for all $k$, $k$-leaf powers can be characterized as chordal graphs that forbid a finite set of induced subgraphs. This is known to be true for $k = 2, 3, 4$ but unknown for higher $k$ [4, 8].

The second open problem is whether one can decide in polynomial time whether a graph $G$ is a $k$-leaf power, where $k$ could be fixed or given. This has been a long-standing problem even in the case $k = 0$. Polynomial-time recognition is possible for $k \leq 6$ [14, 18], and the technical feats required to solve the case $k = 6$ show that extending these results is far from trivial. In this work, we tackle the latter question and show that polynomial-time recognition is indeed possible for any constant $k$.

1.1 Related Work

It is well known that all $k$-leaf powers are chordal, and they are also strongly chordal (e.g., see [7]). The 2-leaf powers are collections of disjoint cliques, and the 3-leaf powers are exactly the chordal graphs that are bull, dart, and gem-free [4, 30]. This can be used to recognize them in linear time. For $k = 4$, a characterization of twin-free 4-leaf powers in terms of chordality and a small set of forbidden induced subgraphs is established, again leading to a linear-time algorithm [8]. For $k \geq 5$, a characterization still escapes us, except for distance-hereditary 5-leaf powers [5]. It is known that all $k$-leaf powers are also $(k + 2)$-leaf powers, but that they are not all $(k + 1)$-leaf powers [10].

Chang and Ko [13] have developed a linear-time recognition algorithm for 5-leaf powers, using a reduction from a similar problem known as the 3-Steiner root, in which members of $V(G)$ can also be in internal nodes in the desired tree. Recently, Ducoffe [18] showed that 6-leaf powers were polynomial-time recognizable, this time using a reduction from the 4-Steiner root problem and an elaborate dynamic programming approach. The case $k = 6$ is the farthest that could be achieved so far. Let us mention that $k$-leaf powers have clique-width at most $k + \max(\lceil k/2 \rceil - 2, 0)$ [22], and that in the work of Dom et al. [16, 17], the problem of editing a graph to a $k$-leaf power is studied.

A recent result of Eppstein and Havvaei [19] states that recognizing $k$-leaf powers is Fixed Parameter Tractable (FPT) in $k + \deg(G)$, where $\deg(G)$ is the degeneracy of the graph. Note that since a $k$-leaf power $G$ is chordal, one can infer that $\deg(G) = \tw(G) = \omega(G) - 1$, where $\tw(G)$ and $\omega(G)$ are the treewidth and clique number, respectively (this holds because chordal graphs satisfy $\tw(G) = \omega(G) - 1 \leq \deg(G) \leq \tw(G)$). This shows that the problem is also FPT in $\tw(G) + k$. We also mention that Chen et al. [15] show that if we require each node of the $k$-leaf root to have degree between 3 and $d$ (except leaves), then finding such a $k$-leaf root, if any, is FPT in $k + d$.

Recognizing the larger class of leaf powers is also a challenging open problem. Subclasses of strongly chordal graphs have been shown to be leaf powers [6, 24, 26], but not all strongly chordal graphs are leaf powers [7, 23, 25]. Let us also mention that leaf powers were shown to have mim-width 1, a parameter related to the maximum induced matching between cuts determined by a tree [23]. Recently, Bergougnoux et al. [1] showed polynomial-time recognition algorithms for leaf powers whose leaf root is a caterpillar with weighted edges, and for leaf powers that admit a star Neumann model. Other variants of $k$-leaf powers have also been proposed [9, 12]. Perhaps the most studied generalization of leaf powers are Pairwise Compatibility Graphs (PCGs), which are graphs $G$ for which there exist a tree $T$ with leaves $V(G)$ and numbers $a, b$ such that $uv \in E(G)$ if and only if $a \leq \text{dist}_T(u, v) \leq b$. The problems of characterizing and recognizing PCGs are also open, and we invite the reader to consult other works [11, 12, 29] for a list of recent results on the topic.

ACM Transactions on Algorithms, Vol. 19, No. 4, Article 35. Publication date: September 2023.
1.2 Our Contributions

In this work, we show that for any constant \( k \geq 2 \), one can decide whether a graph \( G \) is a \( k \)-leaf power in time \( O(n^{\Omega(k)}) \). Here, the function \( f \) depends only on \( k \), and thus \( k \)-leaf powers can be recognized in polynomial time for any constant \( k \). We must reckon that \( f(k) \) grows faster than a power tower function with base \( k \) and height \( 3k \) (i.e., \( f(k) \in \Omega(k \uparrow\uparrow (3k)) \) using Knuth’s up arrow notation). We did not attempt to optimize \( f(k) \), and it is possible that the techniques presented here can be refined in the future to attain a more reasonable \( f(k) \) exponent, or even to obtain an FPT algorithm in parameter \( k \).

To the best of our knowledge, several tools developed for this result have not been applied before. The main idea is that if \( G \) is a \( k \)-leaf power, then either \( G \) admits a \( k \)-leaf root of low maximum degree, in which case it can be found “easily,” or all \( k \)-leaf roots have large maximum degree, in which case it contains redundant substructures. By this, we mean that \( G \) has a large number of vertex-disjoint subgraphs that are easy to solve individually and admit the “same kind” of \( k \)-leaf roots. We can then argue that we can simply remove one of those redundant subsets from \( G \), obtain an equivalent instance, and repeat the process. We do borrow ideas from other works \cite{15, 19}, since we handle the “easy” instances mentioned earlier using dynamic programming on a tree decomposition.

Although the preceding ideas are used for algorithmic purposes, they may shed light on the graph-theoretical characterization of \( k \)-leaf powers. Indeed, our collection of similar subgraphs satisfies a number of graph properties of interest (see the next subsection). Combined with the knowledge gained from prior work, our side results may thus help in understanding the structure of \( k \)-leaf powers. It is also plausible that our techniques can be applied to solve open problems on some classes of PCGs and their variants.

1.3 Overview of Our Algorithm

Our approach requires a bit of a setup in terms of definitions, so here we first provide the main intuitions.

Assume for the moment that \( G \) admits a \( k \)-leaf root \( T \) of maximum degree \( d := d(k) \), some quantity that depends only on \( k \). Then for any leaf \( v \) in \( T \), there are at most \( d^k \) other leaves of \( T \) at distance at most \( k \) from \( v \). This implies that in \( G \), \( v \) has at most \( d^k \) neighbors, and so the maximum degree of \( G \) is at most \( d^k \). This bounds the maximum clique number and, since \( G \) is chordal, also bounds the treewidth by \( d^k \). Eppstein and Havvaei \cite{19} have shown that in this setting, one can decide whether \( G \) is a \( k \)-leaf power in time \( O((kd^k)^{cd^k}n) \) for some constant \( c \).

The difficult cases therefore arise when every \( k \)-leaf root of \( G \) has maximum degree above \( d \). At a very high level, our approach for this case can be described in four essential steps (Figure 1):

1. Find a large collection \( \{C_1 \cup Y_1, \ldots, C_d \cup Y_d\} \) of disjoint subsets of \( V(G) \) that have a “similar” neighborhood structure in the rest of \( G \) and that are easy to solve. Each \( C_i \) is small and cuts \( Y_i \) from the rest of the graph, and each subgraph induced by \( C_i \cup Y_i \) has maximum degree at most \( d^k \). Moreover, there is a special vertex \( z \) that contains every \( C_i \) in its neighborhood, which is used to glue together the \( C_i \)'s.

2. Consider the set of all \( k \)-leaf roots of each \( G[C_i \cup Y_i \cup \{z\}] \) subgraph, and ensure that many of these sets of \( k \)-leaf roots are “similar” (here, \( G[X] \) is the subgraph induced by \( X \)). If \( d \) is large enough, this will be the case. Suppose that \( G[C_i \cup Y_i \cup \{z\}] \) and several other subgraphs have such “similar” \( k \)-leaf roots.

3. Check whether \( G \sim (C_1 \cup Y_1) \) admits a \( k \)-leaf root. If not, we are done, but if so, let \( T \) be such a \( k \)-leaf root.
(4) Look at how the similar $C_i \cup Y_i \cup \{z\}$ subsets are organized inside $T$, for $i > 1$. Since they have a similar neighborhood structure as $C_i \cup Y_i$ and admit the same type of $k$-leaf roots, we can find a $k$-leaf root $T_i$ of $G[C_i \cup Y_i \cup \{z\}]$ that we can embed into $T$, while mimicking the organization of the $C_i \cup Y_i$’s. The common $z$ vertex serves as an anchor point to start this embedding.

Of course, we need to be precise about what is meant by a “similar” neighborhood structure and a “similar” set of $k$-leaf roots. We develop two ingredients: the notion of a similar structure of $G$ and the notion of the signature of a tree.

**Similar Structures.** Let $T$ be a $k$-leaf root of $G$ of maximum degree above some large enough $d$, and suppose that $T$ is rooted. We look at the leaves below a deepest high-degree node of $T$, and want to understand how their structure is reflected in $G$. This is represented in Figure 1. More specifically, $T$ has some deepest node $v$ with at least $d + 1$ children and whose descendants all have at most $d$ children. Take a leaf $z$ of $T$ at minimum distance from $v$. Let $v_1, \ldots, v_d$ be $d$ arbitrary children of $v$ such that none of them has $z$ as a descendant (such a choice exists since $v$ has more than $d$ children). Let $T_1, \ldots, T_d$ denote the subtrees rooted at the $v_i$’s. Then the leaves at distance at most $k$ from $z$ in a $T_i$ subtree form a subset $C_i \subseteq V(G)$ of neighbors of $z$. Since $T_i$ has maximum degree $d$, one can argue that $|C_i| \leq d^k$. Moreover, if we assume that $G$ is connected, it can be argued that $C_i \neq \emptyset$. In fact in $G$, each $C_i$ separates the other leaves contained in $T_i$ from the rest of $G$. Let us call these other leaves $Y_i$. Note that $Y_i$ might be empty, so $C_i$ might not exactly be a separator, and even if it is, it might not be minimal. Nevertheless, the degree bound on $v_i$ and its descendants implies that in $G$, the induced subgraph $G[C_i \cup Y_i \cup \{z\}]$ has maximum degree $d^k$. In other words, a high-degree $k$-leaf root of $G$ implies the existence of a large number of $C_i \cup Y_i \cup \{z\}$ subsets of vertices that are each “easy” to solve because they induce a subgraph of low degree.

There is yet another property of the $C_i$’s that is useful. If we look at the leaves in some $C_i$ at distance $l$ from $z$ for some $l \in [k]$, and the leaves in some other $C_j$ also at distance exactly $l$ from $v$, it can be shown that all these leaves share the same distance to the leaves “outside” of $T_i$ and $T_j$. In terms of $G$, this means that the $C_i$ vertices can be “layered” so that vertices in the same layer
Recognizing $k$-Leaf Powers in Polynomial Time, for Constant $k$

Exploiting Similar Structures. Even if we assume that $G$ has a $k$-leaf root of high degree, we do not have access to that $k$-leaf root and we do not know how to find the $C_i \cup Y_1$ structure. However, it can be found in $G$ by brute force, and our ultimate goal is to show that $G$ is a $k$-leaf power if and only if $G - (C_1 \cup Y_1)$ is a $k$-leaf power (where here we assume that the $C_i \cup Y_1$’s are indexed so that $C_1 \cup Y_1$ is a correct choice (see the following)).

So, imagine that we have found $z$ and the $C_i \cup Y_1$’s as in Figure 1 on the right, along with a layering of the $C_i$’s, but that we do not have the tree $T$ on the left. In fact, the actual $k$-leaf root might not look like $T$ at all, and the $C_i \cup Y_i$ leaf sets might not be in distinct subtrees. This is because even though the existence of a large-degree $k$-leaf root implies that our desired $C_i \cup Y_i$ structure exists, having found such a structure does not imply that the $k$-leaf root of $G$ looks like the $k$-leaf root in the figure. Nevertheless, if $d$ is large enough, we show that many of the $G[C_i \cup Y_i \cup \{z\}]$ subgraphs admit a “similar” set of $k$-leaf roots. To make this notion clearer, suppose that we take every $k$-leaf root of a $G[C_i \cup Y_i \cup \{z\}]$ subgraph, look at their restriction to $C_i \cup \{z\}$, and replace each leaf of $C_i$ by its layer number (Figure 2(c)). By “similar” sets of $k$-leaf roots, we mean that these sets of restricted $k$-leaf roots are exactly the same for many of the $G[C_i \cup Y_i \cup \{z\}]$ subgraphs. These can be computed using dynamic programming on a tree decomposition of $G[C_1 \cup Y_1 \cup \{z\}]$.

This is an oversimplification, but let us go with it for a moment. Assume that we have found a large enough number $l < d$ of the $C_1 \cup Y_1 \cup \{z\}$ subsets that admit exactly the same set of restricted $k$-leaf roots. Without loss of generality, we may assume that these are $C_1 \cup Y_1 \cup \{z\}, \ldots, C_l \cup Y_l \cup \{z\}$ after renumbering appropriately. We say that these form a homogeneous set. Then we can first find a $k$-leaf root $T$ of $G - (C_1 \cup Y_1)$ (if none exists, then $G$ is not a $k$-leaf power). If $d$ and $l$ are large enough, we can use a pigeonhole argument to deduce that several of the $C_i \cup \{z\}$ subsets, $i \in [l]$, are organized in the same manner in $T$, in the sense that restricting $T$ to these $C_i \cup \{z\}$ subsets and replacing $C_i$ leaves by their layer yields the same tree. We can then find a $k$-leaf root $T_1$ of $G[C_1 \cup Y_1 \cup \{z\}]$ with the same structure as these, and embed $T_1$ with the same organization as the others in $T$. Because $C_1$ is layered in the same manner as these other $C_i$’s, mimicking their structure in $T$ ensures that the distance relationships with the other vertices are satisfied. The $z$ vertex is important, since it serve as a common point between all the $C_i$’s and indicates where to start embedding. At this point, it becomes difficult to describe how this embedding is performed.

ACM Transactions on Algorithms, Vol. 19, No. 4, Article 35. Publication date: September 2023.
Fig. 3. An illustration of how a $C_1 \cup Y_1$ subset can be pruned and embedded. On the left, each $C_i \cup Y_i \cup \{z\}$ subset admits a certain set of $k$-leaf roots, and the signature of each of them is listed. As it turns out, the admitted signatures of $C_1 \cup Y_1 \cup \{z\}$ are identical to those of $C_2 \cup Y_2 \cup \{z\}$. We can then remove $C_1 \cup Y_1$ and find a $k$-leaf root $T$ of $G - (C_1 \cup Y_1)$. Its restriction to $C_2 \cup \{z\}$ has the same signature as some $k$-leaf root $T_1$ of $C_1 \cup Y_1 \cup \{z\}$. We use that common structure of $T_1$ to insert it into $T$ (bottom, where fat edges now represent the edges of $T_1$ after the embedding, some of which are new and some of which were already in $T$).

more concretely, and the interested reader is redirected to the proof of Theorem 3.8 for more details. This last step is represented in Figure 3.

Tree Signatures. As we mentioned, it does not quite work to expect a large number $l$ of the $G[C_i \cup Y_i \cup \{z\}]$ subgraphs to admit exactly the same sets of $k$-leaf roots, even if we restrict them to $C_i \cup \{z\}$ and replace the leaves by their layer. First, each restricted subtree must also remember how far the $Y_i$ vertices are from the nodes of the restricted subtree, to ensure that we do not make the $Y_1$ nodes too close to other nodes during the embedding. This can be done by labeling the internal nodes of our restricted trees with the distance to the closest $Y_i$ leaf that it leads to, as in Figure 2(c). Note that a similar trick was done by Chen et al. [15], and that such a way of compress parts of a graph to labeled vertices has been used in kernelization in the context of protrusion decompositions (where the labeled graph is called a boundaried graph (see [3, 21])).

Second, we cannot guarantee that enough of the $G[C_i \cup Y_i \cup \{z\}]$ will admit exactly the same set of restricted $k$-leaf roots, since the number of possibilities is too large. The solution is to define a compact representation of $k$-leaf roots restricted to some $C_i \cup \{z\}$, which we call its signature. This representation prunes information from the tree that is not necessary for our embedding. Conceptually, to obtain the signature of a tree, we look at each node, and if we find a node that has three or more child subtrees that are identical, we remove one of these subtrees. This is fine for our embedding, since the identical subtree can be inserted along the others that are identical. We repeat until this is not possible. This is illustrated in Figure 2(d). From (a) a $G[C_i \cup Y_i \cup \{z\}]$ subgraph, we look at (b) each $k$-leaf root with leaves of the $C_i$’s replaced by their layer, then (c) restrict to $C_i \cup \{z\}$ and remember the distances to the removed leaves, and finally (d) prune redundant subtrees. This gives the compact representation of one $k$-leaf root, and we must obtain them all.

It turns out that the number of possible compact representations of degree-bounded $k$-leaf roots is bounded by a function of $k$. By making $d$ large enough (as a function of $k$), we can guarantee with a pigeonhole argument that a large number $l$ of $G[C_i \cup Y_i \cup \{z\}]$ admit the exact same set of signatures. Note that concretely, signatures are represented as vectors of integers that encode the same information, since it allows for simpler proofs (see Section 3).
We now proceed with the details.

2 PRELIMINARY NOTIONS

For an integer \( n \), we use the notation \([n] = \{1, 2, \ldots, n\} \). All graphs in this work are finite, simple, and undirected. For a graph \( G \) and \( v \in V(G) \), we denote by \( N_G(v) \) the set of neighbors of \( v \) in \( G \), and we write \( N_G[v] = N_G(v) \cup \{v\} \). For \( X \subseteq V(G) \), we denote by \( N_G(X) = \bigcup_{x \in X} (N_G(x) \setminus X) \) the neighbors of members of \( X \) that are outside of \( X \). In addition, we write \( N_G[X] = N_G(X) \cup X \).

We may drop the subscript \( G \) if it is clear from the context. For \( X \subseteq V(G) \), we denote by \( G[X] \) the subgraph of \( G \) induced by \( X \). We define a connected component as a maximal set of vertices \( X \) such that \( G[X] \) is connected.

Unless stated otherwise, all trees in this work are rooted. Hence, we will usually say tree instead of rooted tree. We denote the root of a tree \( T \) by \( r(T) \). For a node \( v \in V(T) \), we write \( ch_T(v) \) for the set of children of \( v \) in \( T \). The arity of \( T \) is \( \max_{v \in V(T)} |ch_T(v)| \). We say that \( v \) is a leaf if \( v \) has no children. We write \( L(T) \) to denote the set of leaves of \( T \). It is important to note that leaves are sometimes defined as nodes with a single neighbor in \( T \). This slightly differs from our definition, since if \( r(T) \) has a single child in \( T \), it is not treated as a leaf here. The only case in which the root is also a leaf is when \( T \) has a single vertex (which has no children). A node \( u \in V(T) \) is a descendant of a node \( v \in V(T) \) if \( v \) is on the path from \( r(T) \) to \( u \). In this case, \( v \) is an ancestor of \( u \). Note that \( v \) is a descendant and ancestor of itself. Given a tree \( T \) and some \( v \in V(T) \), we let \( T(v) \) denote the subtree rooted at \( v \) (i.e., the subgraph of \( T \) induced by \( v \) and all its descendants). The distance between two nodes \( u \) and \( v \) of \( T \) is denoted \( dist_T(u, v) \). We define \( height(T) = 1 + \max_{l \in L(T)} dist_T(r(T), l) \).

Two trees \( T_1 \) and \( T_2 \) are equal if \( r(T_1) = r(T_2) \) and \( (V(T_1), E(T_1)) = (V(T_2), E(T_2)) \), in which case we write \( T_1 = T_2 \). Two trees \( T_1 \) and \( T_2 \) are called leaf-isomorphic, denoted \( T_1 \isom T_2 \), if \( L(T_1) = L(T_2) \), and there exists a bijection \( \mu : V(T_1) \rightarrow V(T_2) \) such that \( \mu(u) = u \) for every \( u \in L(T_1) \), \( \mu(r(T_1)) = r(T_2) \), and such that \( uv \in E(T_1) \) if and only if \( \mu(u)\mu(v) \in E(T_2) \). We call \( \mu \) a leaf-isomorphism. Note that this is stronger than the usual notion of isomorphism, since we require \( T_1 \) and \( T_2 \) to be built with the same set of leaves. Also observe that since leaves must be matched, the \( \mu \) function is unique.\(^1\)

Let \( X \subseteq V(T) \). The restriction of \( T \) to \( X \), denoted \( T|X \), is the subgraph of \( T \) induced by \( X \) and every vertex of \( T \) that belongs to a path between two elements of \( X \). We define \( r(T|X) \) as the lowest common ancestor of \( X \) in \( T \) (which must be in \( T|X \)). We shall repeatedly use the following facts: for \( u, v \in X \), \( dist_T|X(u, v) = dist_T(u, v) \), and that \( V(T|X) \subseteq V(T) \); in addition, \( r(T|X) \) has at least two children if \( |X| \geq 2 \). Note that for \( v \in V(T|X) \), \( ch_T(v) \setminus ch_T|X(v) \) denotes the set of children of \( v \) that are “hidden” by the restriction.

Leaf Powers and Their Properties

The main definition of interest in this article is the following.

Definition 1. Let \( G \) be a graph, and let \( k \) be a positive integer. A \( k \)-leaf root of \( G \) is a tree \( T \) such that \( L(T) = V(G) \), and such that for all distinct \( u, v \in V(G) \), \( uv \in E(G) \) if and only if \( dist_T(u, v) \leq k \).

The graph \( G \) is a \( k \)-leaf power if there exists a \( k \)-leaf root of \( G \).

Note that according to our definitions, the tree \( T \) is implicitly rooted. This is not required in the usual definition of leaf powers, although this is inconsequential for \( k \geq 2 \). This is because if an unrooted \( k \)-leaf root \( T \) has an internal node or has a single node, we may use it as the root. If not, \( T \) has exactly two vertices \( u \) and \( v \), which are both of degree 1. This implies that \( uv \in E(G) \), but

\(^1\)To see this, observe that \( \mu \) is forced for the leaves of \( T_1 \). Then, \( \mu \) is forced for the parent of those leaves, and then \( \mu \) is forced for the grandparents, and so on.
since $k \geq 2$, we may add an internal node between $u$ and $v$ in $T$ and use it as the root. Also observe that the $k$-leaf power property is hereditary. In other words, if $G$ is a $k$-leaf power and $X \subseteq V(G)$, then $G[X]$ is a $k$-leaf power. This is because if $T$ is a $k$-leaf root of $G$, then $T|X$ is a $k$-leaf root of $G[X]$, since restrictions preserve distances.

A graph is chordal if it has no induced cycle of length 4 or more. The following is a well-known property of $k$-leaf powers, for any $k$ (see [7, 27]).

**Lemma 2.1.** Let $G$ be a $k$-leaf power. Then $G$ is chordal.

The treewidth of a graph $G$ is denoted $tw(G)$ (we defer the definition of treewidth to Section 4). We will be interested in $k$-leaf roots that have high arity. Eppstein and Havvaei [19, Section 7] have shown that if $G$ is a graph of bounded treewidth, then deciding if $G$ is a $k$-leaf power is FPT in $k + tw(G)$. As we show, this implies that low-arity $k$-leaf roots, if any, can be found using this result.

**Lemma 2.2.** Let $G$ be a graph with $n$ vertices. If there exists a $k$-leaf root of $G$ of arity at most $d$, then $G$ has maximum degree $d^k$. Moreover, one can decide in time $O(n(d^k)^c d^k)$ whether such a $k$-leaf root exists, where $c$ is a constant.

**Proof.** Assume that $T$ is a $k$-leaf root of $G$ of arity at most $d$. Let $v \in V(G)$. Then in $T$, the number of leaves at distance at most $k$ from $v$ is bounded by $d^k$ (since if we imagine rerooting $T$ at $v$, each node will still have at most $d$ children—the astute reader will see that we could optimize to $d^{k-1}$, but let us not bother). This implies that $v$ has at most $d^k$ neighbors in $G$. Since this holds for every vertex, $G$ has maximum degree $d^k$. As for the second part of the lemma, we know by the work of Eppstein and Havvaei [19] that deciding if $G$ is a $k$-leaf power can be done in time $O(n \cdot (tw(G)k)^c d^k)$ for some constant $c$. To use this, we can first check whether $G$ is chordal in linear time, and if not reject it. Since $G$ is chordal, it is well known that $tw(G) = \omega(G) − 1$, where $\omega(G)$ is the size of a maximum clique in $G$. In any graph, $\omega(G)$ is at most the maximum degree plus 1. In our case, this implies that $tw(G) = \omega(G) − 1 \leq d^k$. Using the algorithm of Eppstein and Havvaei [19], we can check whether $G$ is a $k$-leaf power in time $O(n((d^k)^c d^k))$. □

3 FINDING REDUNDANT STRUCTURES IN $k$-LEAF POWERS

Let us fix a positive integer $k$ and an arbitrary graph $G$ for the rest of the article. We assume that $G$ is connected, as otherwise, each connected component can be treated separately (if a $k$-leaf root is found for each component, we can join their roots under a new root at distance more than $k$ from them). As we mentioned, an important difficulty is when $G$ does admit $k$-leaf roots, but they all have large arity. We now define our similar structures precisely. We then introduce the notion of a signature for their $k$-leaf roots. After that, we argue that many subsets admit the same $k$-leaf root signatures, and that we can prune one. In Section 4, we show how the set of $k$-leaf root signatures can be found.

3.1 Similar Structures

A similar structure of a graph $G$ is a tuple $S = (C, \mathcal{Y}, z, \mathcal{L})$, where

- $C = \{C_1, \ldots, C_d\}$ is a collection of $d \geq 2$ pairwise disjoint, non-empty subsets of vertices of $G$;
- $\mathcal{Y} = \{Y_1, \ldots, Y_d\}$ is a collection of pairwise disjoint subsets of vertices of $G$, some of which are possibly empty, and $C_i \cap Y_j = \emptyset$ for any $i, j \in [d]$;
- $z \in V(G)$ and does not belong to any subset of $C$ or $\mathcal{Y}$; and
\( L = \{\ell_1, \ldots, \ell_d\} \) is a set of functions where for each \( i \in [d] \), we have \( \ell_i : C_i \cup \{z\} \rightarrow \{0, 1, \ldots, k\} \), and the functions in \( L \) are called layering functions.

Additionally, \( S \) must satisfy several conditions. Let us denote \( C^* = \bigcup_{j \in [d]} C_j \). Let \( X = \{X_1, \ldots, X_f\} \) be the connected components of \( G - C^* \). Furthermore, denote by \( X_\ell \) the element of \( X \) that contains \( z \). For each \( i \in [d] \), denote \( X^{(i)} = \{X_j \in X : N_G(X_j) \subseteq C_i\} \)—that is, the components that have neighbors only in \( C_i \).

Then all the following conditions must hold:

1. For each \( i \in [d] \), \( Y_i = \bigcup_{X_j \in X^{(i)}} X_j \) (\( Y_i = \emptyset \) is possible).
2. \( C^* \subseteq N_G(z) \).
3. For all \( X_j \in X \setminus \{X_z\}, X_j \subseteq Y_i \) for some \( i \in [d] \). In particular, \( X_z \) is the only connected component of \( G - C^* \) with neighbors in two or more \( C_i \)’s.
4. The layering functions \( L \) satisfy the following:
   - (a) \( z \) has special layer 0: For each \( i \in [d] \), \( \ell_i(z) = 0 \). Moreover, \( \ell_i(x) > 0 \) for any \( x \in C_i \).
   - (b) Same layer implies same neighbors: For any \( i, j \in [d] \) and any \( x \in C_i, y \in C_j \), it holds that \( \ell_i(x) = \ell_j(y) \) implies \( N_G(x) \setminus (C_i \cup Y_i \cup C_j \cup Y_j) = N_G(y) \setminus (C_i \cup Y_i \cup C_j \cup Y_j) \). Note that this includes the case \( i = j \).
   - (c) Close layers implies edge: For any \( i, j \in [d] \) and any \( x \in C_i, y \in C_j \), \( \ell_i(x) + \ell_j(y) \leq k \) implies \( xy \in E(G) \). Note that this includes the case \( i = j \).
   - (d) Far layers implies non-edge: For any two distinct \( i, j \in [d] \) and any \( x \in C_i, y \in C_j \), \( \ell_i(x) + \ell_j(y) > k \) implies \( xy \notin E(G) \). Note that this does not include the case \( i = j \).

We will refer to the value of \( d \) as the size of \( S \).

Although somewhat burdensome, the properties of a similar structure occur naturally when we look at the subtrees under a given node of a \( k \)-leaf root. The similar structure is the one that is described in Figure 1, but in a more precise manner. The properties of similar structures essentially say that after removing each \( C_i \in C \) from \( G \), there is one connected component \( X_z \) with a \( z \in X_z \) that is a neighbor of each \( C_i \). All the other connected components are separated from the rest of the graph by exactly one \( C_i \), and these form the \( Y_i \)’s. As for the layering functions, for the \( C_i \) vertices, the layers represent how the neighborhoods of the \( C_i \) members are organized. One can imagine that \( G \) has a \( k \)-leaf root and that the layer of \( x \in C_i \) is the distance from \( x \) to the lowest common ancestor of all the \( C_i \)’s (of course, this is conceptual since we do not have this \( k \)-leaf root). Any two vertices at the same layer must have the same neighborhood outside of their \( C_i \)’s. Vertices from “close” layers (i.e., with sum at most \( k \)) must be neighbors. If the layers are “far” (i.e., with sum more than \( k \)), then the vertices should not be neighbors (unless they are in the same \( C_i \)). One point to remember is that \( z \) is a special vertex with layer 0. This is set artificially to distinguish \( z \) from the vertices that are in some \( C_i \), and setting \( \ell(z) = 0 \) will be convenient in some of the proofs.

We first show that similar structures can always be found on graphs with \( k \)-leaf roots of high arity. This is essentially a formalization of Figure 1.

**Lemma 3.1.** Let \( d \geq 2 \) and let \( G \) be a connected graph that admits a \( k \)-leaf root of arity at least \( d + 1 \). Then there exists a similar structure \( S = (C, \mathcal{Y}, z, L) \) of \( G \) such that \( |C| = d \). Moreover, for each \( C_i \in C, |C_i| \leq d^k \) and \( G[C_i \cup Y_i \cup \{z\}] \) has maximum degree at most \( d^k \).

**Proof.** Let \( T \) be a \( k \)-leaf root of \( G \) of arity at least \( d + 1 \). We show how to construct \( C = \{C_1, \ldots, C_d\}, \mathcal{Y} = \{Y_1, \ldots, Y_d\}, z, \) and \( L = \{\ell_1, \ldots, \ell_d\} \) using the relationship between \( T \) and \( G \). Let \( v \) be a deepest node of \( T \) with \( d + 1 \) or more children. By the choice of \( v \), all the descendants of \( v \) have at most \( d \) children.

ACM Transactions on Algorithms, Vol. 19, No. 4, Article 35. Publication date: September 2023.
Let $z \in L(T)$ be a leaf of $T$ at minimum distance from $v$—that is, $z$ minimizes $\text{dist}_T(z, v)$ among all leaves of $T$. Note that $z$ may or may not be a descendant of $v$. Let $v_1, \ldots, v_d$ be $d$ children of $v$, none of which is an ancestor of $z$ (such a choice of $d$ children exists since $v$ has at least $d + 1$ children). For each $i \in [d]$, define $C_i = L(T(v_i)) \cap N_G(z)$. Note that since $z$ is not a descendant of $v_i$, the vertices of $L(T(v_i)) \cap N_G(z)$ must be at distance at most $k$ from $v$ (actually, $k - 1$, but we shall not bother). Since the $T(v_i)$ subtree has arity at most $d$, there are at most $d^k$ leaves in $L(T(v_i))$ at distance at most $k$ from $v$. It follows that $|C_i| \leq d^k$, for each $i \in [d]$. Note that by construction, the $C_i$ subsets are pairwise disjoint. To see that the $C_i$’s are non-empty, assume that $C_i$ is empty for some $i \in [d]$. Since $z$ is the closest leaf to $v$, and since every path from a member of $L(T(v_i))$ to a member of $V(G) \setminus L(T(v_i))$ passes through $v$, no element of $L(T(v_i))$ is at distance $k$ or less in $T$ from and element of $V(G) \setminus L(T(v_i))$. Since $L(T(v_i)) \neq \emptyset$, this implies that $G$ is disconnected, a contradiction. We may thus assume that $C_i \neq \emptyset$.

Now, for convenience, define $C^* = \bigcup_{i \in [d]} C_i$. Also define $G' = G - C^*$.

Let $X_z$ be the connected component of $G'$ that contains $z$. By construction, $z$ is a neighbor of every vertex in $C^*$. We must show that only $N_G(X_z)$ intersects with every $C_i$. Let $Z = L(T) \setminus (\bigcup_{i \in [d]} L(T(v_i)))$, and notice that $z \in Z$. We argue that $G'[Z]$ is connected. Assume otherwise. Then $G'[Z]$ has at least two connected components: one of them being $X_z$ and the other some other component $X_q$. Since $G$ is connected and removing $C^*$ separates $X_z$ from $X_q$, there is some $q \in X_q$ such that $N_G(q) \cap C^* \neq \emptyset$. Let $c \in N_G(q) \cap C^*$. Consider the position of $q$ in $T$. Because $q \in Z$, $q$ is not a descendant of any $v_i$, and thus the path from $q$ to $c$ in $T$ passes through $v$. But the choice of $z$ implies

$$k \geq \text{dist}_T(q, c) = \text{dist}_T(q, v) + \text{dist}_T(v, c) \geq \text{dist}_T(q, v) + \text{dist}_T(v, z) \geq \text{dist}_T(q, z).$$

Since $T$ is a $k$-leaf root of $G$, we have $qz \in E(G)$, contradicting that they belong to different connected components of $G'$. Thus, $G'[Z]$ is connected.

Now for $i \in [d]$, consider a leaf $x \in L(T(v_i)) \setminus C_i$. We argue that all neighbors of $x$ are in $L(T(v_i))$. Suppose that $x$ has a neighbor $q \in L(T) \setminus L(T(v_i))$. Then $\text{dist}_T(x, q) \leq k$, and the path from $x$ to $q$ goes through $v$. But again, by the choice of $z$,

$$k \geq \text{dist}_T(x, q) = \text{dist}_T(x, v) + \text{dist}_T(v, q) \geq \text{dist}_T(x, v) + \text{dist}_T(v, z) = \text{dist}_T(x, z).$$

But then, $x$ should be in $C_i$, which is a contradiction. It follows that $x$ has no neighbor outside of $L(T(v_i))$. In particular, $x \notin Z$.

It follows that $Z$ is a connected component of $G'$ and that it is equal to $X_z$. We may then define $Y_i = L(T(v_i)) \setminus C_i$ for each $i \in [d]$ (it is clear that $C_i \cap Y_j = \emptyset$ for all $i, j \in [d]$). The first three properties of similar structures can be verified from what we have gathered:

- Property 1 states that for each $i \in [d]$, $Y_i = \bigcup_{j \in X^{(i)}(v)} X_j$, where $X^{(i)}$ has the connected components of $G - C^*$ that have only neighbors in $C_i$. As we have argued, the leaves in $Y_i = L(T(v_i)) \setminus C_i$ have only neighbors in $C_i$ and the property holds.
- Property 2 states that $C^* \subseteq N_G(z)$, which holds by the construction of the $C_i$’s.
- Property 3 states that each connected component $X_j$ of $G - C^*$ belongs to some $Y_i$. This holds because $X_z = Z$, and thus any connected component $X_j$ other than $X_z$ must be a subset of some $L(v_i) \setminus C_i$. Such a connected component is included in $Y_i$ by construction.

It remains to describe our layering functions $\ell$. For $i \in [d]$, put $\ell_i(z) = 0$ and for $x \in C_i$, define

$$\ell_i(x) = \text{dist}_T(x, v).$$

Note that $\ell_i(x) > 0$ as required ($\ell_i(x) = 1$ is possible if $C_i$ only contains a leaf of $T$). Also note that $\ell_i(x) \leq k$ since each $x$ is a neighbor of $z$. Property 4a ($z$ has special layer 0) is thus satisfied. We show that each remaining property is satisfied.
Property $4b$ (same layer implies same neighbors). Let $i, j \in [d]$ and let $x \in C_i, y \in C_j$ such that $\ell_i(x) = \ell_j(y)$. Note that $q \in N_G(x) \setminus (C_i \cup Y_i \cup C_j \cup Y_j)$ only if $q$ is a leaf of $T$ at distance at most $k$ from $x$, and $q$ does not descend from $v_i$ or $v_j$. Therefore, the path from $x$ to $q$ goes through $v$. Since $\text{dist}_T(x, v) = \text{dist}_T(y, v)$ (because they have the same layer), we have $\text{dist}_T(x, q) = \text{dist}_T(y, q)$, and thus $q \in N_G(y) \setminus (C_i \cup Y_i \cup C_j \cup Y_j)$ as well. Thus, $N_G(x) \setminus (C_i \cup Y_i \cup C_j \cup Y_j)$, and the other containment direction can be argued symmetrically. This argument holds even when $i = j$. Thus, $S$ satisfies Property $4b$.

Property $4c$ (close layers implies edge). Let $i, j \in [d]$ and let $x \in C_i, y \in C_j$ with $\ell_i(x) + \ell_j(y) \leq k$. We see that $\text{dist}_T(x, y) \leq \text{dist}_T(x, v) + \text{dist}_T(y, v) = \ell_i(x) + \ell_j(y) \leq k$, implying that $xy \in E(G)$. This holds even when $i = j$. Thus, Property $4c$ is satisfied.

Property $4d$ (far layers implies non-edge). Let $i, j \in [d], i \neq j$, and let $x \in C_i, y \in C_j$ with $\ell_i(x) + \ell_j(y) > k$. Since $x$ and $y$ descend from distinct $v_i$ and $v_j$, $\text{dist}_T(x, y) = \text{dist}_T(x, v) + \text{dist}_T(v, y) = \ell_i(x) + \ell_j(y) > k$, and thus $xy \notin E(G)$. Thus, Property $4d$ is satisfied.

We have therefore shown that $S$ satisfies all requirements to be a similar structure, and that $|C_i| \leq d^k$ for each $i \in [d]$.

To finish the proof, recall that the lemma requires that each $G[C_i \cup Y_i \cup \{z\}]$ has maximum degree at most $d^k$. First note that $z$ has $|C_i| \leq d^k$ neighbors in this subgraph. Now let $x \in C_i \cup Y_i$. Any neighbor of $x$ in $G[C_i \cup Y_i \cup \{z\}]$ is at distance at most $k - 1$ from the parent $p$ of $x$ in $T$. Since $T_i$ has arity at most $d$, the number of other leaves of $C_i \cup Y_i$ at distance at most $k - 1$ from $p$ is at most $d^{k-1}$, and thus $x$ has at most $d^{k-1}$ neighbors in $C_i \cup Y_i$ (this also counts the leaves that do not descend from $p$). Note that $x$ could also have $z$ as a neighbor, so its degree in $G[C_i \cup Y_i \cup \{z\}]$ is at most $d^{k-1} + 1 \leq d^k$, as desired. □

3.2 Valued Trees and Signatures

We now know that similar structures exist in difficult $k$-leaf powers, but this is not enough. We would like to say that the $G[C_i \cup Y_i \cup \{z\}]$ subgraphs each admit “similar” sets of $k$-leaf roots, so that pruning one $C_i \cup Y_i$ does not matter. In fact, we are only interested in how the layer numbers of the $C_i \cup \{z\}$ subsets behave in these $k$-leaf roots, so we develop the notion of a valued restriction to remove the $Y_i$’s, while retaining the essential distance information to these $Y_i$’s. This is still not enough though, since the sets of such restricted $k$-leaf roots may still differ. We introduce the notion of a signature for these pruned trees, which is a compact representation that retains the essence of the structure of the trees.

A valued tree $T'$ is a pair $(T, \sigma)$, where $T$ is a tree and $\sigma : V(T) \setminus L(T) \to \mathbb{N} \cup \{\infty\}$ assigns each internal node of $T$ an integer, or possibly the special value $\infty$. We say that $T'$ is $s$-bounded if $\sigma(v) \leq s$ or $\sigma(v) = \infty$ for each $v \in V(T) \setminus L(T)$. We define $\text{height}(T') = \text{height}(T)$. For $v \in V(T'), T'(v) = (T', \sigma')$ denotes the valued tree in which $T' = T(v)$ and $\sigma'(w) = \sigma(w)$ for $w \in V(T'(v)) \setminus L(T')$. We say that two valued trees $(T_1, \sigma_1)$ and $(T_2, \sigma_2)$ are value-isomorphic if $T_1 \approx_{L_1} T_2$ and, denoting the leaf-isomorphism from $T_1$ to $T_2$ by $\mu$, it holds that $\sigma_1(w) = \sigma_2(\mu(w))$ for all $w \in V(T_1) \setminus L(T_1)$. The notion of valued restrictions will be fundamental for the rest of this article.

Definition 2. Let $T$ be a tree, and let $X \subseteq L(T)$. We say that $(T', \sigma)$ is the valued restriction of $T$ to $X$ if it satisfies the following:

- $T' = T[X]$.
- For each $v \in V(T') \setminus L(T')$, let $L'(v) = \bigcup_{x \in ch_T(v) \setminus ch_{T'}(v)} L(T(x))$ be the leaves below the children of $v$ that are hidden by the restriction. Then
  - if $L'(v) = \emptyset$, $\sigma(v) = \infty$;
  - otherwise, $\sigma(v) = \min_{l \in L'(v)} \text{dist}_{T'}(v, l)$.
Fig. 4. (a) A tree $T$. The set of leaves $v$ with a label $\ell(v)$ in $[0, 1, \ldots, 4]$ forms a set $X$, and the leaves in gray are those not in $X$. Note that these labels do not take part of the definition of a valued restriction, but they are needed for signatures. (b) The valued restriction $T'$ of $T$ to $X$, with the leaves of $X$ preserving their labels. (c) An illustration of the signature of each rooted subtree of the valued restriction. For each leaf $v$, the signature of $T(v)$ is $(\ell(v))$. We only give the full signature for one internal node. We chose to order its entries such that for $i \in 1, \ldots, 5$, the $i$-th coordinate is the number of children of that node that have signature $(i - 1)$, or 2 if this value is above 2. The last entry is the $\sigma$ value of the node (as for every other node). The parent of that node would then have one entry for each possible signature for a tree of height 2, and only one of those entries would be set to 1.

Figure 4(b) presents an illustration. Intuitively, the valued restriction of $T$ to $X$ takes $T' = T|X$ as a tree. By doing so, each remaining $v \in V(T')$ might have lost some children that were in $T$. We want $v$ to remember how far it is from a “hidden” leaf (i.e., descending from one of these lost children), and $\sigma(v)$ stores this information. As we will see later on, these removed leaves will correspond to $Y_i$ vertices. The tree $T'$ is a compressed version of $T$, and we will embed $T'$ into a $k$-leaf root instead of $T$. During this embedding, we want to know how far the hidden leaves are to ensure that outside leaves remain at distance more than $k$ from them.

**Valued Tree Signatures.** Let $T' = (T, \sigma)$ be a valued tree that is $s$-bounded for some $s$. Furthermore, let $\ell$ be a layering function that maps each leaf in $L(T)$ to an integer in $[0, 1, \ldots, k]$. We now define the signature of $T'$ with respect to $\ell$, denoted $\text{sig}_\ell(T')$, which encodes some properties of $T'$ in a vector of integers. The signature is defined recursively and depends on the height of $T'$, as illustrated in Figure 4(c).

If $\text{height}(T') = 1$, then $T$ has a single node $v$, which is a leaf. In this case, define $\text{sig}_\ell(T') = (\ell(v))$.

Now, assume that $\text{height}(T') = h > 1$. Let $S(s, h - 1) = \{s_1, \ldots, s_m\}$ denote the set of all possible signatures for an $s$-bounded valued tree of height $h - 1$ or less, with respect to any layering function that assigns leaves to $\{0, 1, \ldots, k\}$. We will later show that $m$ is finite. We may assume that the $s_i$ subscripts order the the signatures lexicographically (but this is merely for concreteness, the ordering does not matter). Then $\text{sig}_\ell(T')$ is a vector of dimension $m + 1$ in which, for $i \in [m]$, the $i$-th coordinate of $\text{sig}_\ell(T')$ is defined as

$$\text{sig}_\ell(T')[i] = \min(2, |\{v \in ch_T(r(T)) : \text{sig}_\ell(T'(v)) = s_i\}|).$$

Moreover, $\text{sig}_\ell(T')[m + 1] = \sigma(r(T))$.

It should be obvious that if $T'$ has height $h$, then $\text{sig}_\ell(T'(v)) \in S(s, h - 1)$ for each $v \in ch_T(r(T))$, and hence the signature summarizes every child subtree of $r(T)$. Two signatures are equal in the usual sense of vector equality—that is, if they have the same length and, at every position, the values of the two vectors are equal.

In other words, one may think of each integer $i \in [m]$ as a code for the $i$-th signature $s_i$, and $\text{sig}_\ell(T')[i]$ is the number of children of $r(T)$ whose subtree has signature $s_i$, except that we only...
both remembering whether there are 0, 1 or more of those (which we encode by 2). This encoding allows us to bound the number of possible signatures. We also reserve the last coordinate of $\text{sig}_r(T)$ for $\sigma(r(T))$. For valued trees with one node, we only need to remember the layer of the leaf.

We list the basic properties of signatures that will be needed, and then we will proceed with our $k$-leaf power algorithm.

**Lemma 3.2.** Let $T_1 = (T_1, \sigma_1)$ and $T_2 = (T_2, \sigma_2)$ be valued trees satisfying $\text{sig}_{\ell_1}(T_1) = \text{sig}_{\ell_2}(T_2)$ for some layering functions $\ell_1$ and $\ell_2$. Then the following holds:

1. If $r(T_1)$ has a child $u$ such that $\text{sig}_{\ell_1}(T_1(u)) \neq \text{sig}_{\ell_2}(T_2(u'))$ for every $v \in \text{ch}_T(r(T_1)) \setminus \{u\}$, then $r(T_2)$ has exactly one child $u'$ such that $\text{sig}_{\ell_1}(T_1(u)) = \text{sig}_{\ell_2}(T_2(u'))$.
2. If $r(T_1)$ has two distinct children $u$, $v$ such that $\text{sig}_{\ell_1}(T_1(u)) = \text{sig}_{\ell_2}(T_2(v))$, then $r(T_2)$ has at least two distinct children $u'$, $v'$ such that $\text{sig}_{\ell_1}(T_1(u)) = \text{sig}_{\ell_2}(T_2(u')) = \text{sig}_{\ell_2}(T_2(v'))$.
3. For any $x \in V(T_1)$, there exists $y \in V(T_2)$ such that $\text{dist}_{T_1}(x, r(T_1)) = \text{dist}_{T_2}(y, r(T_2))$ and $\text{sig}_{\ell_1}(T_1(x)) = \text{sig}_{\ell_2}(T_2(y))$. In particular, this includes the case $x \in L(T_1)$.

**Proof.** Property 1 is due to the fact that the entry corresponding to $\text{sig}_{\ell_1}(T_1(u))$ must be equal to 1 in both the $\text{sig}_{\ell_1}(T_1)$ and $\text{sig}_{\ell_2}(T_2)$ vectors. Property 2 is because the entry corresponding to $\text{sig}_{\ell_1}(T_1(u)) = \text{sig}_{\ell_2}(T_2(v))$ must be equal to 2 in both the $\text{sig}_{\ell_1}(T_1)$ and $\text{sig}_{\ell_2}(T_2)$ vectors.

Property 3 can be argued inductively on $\text{dist}_{T_1}(x, r(T_1))$. If the distance is 0, then $x = r(T_1)$ and $y = r(T_2)$ shows that the property holds. If $\text{dist}_{T_1}(x, r(T_1)) > 0$, let $x'$ be the child of $r(T_1)$ on the path between $x$ and $r(T_1)$. By the previous properties, $r(T_2)$ has a child $y'$ such that $\text{sig}_{\ell_1}(T_1(x')) = \text{sig}_{\ell_2}(T_2(y'))$. By induction, there is $y \in V(T_2(y'))$ such that $\text{dist}_{T_1(x')}(x, x') = \text{dist}_{T_2(y')}(y, y')$ and $\text{sig}_{\ell_1}(T_1(x')) = \text{sig}_{\ell_2}(T_2(y'))$. The distance from $x$ to $r(T_1)$ is thus the same as the distance from $y$ to $r(T_2)$, and therefore $y$ shows that the property holds.

Importantly, we can argue that the number of possible signatures depends only on height and $s$-boundedness.

**Lemma 3.3.** Let $S(s, h)$ denote the set of all possible signatures for $s$-bounded valued trees of height $h$ or less, for any layering function that map leaves to $\{0, 1, \ldots, k\}$. Then

$$|S(s, h)| \leq \begin{cases} k + 1 & \text{if } h = 1, \\ (s + 2) \cdot 3^{|S(s, h-1)|} + |S(s, h - 1)| & \text{otherwise.} \end{cases}$$

**Proof.** All valued trees with $h = 1$ have a signature of the form $(\ell(v))$, and $\ell(v)$ can take up to $k + 1$ values. For $h > 1$, consider an $s$-bounded valued tree of height $h$. For each of the $S(s, h - 1)$ possible signatures of valued trees of height $h - 1$ or less, the signature vector has an entry in $\{0, 1, 2\}$ (i.e., three possible values for each element of $S(s, h - 1)$). Moreover, the last entry is $\sigma(r(T))$, which can take up to $s + 2$ values in $\{0, 1, \ldots, s, \infty\}$. This counts the number of signatures for a valued tree of height exactly $h$. Since $S(s, h)$ includes signatures for valued trees of height $h$ or less, we must add the term $|S(s, h - 1)|$ to count the valued trees of height $h - 1$ or less.

We note that the upper bound on $|S(s, h)|$ can be computed in time proportional to $h$ (omitting the bit manipulations required to handle the memory required to represent the large value of $|S(s, h)|$), since for each $h \geq 2$, it suffices to store $|S(s, h - 1)|$ after computing it, and use it to compute $|S(s, h)|$ in constant time, and repeat.

We now come back to $k$-leaf powers, and show that in all the cases that will be of interest to us, $s$ and $h$ are bounded by a function of $k$. 

ACM Transactions on Algorithms, Vol. 19, No. 4, Article 35. Publication date: September 2023.
Lemma 3.4. Assume that $G$ is a connected $k$-leaf power, and let $X \subseteq V(G)$. Assume that there is $C \subseteq X$ such that $C$ is a clique and $X \setminus C \subseteq N_G(C)$. Let $T^*$ be a $k$-leaf root of $G$, and let $T = (T, \sigma)$ be the valued restriction of $T^*$ to $X$. Then $T$ is $k$-bounded and $\text{height}(T) \leq 3k$.

Proof. Let us first argue that $T$ has height at most $3k$. Consider the tree $T|C$. Then $T|C$ has height at most $k$, as otherwise the members of the $C$ clique cannot be at distance at most $k$ from each other. Moreover, any $x \in X \setminus C$ is at distance at most $k$ from some node of $T|C$, as otherwise $x$ cannot be at distance at most $k$ from any member of $C$. Thus, in $T$, $r(T)$ is at distance at most $k$ from $r(T|C)$, and any leaf in $X \setminus C$ is at distance at most $2k$ from $r(T|C)$, since the farthest possible leaf is at distance $k$ below the deepest node of $T|C$. Hence, $\text{height}(T) \leq 3k$.

As for $k$-boundedness, suppose that $T$ is not $k$-bounded. Then there is some $v \in V(T) \setminus L(T)$ such that $\sigma(v) > k$ but $\sigma(v) \neq \infty$. Let

$$L^*_v = \bigcup_{v' \in ch_T(v)} L(T' (v')).$$

Since $(T, \sigma)$ is the valued restriction of $T^*$ to $X$, each leaf $w \in L^*_v$ is in $V(G) \setminus X$ and is at distance greater than $k$ from $v$. Moreover, we know that at least one such $w$ exists, as otherwise we would have $\sigma(v) = \infty$. Then $w$ only has neighbors in $L^*_v$, since any other neighbor would require a path of length at most $k$ that passes through $v$, which is not possible. It follows that every member of $L^*_v$ only has neighbors in $L^*_v$. In other words, $L^*_v$ is disconnected from the rest of the graph, a contradiction since $G$ is connected. Therefore, $T$ is $k$-bounded.

Lemma 3.4 foreshadows the fact that only $k$-bounded values trees of height at most $3k$ will be of interest to us. Using the fact that $k + 2 \leq 2k$, we can plug in $s + 2 \leq 2k$ and $h = 3k$ in Lemma 3.4 and provide a rough upper bound on the number possible signatures for such values trees.

Lemma 3.5. The number of possible signatures for $k$-bounded valued trees of height at most $3k$ satisfies

$$|S(k, 3k)| \leq 4k \cdot 3^{4k} \cdot 3^{4k} \cdot 3^{4k} \cdot 3^{-4k} \cdot 3^{k+1}$$

where the height of the power tower is $3k$.

Proof. For $h > 1$, Lemma 3.4 can be simplified to $|S(s, h)| \leq (s + 2) \cdot 3^{S(s, h - 1)} + |S(s, h - 1)| \leq 2 \cdot (s + 2) \cdot 3^{S(s, h - 1)}$ (since obviously, $|S(s, h - 1)| \leq (s + 2)3^{S(s, h - 1)}$). We can plug in $s = k$ and $h = 3k$ and use the fact that $s + 2 \leq 2k$, which leads us to $|S(k, 3k)| \leq 2 \cdot 2k \cdot 3^{S(k, 3k - 1)}$, which yields the power tower expression. The last exponent is $k + 1$ because this corresponds to the $h = 1$ case.

Note that the $k$ and $3k$ bounds could be slightly improved with a more detailed analysis, but we would still obtain a power tower behavior for the number of possible signatures.

3.3 Pruning Subsets with Redundant $k$-Leaf Root Signatures

We now establish the connections between similar structures and signatures. Let $S = (C, \mathcal{Y}, z, \mathcal{L})$ be a similar structure of $G$. Unless mentioned otherwise, we will always assume that $C = \{C_1, \ldots, C_d\}$, $\mathcal{Y} = \{Y_1, \ldots, Y_d\}$ and $\mathcal{L} = \{\ell_1, \ldots, \ell_d\}$ for some $d$. We will mainly look at the $k$-leaf roots of the $G[C_i \cup Y_i \cup \{z\}]$ subgraphs. For $i \in [d]$, let $LR_k(C_i \cup Y_i \cup \{z\})$ be the set of all $k$-leaf roots of $G[C_i \cup Y_i \cup \{z\}]$ whose root is the unique neighbor of $z$ in the tree. Note that $LR_k(C_i \cup Y_i \cup \{z\})$ captures every possible $k$-leaf root, except that a specific node is chosen as the root.
Lemma 3.6. Assume that $G$ is a connected $k$-leaf power, and let $S = (C, Y, z, L)$ be a similar structure of $G$. For $i \in [d]$, let $T^* \in LR_z(C_i \cup Y_i \cup \{z\})$, and let $T = (T, \sigma)$ be the valued restriction of $T^*$ to $C_i \cup \{z\}$. Then $T$ is $k$-bounded and $\text{height}(T) \leq 3k$.

Proof. Note that $G[C_i \cup Y_i \cup \{z\}]$ is connected, since $Y_i$ consists of connected components that have neighbors in $C_i$, and all vertices in $C_i$ have $z$ as a neighbor. Thus, we can apply Lemma 3.4, since $\{z\}$ is a clique and $C_i \subseteq N_C(z)$.

Together, Lemmas 3.3 and 3.6 allow us to bound the number of possible signatures of $k$-leaf roots of $G[C_i \cup Y_i \cup \{z\}]$ subgraphs by $|S(k, 3k)|$, which grows quickly but is a function of $k$ only. The last piece we need is to prove the existence of a similar structure where all the $(C_i \cup Y_i \cup \{z\})'$s have $k$-leaf roots with the same signatures.

Let $s \in S(k, 3k)$ be a possible signature for a $k$-bounded valued tree of height at most $3k$, for any layering function. We say that $C_i \cup Y_i \cup \{z\}$ accepts $s$ if there exists $T^* \in LR_z(C_i \cup Y_i \cup \{z\})$ such that $\text{sig}_T(T) = s$, where $T$ is the valued restriction of $T^*$ to $C_i \cup \{z\}$ and $\ell_i$ is the $i$-th layering function of $L$.

We then define

$$\text{accept}(S, C_i) = \{s \in S(k, 3k) : C_i \cup Y_i \cup \{z\} \text{ accepts } s\}.$$ 

It is important to note that by Lemma 3.6, for any $T^* \in LR_z(C_i \cup Y_i \cup \{z\})$, $\text{sig}_T(T) \in S(k, 3k)$, and therefore $\text{sig}_T(T) \in \text{accept}(S, C_i)$, where $T$ is the valued restriction of $T^*$ to $C_i \cup \{z\}$. In other words, $\text{accept}(S, C_i)$ captures the signature of every $k$-leaf root of $G[C_i \cup Y_i \cup \{z\}]$.

We need one last definition.

Definition 3. A similar structure $S = (C, Y, z, L)$ of $G$ is called homogeneous if, for each $C_i, C_j \in C$, $\text{accept}(S, C_i) = \text{accept}(S, C_j) \neq \emptyset$.

For our purposes, we need a homogeneous similar structure of size at least $3|S(k, 3k)|$. If all $k$-leaf roots have large enough arity, this is guaranteed to exist.

Lemma 3.7. Let $G$ be a connected $k$-leaf power. Let $d = |S(k, 3k)| \cdot 2^{|S(k, 3k)|}$, and assume that $G$ admits a $k$-leaf root of arity at least $d + 1$. Then there is a homogeneous similar structure $S = (C, Y, z, L)$ of $G$ such that $|C| = 3|S(k, 3k)|$. Moreover, for each $C_i \in C$, $|C_i| \leq d^k$ and $G[C_i \cup Y_i \cup \{z\}]$ has maximum degree at most $d^k$.

Proof. By Lemma 3.1, there exists a similar structure $S' = (C', Y', z, L')$ of $G$ with $|C'| = d$, with each $C'_i \in C'$ having $|C'_i| \leq d^k$ and $G[C'_i \cup Y'_i \cup \{z\}]$ having maximum degree $d^k$ or less. Denote $C' = \{C'_1, \ldots, C'_d\}$, $Y' = \{Y'_1, \ldots, Y'_d\}$, and $L' = \{\ell'_1, \ldots, \ell'_d\}$. The problem is that $S'$ might not be homogeneous.

Notice that by Lemma 3.6, for any $C'_i \in C$, $\text{accept}(S', C'_i)$ is a subset of $S(k, 3k)$. Thus, the number of possible distinct $\text{accept}(S, C'_i)$ sets is $2^{|S(k, 3k)|}$. By the pigeonhole principle, the fact that $|C'| = d = |S(k, 3k)|2^{|S(k, 3k)|}$ implies that there is some $C \subseteq C'$ such that $|C| = 3|S(k, 3k)|$ and such that $\text{accept}(S', C'_i) = \text{accept}(S', C'_j)$ for every $C'_i, C'_j \in C$. Note that since $G$ is a $k$-leaf power, none of these accept sets is empty, since each $G[C_i \cup Y_i \cup \{z\}]$ is a $k$-leaf power.

We thus know that there exists a set of indices $h_1, \ldots, h_3|S(k, 3k)|$ such that $C = \{C'_{h_1}, C'_{h_2}, \ldots, C'_{h_3|S(k, 3k)|}\}$. Let $Y = \{Y'_{h_1}, Y'_{h_2}, \ldots, Y'_{h_3|S(k, 3k)|}\}$, and similarly let $L = \{\ell'_{h_1}, \ell'_{h_2}, \ldots, \ell'_{h_3|S(k, 3k)|}\}$. One can easily verify that all properties of a similar structure hold for $S = (C, Y, z, L)$ (in particular, all the $C'_j \cup Y'_j$ that were not kept in $C$ now join the same connected component as $z$). We have $|C| = 3|S(k, 3k)|$. This similar structure is homogeneous. Finally, the bounds for $|C_{h_i}| \leq d^k$ and the maximum degree at most $d^k$ for $G[C_{h_i} \cup Y_{h_i} \cup \{z\}]$ still hold for every $h_i$. 

ACM Transactions on Algorithms, Vol. 19, No. 4, Article 35. Publication date: September 2023.
We can finally present the main result of this section.

**Theorem 3.8.** Let $G$ be a connected graph. Let $S = (C, S, z, L)$ be a homogeneous similar structure of $G$, with $C = \{C_1, \ldots, C_l\}$, $Y = \{Y_1, \ldots, Y_l\}$ and $l = 3|S(k, 3k)|$. Then $G$ is a $k$-leaf power if and only if $G - (C_1 \cup Y_1)$ is a $k$-leaf power.

**Proof.** First, note that if $G - (C_1 \cup Y_1)$ is not a $k$-leaf power, then by heredity, $G$ is not a $k$-leaf power. We now focus on the other direction of the statement.

Suppose that $G - (C_1 \cup Y_1)$ is a $k$-leaf power, and let $R$ be a $k$-leaf root of $G - (C_1 \cup Y_1)$. Assume without loss of generality that $R$ is rooted at the single neighbor of $z$. The proof is constructive: we show algorithmically how to insert $C_1 \cup Y_1$ into $R$ to obtain a $k$-leaf root of $G$.

For $i \in \{2, \ldots, l\}$, let $T_i^* = R|(C_1 \cup Y_1 \cup \{z\})$, which is a $k$-leaf root of $G[C_1 \cup Y_1 \cup \{z\}]$. Furthermore, let $T_i = (T_i, \sigma_i)$ be the valued restriction of $T_i^*$ to $C_1 \cup \{z\}$. Note that $V(T_i) \subseteq V(T_i^*) \subseteq V(R)$.

Since $l - 1 > |S(k, 3k)|$, there must be distinct $i, j \in \{2, \ldots, l\}$ such that $\text{sig}_{C_1}(T_i) = \text{sig}_{C_1}(T_j)$, by the pigeonhole principle. Assume, without loss of generality, that $i = 2$ and $j = 3$ (otherwise, simply rename the $C_i$’s). Note that since $V(T_2) \subseteq V(T_2^*) \subseteq V(R)$, each node of $T_2$ and $T_2^*$ is in $R$. The same holds for $T_3$ and $T_3^*$. Moreover, all of $T_2, T_2^*, T_3,$ and $T_3^*$ contain $z$ and hence also contain the root of $R$. It follows that $r(T_2) = r(T_2^*)$, $r(T_2) = r(T_3^*)$, and $r(T_3^*) = r(R)$.

Since $\text{accept}(S, C_2 \cup Y_1 \cup \{z\}) = \text{accept}(S, C_2 \cup Y_2 \cup \{z\})$ by homogeneity, there exists a $k$-leaf root $T_1^*$ of $G[C_1 \cup Y_1 \cup \{z\}]$, with $r(T_1^*)$ being the parent of $z$, such that $\text{sig}_{C_1}(T_1^*) = \text{sig}_{C_2}(T_2) = \text{sig}_{C_3}(T_3)$, where $T_2 = (T_1, \sigma_1)$ is the valued restriction of $T_2^*$ to $C_1 \cup \{z\}$. Note that $r(T_1) = r(T_1^*)$, again because of $z$. We now describe how to insert $T_1^*$ into $R$, based on the signature of $T_1^*$. This is shown in Algorithm 1. By inserting a subtree $T_1^*(u)$ as a child of $r$, we mean to add all the nodes and edges of the $T_1^*(u)$ tree to $R$, and to add an edge between $r$ and $r(T_1^*(u))$.

Let us begin with a bit of intuition on this algorithm, which is illustrated in Figure 5. The idea is that $\text{insert}(t, r)$ embeds $T_1^*(t)$ into $R(r)$, where $t$ should be a node of $T_1$ and $r$ a node of $R$. The initial call says that $r(T_1^*) = r(T_1)$ should correspond to $r(R)$ after $T_1^*$ is inserted, and the recursive calls make similar correspondences with children of $r(T_1^*)$ and $r(R)$, recursively. We will maintain the invariant that at any point, $r \in V(R) \cap V(T_2) \cap V(T_3)$, such that $T_2(r), T_3(r)$ have the same signature as $T_1^*(t)$ (we prove this in the following). Then in any recursion, when $t$ has children $u$ in $T_1^*$ but not in $T_1$, we know that $T_1^*(u)$ only has leaves in $Y_1$ (line 7). In this case, we just insert the $T_1^*(u)$ subtree. The $Y_1$ leaves to insert must be at distance more than $k$ to all of $L(R)$, and the fact that $T_2(r)$ and $T_3(r)$ have similar subtrees at the $r$ location helps us guarantee it (see Figure 5(1)). In other words, $T_2(r)$ lets us argue on the distance relationships between $Y_1$ and the leaves in $L(R) \setminus L(T_2(r))$, and $T_3(r)$ helps us with all relationships with leaves in $L(T_2(r))$. This idea of complementarity is the reason we need both $T_2$ and $T_3$. A similar idea applies when $T_1^*(u)$ is inserted on lines 11 and 16. A recursion is needed for each $u \in ch_{T_1}(t)$ such that the node of $T_2$ and $T_3$ corresponding to $u$ is the same in $ch_R(t)$, since the complementarity trick cannot be applied. Let us proceed with the details.

Since the algorithm only inserts subtrees as children of nodes of $R$, it is not hard to see that the modified $R$ will also be a tree. For the rest of the proof, let $R'$ be the tree obtained after the algorithm has terminated, assuming that the initial call is $\text{insert}(r(T_1^*), r(R))$. The rest of the proof is dedicated to showing that $R'$ is a $k$-leaf root of $G$. The proof is divided in a series of claims.

**Claim 1.** At any point that $\text{insert}(t, r)$ is called and arguments $t$ and $r$ are received, the following holds:

- $t \in V(T_1) \setminus L(T_1)$,
- $r \in V(T_2) \cap V(T_3)$,
- $\text{sig}_{C_1}(T_1(t)) = \text{sig}_{C_2}(T_2(r)) = \text{sig}_{C_3}(T_3(r))$. 

ACM Transactions on Algorithms, Vol. 19, No. 4, Article 35. Publication date: September 2023.
Recognizing $k$-Leaf Powers in Polynomial Time, for Constant $k$

Fig. 5. Embedding $T_1^*$ ($t$) into $R(r)$. Here, $s$ represents subtrees whose signature is $s$. (1) Child subtrees of $t$ not kept by the restriction are inserted as a new child of $r$. (2) Child subtrees of $t$ with repeated signature are inserted. (3) and (4) Child subtrees with a unique signature are inserted if $r$ has two distinct children of $T_2^*$ and $T_3^*$ with that signature, and otherwise we recurse into the nodes with that signature.

**Algorithm 1: Insertion of $T_1^*$ into $R$.**

```
1 insert(r($T_1^*$), r(R)) //initial call
2
3 Function insert($t$, $r$)
4    // $t \in V(T_1^*)$ is the node of $T_1^*$ we are inserting
5    // $r \in V(R)$ is the node of $R$ we are inserting on
6    foreach child $u \in ch_{T_1^*}(t) \setminus ch_{T_1}(t)$ do
7         Insert the $T_1^*$ subtree as a child of $r$
8    end
9
10    foreach child $u \in ch_{T_1}(t)$ do
11        if $\exists w \in ch_{T_1}(t) \setminus \{u\}$ such that $\text{sig}_{T_1}(T_1(t)) = \text{sig}_{T_1}(T_1(u))$ then
12            Insert the $T_1^*$ subtree as a child of $r$
13        else
14            Let $u_2 \in ch_{T_2}(r)$ such that $\text{sig}_{T_2}(T_2(u_2)) = \text{sig}_{T_1}(T_1(u))$
15            Let $u_3 \in ch_{T_3}(r)$ such that $\text{sig}_{T_3}(T_3(u_3)) = \text{sig}_{T_1}(T_1(u))$
16            if $u_2 \neq u_3$ then
17                Insert the subtree $T_1^*(u)$ as a child of $r$
18            else
19                if $u_2 \neq z$ then
20                    Recursively call $insert(u, u_2)$
21                end
22        end
23    end
24
25 Proof. We argue this by induction on the depth of the recursion. As a base case, consider the initial call with $t = r(T_1^*)$ and $r = r(R)$. We have $t \in V(T_1)$ since $r(T_1^*) = r(T_1)$. To see that $t \notin L(T_1)$, note that $|L(T_1)| = |C_1 \cup \{z\}| \geq 2$ implies that $r(T_1)$ is not a leaf itself, since leaves have 0 children. Thus, initially, $t \in V(T_1) \setminus L(T_1)$. In addition, we have already argued that $r = r(R) = r(T_2) = r(T_3)$. Moreover, in this initial case, $\text{sig}_{T_1}(T_1(t)) = \text{sig}_{T_1}(T_1) = \text{sig}_{T_2}(T_2) = \text{sig}_{T_3}(T_2(r))$ and the same holds for $T_3$.

Now, assume that the statement holds for any recursive call of depth $\delta$. We argue that it also holds for recursive calls at depth $\delta + 1$. When a call at recursion depth $\delta$ with parameters $t$ and
$r$ is made, the only way a deeper recursive call could be made is when line 19 is reached. This happens when $t$ has a child $u$ in $T_1$ such that no other child of $t$ in $T_1$ has the same signature (see Figure 5(4)). Since we assume by induction that $\text{sig}_{\ell_1}(T_1(t)) = \text{sig}_{\ell_1}(T_2(r))$, by Lemma 3.2.1, $r$ must also have exactly one child $u_2$ in $T_2$ such that $\text{sig}_{\ell_1}(T_2(u_2)) = \text{sig}_{\ell_1}(T_1(u))$. By the same argument, $r$ has exactly one child $u_3$ in $T_3$ such that $\text{sig}_{\ell_1}(T_3(u_3)) = \text{sig}_{\ell_1}(T_1(u))$. In particular, $u_2$ and $u_3$ as used in the algorithm always exist.

A recursive call is only made if $u_2 = u_3$ and $u_2 \neq z$, in which case $u$ and $u_2$ are passed to a recursive call. We know that $u \in V(T_1)$, and must argue that $u \notin L(T_1)$. Assume instead that $u$ is a leaf of $T_1$. Since $\text{sig}_{\ell_1}(T_1(u)) = \text{sig}_{\ell_1}(T_2(u_2)) = \text{sig}_{\ell_1}(T_3(u_3))$, $u_2$ would also be a leaf of $T_2$ and $u_3$ a leaf of $T_3$ (because only leaves have a 1-dimensional signature vector). We have $L(T_2) = C_2 \cup \{z\}$ and $L(T_3) = C_3 \cup \{z\}$. Since $C_2$ and $C_3$ are disjoint, only $u_2 = u_3 = z$ is possible, but this is a contradiction since the algorithm explicitly states that line 19 could not be reached in this case. Hence, $u \notin L(T_1)$, which proves the first property of the claim.

The second and third properties of the claim hold for the recursive call because $u_2 = u_3$, and because they are explicitly chosen to have the same signature.

We show that to create $R'$, the algorithm inserts a subtree that is leaf-isomorphic to $T_1^\ast$. This ensures that distance relationships between leaves of $T_1^\ast$ are the same in $R'$.

Claim 2. $R'|(C_1 \cup Y_1 \cup \{z\}) \simeq_L T_1^\ast$.

Proof. We prove the claim by induction on the height of $T_1(t)$. In other words, we show that for any $t$ and $r$ passed to the algorithm, $R'$ contains a subtree $R'_t$ that is leaf-isomorphic to $T_1^\ast(t)$, and that the leaf-isomorphism from $T_1^\ast(t)$ to $R'_t$ maps $t$ to $r$.

By Claim 1, $t$ is in $T_1$ but not a leaf of $T_1$. So as a base case, consider that the algorithm receives arguments $t$ and $r$ such that $T_1(t)$ is a subtree of $T_1$ of height 2. In other words, all the children of $t$ in $T_1$ are leaves (but not necessarily all the children of $t$ in $T_1^\ast$). Consider a child $u \in ch_{T_1}(t) \setminus ch_{T_1}(t)$. Then line 7 ensures that $T_1^\ast(u)$ is inserted as a child subtree of $r$ in $R'$. Consider now a child $u \in ch_{T_1}(t)$, which is a leaf by assumption. If some other child $w$ of $T_1$ has the same signature (i.e., $\text{sig}_{\ell_1}(T_1(u)) = \text{sig}_{\ell_1}(T_1(w))$), line 11 ensures that $u$ is inserted as a child of $r$. Otherwise, there are $u_2 \in ch_{T_2}(r)$ and $u_3 \in ch_{T_3}(r)$ with the same signature as $u$ (i.e., $\text{sig}_{\ell_1}(T_2(u_2)) = \text{sig}_{\ell_1}(T_3(u_3)) = \text{sig}_{\ell_1}(T_1(u))$). As in the previous claim, $u_2$ and $u_3$ must also be leaves and one of $u_2 \neq u_3$ or $u_2 = u_3 = z$ must hold, because $C_2$ and $C_3$ are disjoint. If $u_2 = u_3 = z$, then $u = z$ as well, since this is the only leaf with signature (0). In this case, $t = r(T_1)$ and $r = r(R)$, and the check that $u_2 \neq z$ in the algorithm ensures that nothing happens (which is correct, since $z$ is already a child of $r$). If $u_2 \neq u_3$, $u$ is added as a child of $r$ on line 16 (which is correct since $u \neq z$). It follows that all children of $t$ in $T_1^\ast$ are children of $r$ in $R'$. Therefore, $R'$ contains a subtree leaf-isomorphic to $T_1^\ast(t)$ with $t$ mapped to $r$.

Now assume that $T_1(t)$ is a subtree of height greater than 2. Notice that for each child $u \in ch_{T_1}(t)$, $T_1^\ast(u)$ is inserted as a child subtree of $r$ in $R'$, unless $t$ has no other child with the same signature as $u$, and there are $u_2 \in ch_{T_2}(r)$ and $u_3 \in ch_{T_3}(r)$, with $u_2 = u_3$, with the same signature as $u$. As before, if $u = z$, then $u_2 = u_3 = z$ and nothing happens, which is correct. Otherwise, line 19 is reached and the algorithm is called recursively with arguments $u$ and $u_2$. In this case, we may assume by induction that $R'$ contains a subtree leaf-isomorphic to $T_1^\ast(u)$, with the leaf-isomorphism mapping $u$ to $u_2$. Let $U$ be the set of children of $t$ in $T_1$ for which line 19 is reached. Notice that for each distinct $u, u' \in U$, $\text{sig}_{\ell_1}(T_1(u)) \neq \text{sig}_{\ell_1}(T_1(u'))$, since children with a non-unique signature are inserted on line 11. Moreover, each $u \in U$ is put in correspondence with a child $u_2$ of $r$ with the same signature as $u$, implying that every $u \in U$ has a distinct correspondent among the children of $r$. We can therefore conclude that for each $u \in ch_{T_1}(t)$ (other than $z$), either $T_1^\ast(u)$ is inserted as
a child subtree of \( r \), or \( T^*_r(u) \) is inserted recursively with \( u \) being mapped to a unique child \( u_2 \) or \( r \). It follows that \( R' \) contains a subtree isomorphic to \( T^*_r(t) \) with \( t \) mapped to \( r \).

To finish the proof, we observe that the preceding statement holds for \( t = r(T_1) = r(T^*_1) \) and \( r = r(R) \). Thus, \( R' \) contains a subtree leaf-isomorphic to \( T^*_1 \) with a leaf-isomorphism that maps \( r(T^*_1) \) to \( r(R) \).

It remains to argue that distance relationships in \( R' \) are also correct for pairs of leaves with one in \( C_1 \cup Y_1 \cup \{ z \} \) and the other in \( V(G) \setminus (C_1 \cup Y_1 \cup \{ z \}) = L(R) \setminus \{ z \} \).

**Claim 3.** Let \( x \in L(T^*_1) \) and \( y \in L(R) \setminus \{ z \} \). If \( xy \in E(G) \), then \( \text{dist}_{R'}(x, y) \leq k \), and otherwise, \( \text{dist}_{R'}(x, y) > k \).

**Proof.** We argue that any time a leaf of \( T^*_1 \) is inserted, the conditions of the claim are satisfied. Consider any recursion of the algorithm where arguments \( t \) and \( r \) are given, and let us focus on the set of leaves of \( T^*_r \) inserted during that recursion in particular. To prove the claim, it is sufficient to prove that the distances between these leaves and \( L(R) \setminus \{ z \} \) are correct.

Let \( u \in ch_{T^*_1}(t) \) and assume that \( T^*_1(u) \) is inserted as a child of \( r \) in the current recursion. Let \( x \in L(T^*_1(u)) \) be an inserted leaf. We consider two possible cases.

**Case 1:** \( x \in L(T^*_1(u)) \setminus L(T_1) \). Observe that \( x \in L(T^*_1) \setminus L(T_1) = (C_1 \cup Y_1 \cup \{ z \}) \setminus (C_1 \cup \{ z \}) = Y_1 \), and thus \( x \) has no neighbors in \( L(R) \). This is illustrated in Figure 5(1). Thus, we must argue that \( x \) is at distance more than \( k \) to any leaf \( y \) of \( R \). Let \( x' \) be the lowest ancestor of \( x \) in \( T^*_1 \) (i.e., farthest from the root) such that \( x' \in V(T_1) \). Note that if \( T^*_1(u) \) was inserted on line 7, then \( x' = t \) since \( u \notin V(T_1) \) but \( t \in V(T_1) \) by Claim 1, and otherwise, \( x' \) is a descendant of \( u \). In any case, \( x' \) is in \( V(T_1(t)) \). Recall that \( T^*_1 = (T_1, \sigma_1) \), and that \( \sigma_1(x') \) is the minimum distance from \( x' \) to a leaf descending from a node in \( ch_{T^*_1}(x') \). We note that \( x \) is such a leaf, and thus \( \text{dist}_{T^*_1}(x, x') \geq \sigma_1(x') \).

Also recall that by Claim 1, \( r \in V(T_2) \cap V(T_3) \) and \( \text{sig}_{\ell_1}(T_1(t)) = \text{sig}_{\ell_2}(T_2(r)) = \text{sig}_{\ell_3}(T_3(r)) \). By Lemma 3.2.3, there is \( x'_2 \in V(T_2(r)) \) such that \( \text{dist}_{T^*_1}(x', t) = \text{dist}_{T^*_2}(x'_2, r) \) and \( \text{sig}_{\ell_1}(T_1(x')) = \text{sig}_{\ell_2}(T_2(x'_2)) \). In particular, \( \sigma_1(x') = \sigma_2(x'_2) \). Since \( T_2 \) is the valued restriction of \( T^*_2 \) to \( C_2 \cup \{ z \} \), this implies that there exists \( x_2 \in L(T_2) \setminus L(T_2) = Y_2 \), descending from a node in \( ch_{T^*_2}(x'_2) \), such that \( \text{dist}_{T^*_2}(x_2, x'_2) = \text{dist}_{R^*}(x_2, x'_2) = \sigma_2(x'_2) = \sigma_1(x') \leq \text{dist}_{T^*_1}(x, x') \). Since the distance from \( x'_2 \) to \( r \) is no more than the distance from \( x' \) to \( t \), this also implies that \( \text{dist}_{R^*}(x_2, r) \leq \text{dist}_{T^*_1}(x, t) \). Now, let \( y \in L(R) \setminus (C_2 \cup Y_2) \). We have \( x_2y \notin E(G) \) and obtain

\[
 k < \text{dist}_{R^*}(x_2, y) \leq \text{dist}_{R^*}(x_2, r) + \text{dist}_{R}(r, y) \\
= \text{dist}_{T^*_1}(x, t) + \text{dist}_{R}(r, y) \\
= \text{dist}_{R^*}(x, r) + \text{dist}_{R}(r, y) \\
= \text{dist}_{R^*}(x, y),
\]

where, for the last two equalities, we used the fact that \( T^*_1(u) \) was inserted as a child subtree of \( r \), and thus the path from \( x \) to \( y \) in \( R' \) must first go to \( r \), and then to \( y \). This shows that \( x \) has the correct distance to all \( y \in L(R) \setminus (C_2 \cup Y_2) \). It remains to handle the vertices in \( C_2 \cup Y_2 \). For the moment, let us instead consider \( y \in L(R) \setminus (C_3 \cup Y_3) \). We can apply the same argument as earlier, but using \( T_3 \) instead. In other words, by the equality of the signatures, there is some \( x_3 \in Y_3 \) such that \( \text{dist}_{T^*_1}(x_3, r) = \text{dist}_{R^*}(x_3, r) \leq \text{dist}_{T^*_1}(x, t) \). By repeating the preceding argument, we deduce that for any \( y \in L(R) \setminus (C_3 \cup Y_3) \), \( \text{dist}_{R}(x_3, y) > k \), which in turn implies \( \text{dist}_{R^*}(x, y) > k \). In particular, this holds for any \( y \in C_2 \cup Y_2 \). We have therefore shown that \( \text{dist}_{R^*}(x, y) > k \) for any \( y \in L(R) \), and thus that \( x \) is correctly inserted.

**Case 2:** \( x \in L(T_1) \). In this case, \( u \in ch_{T}(t) \) and \( T^*_1(u) \) must have been inserted as a child of \( r \) on line 11 or on line 16, which are illustrated in Figure 5(2) and (3), respectively. Note that \( x \in C_1 \) (\( x = z \) is not possible since \( z \) is not reinserted in \( R' \)).
Let us define $u_2 \in ch_{T_r}(r)$ and $u_3 \in ch_{T_l}(r)$ to handle both cases. If $T_r^1(u)$ is inserted on line 16, then define $u_2$ and $u_3$ as in the algorithm. In this case, we must have $u_2 \neq u_3$. If instead $T_l^1(u)$ is inserted on line 11, then there is $w \in ch_{T_l}(t)$ such that $\text{sig}_{\ell}(T_l(u)) = \text{sig}_{\ell}(T_l(w))$ and $w \neq u$. By Lemma 3.2.2, there exist distinct $u_2, u_2' \in ch_{T_l}(r)$ and $u_3, u_3' \in ch_{T_l}(r)$ such that $\text{sig}_{\ell}(T_l(u_2)) = \text{sig}_{\ell}(T_l(u_2')) = \text{sig}_{\ell}(T_l(u_3)) = \text{sig}_{\ell}(T_l(u_3'))$. Observe that at least one of $u_2 \neq u_3$ or $u_2 \neq u_2'$ holds. Assume without loss of generality that $u_2 \neq u_3$.

In either case, the important property to observe is that $u_2$ and $u_3$ are defined so that $u_2 \neq u_3$, $u_2 \in ch_{T_l}(r)$, $u_3 \in ch_{T_r}(r)$, $\text{sig}_{\ell}(T_l(u)) = \text{sig}_{\ell}(T_r(u_2)) = \text{sig}_{\ell}(T_r(u_3))$.

By Lemma 3.2.3, there exist $x_2 \in L(T_l(u_2))$ such that $\text{dist}_{T_l}(x_2, u_2) = \text{dist}_{T_l}(x, u)$ and $\text{sig}_{\ell}(T_l(x)) = \text{sig}_{\ell}(T_l(x_2))$. Since the signature of a leaf is just its layer, this implies that $\ell_1(x) = \ell_2(x_2)$. Also note that $\text{dist}_{T_r}(x, r) = \text{dist}_{T_r}(x_2, r)$. Similarly, there exists $x_3 \in L(T_r)$ such that $\ell_1(x) = \ell_3(x_3)$ and such that $\text{dist}_{T_r}(x_3, u_3) = \text{dist}_{T_r}(x, u)$ (i.e., that $\text{dist}_{T_r}(x, r) = \text{dist}_{T_r}(x_3, r)$). Also note that in $R'$, the paths from $x$ to $x_2$ to $r$ and $x_3$ to $r$ only intersect at $r$, since $u, u_2$ and $u_3$ are all distinct children of $r$.

First consider $y \in L(R) \setminus (C_2 \cup Y_2 \cup C_3 \cup Y_3)$. Since the layers are all equal, by Property 4b (same layer implies same neighbors) of similar structures, $xy \in E(G) \iff x_2y \in E(G) \iff x_3y \in E(G).$ If $y$ is not a descendant of $u_2$, then the path from $x_2$ to $y$ goes through $r$, implying $\text{dist}_{T_l}(x, y) = \text{dist}_{T_r}(x, y)$. If $y$ is not a descendant of $u_3$, then the path from $x_3$ to $y$ goes through $r$, implying $\text{dist}_{T_r}(x, y) = \text{dist}_{T_r}(x_3, y)$. Since $u_2 \neq u_3$, $y$ cannot be a descendant of both $u_2$ and $u_3$, and so at least one of $\text{dist}_{T_l}(x, y) = \text{dist}_{T_l}(x_2, y)$ or $\text{dist}_{T_r}(x, y) = \text{dist}_{T_r}(x_3, y)$ must hold. If $xy \in E(G)$, then $x_2y, x_3y \in E(G)$ and $\text{dist}_{T_l}(x_2, y) \leq k$, $\text{dist}_{T_r}(x_3, y) \leq k$ both hold, implying $\text{dist}_{T_r}(x, y) \leq k$. Similarly, if $xy \notin E(G)$, then $x_2y, x_3y \notin E(G)$ and, in the same manner, and $\text{dist}_{T_r}(x_2, y) > k$, $\text{dist}_{T_l}(x_3, y) > k$ both hold, implying $\text{dist}_{T_l}(x, y) > k$.

Now consider $y \in C_2 \cup Y_2$. Assume that $xy \notin E(G)$. Then by Property 4b, $x_3y \notin E(G)$. If $y$ is not a descendant of $u_3$, then the path from $x_3$ to $y$ goes through $r$ and $\text{dist}_{T_r}(x, y) = \text{dist}_{T_r}(x_3, y) > k$. If $y$ is a descendant of $u_3$, then we have $k \leq \text{dist}_{T_r}(x_3, y) \leq \text{dist}_{T_r}(x_3, r) + \text{dist}_{T_r}(r, y) = \text{dist}_{T_r}(x, r) + \text{dist}_{T_r}(r, y)$, and since $\text{dist}_{T_r}(x, r) + \text{dist}_{T_l}(r, y) = \text{dist}_{T_l}(x, y)$, the distance from $x$ to $y$ is correctly above $k$.

Assume that $xy \in E(G)$. Again by Property 4b, $x_3y \in E(G)$. Note that $y \in Y_2$ is not possible since $Y_2$ only has neighbors in $C_2 \cup Y_2$. So we know that $y \in C_2$, and hence $\ell_2(y)$ is defined. If $y$ is not a descendant of $u_3$, the usual argument applies, since the path from $x_3$ to $y$ goes through $r$, implying $\text{dist}_{T_r}(x, y) = \text{dist}_{T_r}(x_3, y) \leq k$. Suppose that $y$ is a descendant of $u_3$, as shown in Figure 6. By Property 4d (far layers implies non-edge) of similar structures, we must have $\ell_1(x) + \ell_2(y) \leq k$ (because if $\ell_1(x) + \ell_2(y) > k$, $xy$ could not be an edge). Since $\ell_1(x) = \ell_2(x_2)$, we have $\ell_2(x_2) + \ell_2(y) \leq k$ as well. Luckily, Property 4c (close layers implies edge) is applicable to vertices in the same $C_1$, and so $x_2y \in E(G)$. The path from $x_2$ to $y$ goes through $r$, and thus $\text{dist}_{T_r}(x, y) = \text{dist}_{T_r}(x_2, y) \leq k$. This covers the case $y \in C_2 \cup Y_2$.

The last remaining case is $y \in C_3 \cup Y_3$. This can be handled exactly as the previous case, but swapping the roles of $x_2$ and $x_3$. This completes the proof, as we have covered every possible case for a leaf inserted by the algorithm, at any point.

To conclude the proof, we can argue that $R'$ is a $k$-leaf power of $G$ by looking at every pair of vertices of $G$. First note that by Claim 2, all the leaves of $T_r^1$ are present in $R'$, and so $V(G) = L(R')$. Let $x, y \in V(G)$, with $x \neq y$. If $x, y \in V(G) \setminus (C_1 \cup Y_1)$, then $x, y \in L(R)$ and, since $\text{dist}_{T_r}(x, y) = \text{dist}_{T_r}(x, y)$, we know that $xy \in E(G) \iff \text{dist}_{T_r}(x, y) \leq k$. If $x, y \in C_1 \cup Y_1 \cup \{z\}$, then by Claim 2, $R'[\{C_1 \cup Y_1 \cup \{z\}\}]$ is leaf-isomorphic to $T_r^1$. Since $T_r^1$ is a $k$-leaf root of $G[C_1 \cup Y_1 \cup \{z\}]$, we know that $xy \in E(G) \iff \text{dist}_{T_r}(x, y) \leq k$. Finally, if $x \in C_1 \cup Y_1$ and $y \in V(G) \setminus (C_1 \cup Y_1)$, Claim 3 ensures that $xy \in E(G) \iff \text{dist}_{T_r}(x, y) \leq k$. Therefore, $R'$ is a $k$-leaf power of $G$. 

ACM Transactions on Algorithms, Vol. 19, No. 4, Article 35. Publication date: September 2023.
Fig. 6. The final case of Claim 3. The fat edges represents those of the $T'_1(u)$ subtree that contains $x$ and was inserted into $R$.

4 COMPUTING THE SET OF ACCEPTED SIGNATURES

We have not yet discussed how to find a homogeneous similar structure $S$. Since $k$ is fixed, it is not too hard to find a similar structure of $G$ with the properties of Lemma 3.7, if one exists. It suffices to brute-force every combination of $3|S(k,3k)|$ subsets of $V(G)$ of size at most $d^k$ in $G$ to find the $C_i$’s, and check that all similar structure properties hold. In particular, each $G[C_j \cup Y_i \cup \{z\}]$ has maximum degree at most $d^k$, so using Lemma 2.2, we can check whether this is a $k$-leaf power. However, this is not enough, since homogeneity requires enumerating all accepted signatures for the found $C_j \cup Y_i \cup \{z\}$, in order to construct $\text{accept}(S, C_j \cup Y_i \cup \{z\})$ and ensure that they are all equal. We will achieve this through a tree decomposition of $G[C_j \cup Y_i \cup \{z\}]$.

Let us assume that $S = (C, \mathcal{Y}, z, L)$ is a similar structure satisfying $|C| = 3|S(k,3k)|$, with each $|C_i| \leq d^k$ such that $G[C_j \cup Y_i \cup \{z\}]$ has maximum degree at most $d^k$. We want to compute $\text{accept}(S, C_i)$ for each $i \in [d]$. This can be achieved using a slightly more general result.

Lemma 4.1. Let $G$ be a connected graph of maximum degree at most $d^k$, and let $z \in V(G)$. Then in time $O(n \cdot d^{k^2} \cdot f(k)^4)$, where $n = |V(G)|$ and $f(k) = d^{4kd^k} \cdot (k+2)^d^k$, one can enumerate the set of all valued trees $\mathcal{T} = \{T_1, \ldots, T_l\}$, up to value-isomorphism, such that $T_i \in \mathcal{T}$ if and only if there exists a $k$-leaf root $T^*$ of $G$ such that (1) $T^*$ is rooted at the parent of $z$ and (2) $T_i$ is the valued restriction of $T^*$ to $N_G[z]$.

To see why Lemma 4.1 allows us to compute $\text{accept}(S, C_i)$, recall that the latter contains the signature of every $T$ that is the valued restriction of $T^*$ to $C_i \cup \{z\}$, where $T^*$ is a $k$-leaf root of $G[C_j \cup Y_i \cup \{z\}]$ with the root as the parent of $z$. Since $N_{G[C_j \cup Y_i \cup \{z\}]}[z] = C_j \cup \{z\}$, we can pass $G[C_j \cup Y_i \cup \{z\}]$ and $z$ to the preceding lemma. By taking the signature of every $T_i$ returned, we obtain $\text{accept}(S, C_i)$. Note that the lemma does not deal with layers, so the leaf sets of the desired $T_i$’s are $N_G[z]$ (instead of integers representing layers as in the previous section). Nonetheless, the valued trees in $\mathcal{T}$ can easily be converted into signatures by replacing the $C_i$ leaves by their layer.

The rest of this section is dedicated to the proof of Lemma 4.1. We will write $N(v)$ and $N[v]$ instead of $N_G(v)$ and $N_G[v]$, respectively, with the understanding that $G$ is the graph stated in Lemma 4.1. Likewise, for $X \subseteq V(G)$, we write $N(X)$ and $N[X]$ instead of $N_G(X)$ and $N_G[X]$, respectively. The proof is based on the tree decomposition of $G$ and uses a relatively standard dynamic programming algorithm. Recall that given a graph $G$, a nice tree decomposition of $G$ is a tree $B = (V_B, E_B)$ in which (1) $B_1 \subseteq V(G)$ for each $B_i \in V_B$; (2) for each $uv \in E(G)$, there is some $B_i \in V_B$ with $u, v \in B_i$; and (3) for each $u \in V(G)$, the set of $B_i$’s that contain $u$ form a connected subgraph of $B$. Moreover, each $B_i \in V_B$ can be one of four types: $B_i \in L(B)$, in which case $B_i = \{v\}$ for some $v \in V(G)$; $B_i$ is an introduce node, in which case $B_i$ has a single child $B_j$ with $B_j = B_i \setminus \{v\}$ for some $v \in B_i$; $B_i$ is a forget node, in which case $B_i$ has a single child $B_j$ with $B_j = B_i \setminus \{v\}$ for some $v \in B_j$; and $B_i$ is a join node, in which case $B_i$ has two children $B_lB_l$, and $B_i = B_l = B_r$.
The width of $B$ is $\max_{B_i \in V_B} (|B_i| - 1)$, and the treewidth of $G$ is the minimum width of a nice tree decomposition of $G$.

We note that Eppstein and Havelavi [19] also use a tree decomposition based algorithm. However, it does not seem adaptable directly for our purposes, since it is not guaranteed to return every structure of every $k$-leaf root of the given graph, and it does not maintain the $\sigma$ information that we need.

For our algorithm, we first check whether $G$ is chordal: if not, we know by Lemma 2.1 that $G$ is not a $k$-leaf power and we can return $T = \emptyset$. Otherwise, since $G$ has maximum degree at most $d^k$ and is chordal, $G$ has clique number at most $d^k + 1$ and thus treewidth at most $d^k$. The properties of the tree decomposition that we will need are summarized here.

**Lemma 4.2.** Let $G$ be a connected chordal graph of maximum degree at most $d^k$, and let $z \in V(G)$. Then there exists a nice tree decomposition $B = (V_B, E_B)$ of $G$ with $O(|V(G)|)$ nodes and of width at most $d^k$, such that $r(B) = \{z\}$ and such that each $B_i \in V_B$ is a non-empty clique.

We omit the proof, which uses standard arguments. The idea is that connected chordal graphs admit a tree decomposition in which every bag is a non-empty clique (e.g., see [2]). We can take such a decomposition, root it at a bag containing $z$, and apply the standard transformation to make it nice (if the root is not exactly $\{z\}$, it can be made $\{z\}$ by adding enough forget nodes above).

Let $B = (V_B, E_B)$ be a nice tree decomposition that satisfies all the properties of Lemma 4.2. For $B_i \in V_B$, we will denote by $V_i = \bigcup_{B_j \in B(B_i)} B_j$ the set of vertices of $G$ found in the bags at $B_i$ or its descendants. Note that for each $V_i, G[V_i]$ is connected. This can be seen inductively. First, it is true for leaves. At introduce nodes, we introduce a vertex $v$ connected to every vertex of $B_i$, so $G[V_i]$ remains connected. At forget nodes, $V_i$ is the same as $V_j$ and remains connected. At join nodes, $V_i$ is the union of the vertices of two connected graphs that intersect at $B_i$, and thus remains connected.

Let $B_i$ be a bag of $B$, and let $(T, \sigma)$ be a valued tree. We say that $(T, \sigma)$ is valid for $B_i$ if it satisfies the following conditions:

- $L(T) = N[B_i] \cap V_i$;
- there exists a $k$-leaf root $T^*$ of $G[V_i]$ such that $(T, \sigma)$ is the valued restriction of $T^*$ to $N[B_i] \cap V_i$, and such that $r(T) = r(T^*)$.

Since $r(B) = \{z\}$, our goal is to obtain the set all of valued trees that are valid for $r(B)$, as this will yield the valued restrictions to $N[z]$ required by Lemma 4.1. Note that unlike in the previous section, there is no requirement on the root of $T$ or $T^*$ being the parent of $z$. The requirement that $r(T) = r(T^*)$ is there to ensure that the root we see in the restricted $T$ is also the root in $T^*$ (since otherwise, the restriction could hide nodes of $T^*$ above the root of $T$). This is important for our purposes, since it allows us to safely look at the valid valued trees for $r(B) = \{z\}$ whose root is the parent of $z$. Also note that it is tempting to define $(T, \sigma)$ as the valued restriction of $T^*$ to $B_i$, and not bother with $N[B_i] \cap V_i$. This does not quite work—the inclusion of the neighborhood is necessary to retain enough information to update the $\sigma(\omega)$ values accurately (see the proof for details).

For the rest of this section, we shall slightly abuse notation and treat two valued trees that are value-isomorphic as the same. In other words, we assume the understanding that two valued trees are distinct only if they are not value-isomorphic.

We first show that the number of distinct $(T, \sigma)$ valid valued trees to consider is bounded (with a crude estimate), meaning that we can afford to enumerate all candidates.

**Lemma 4.3.** Let $B_i \in V(B)$, and let $(T, \sigma)$ be a valid valued tree for $B_i$. Then $T$ has at most $d^k$ nodes and is $k$-bounded. Consequently, there are at most $d^k d^k \cdot (k + 2)^{d^k}$ possible valid valued trees for $B_i$ (up to value-isomorphism).
The number of trees with at most $d^{4k}d^{4k}$ internal nodes is bounded by $(d^{4k})d^{4k} = d^{4kd^{4k}}$. We know that $(T, \sigma)$ is $k$-bounded by Lemma 3.4 (since $N[B_i] \cap V_i$ consists of a clique $B_i$ and a subset of $N[B_i]$). Thus, for each possible tree, each of the at most $d^{4k}$ internal nodes can receive a value in $\{0, 1, \ldots, k, \infty\}$ (i.e., $k + 2$ possibilities). Thus, the number of valued trees is bounded by $d^{4kd^{4k}} \cdot (k + 2)^{d^{4k}}$. \hfill \Box

We now describe a dynamic programming recurrence over $B$ that constructs a set $Q[B_i]$ for each $B_i$. The set $Q[B_i]$ must contain every valid valued tree for $B_i$—that is, $T \in Q[B_i]$ if and only if $T$ is valid for $B_i$. We assume that we are enumerating each candidate valued tree $T = (T, \sigma)$ from the preceding lemma and must decide whether to put it in $Q[B_i]$ or not. For convenience, denote $N[B_i] := N[B_i] \cap V_i$ for the rest of this section. We have the following:

- **Leaf node**: Let $B_i = \{v\}$ be a leaf of $B$. Then $(T, \sigma) \in Q[B_i]$ if and only if $T$ is the single node $v$ (and $\sigma$ has an empty domain).

- **Introduce node**: Let $B_i$ be an introduce node with child $B_j = B_i \cup \{v\}$. Then put $(T, \sigma) \in Q[B_i]$ if and only if there exists $(T_j, \sigma_j) \in Q[B_j]$ such that all the following conditions are satisfied:
  - $T|N[B_j] \approx_T T_j$.
  - Let $\mu$ be the leaf-isomorphism from $T|N[B_j]$ to $T_j$.
  - For every internal node $w \in V(T|N[B_j])$, $\sigma(w) = \sigma_j(\mu(w))$, and for every internal node $w \in V(T \setminus (V(T|N[B_j]))$, $\sigma(w) = \infty$.
  - For each $w \in L(T) \setminus \{v\}$, $vw \in E(G)$ if and only if $\text{dist}_T(v, w) \leq k$.
  - For each internal node $w \in V(T)$, $\text{dist}_T(v, w) + \sigma(w) > k$.

- **Forget node**: Let $B_i$ be a forget node with child $B_j = B_i \cup \{v\}$. Then put $(T, \sigma) \in Q[B_i]$ if and only if there exists $(T_j, \sigma_j) \in Q[B_j]$ such that
  - $r(T_j)$ has at least two distinct children $x, y$ such that $T_j(x)$ and $T_j(y)$ both have a leaf in $N[B_j]$.
  - $T \approx_T T_j|N[B_i]$.
  - Let $\mu$ be the leaf-isomorphism from $T$ to $T_j|N[B_i]$.
  - For each $w \in V(T) \setminus L(T)$, the value of $\sigma(w)$ is updated correctly with respect to $\sigma_j(\mu(w))$ and the leaves removed from $T_j$ by the restriction to $N[B_i]$. Precisely, let $C(w) = ch_{T_j}(\mu(w)) \setminus ch_{T_j}|N[B_i](\mu(w))$, and let
    \[
    \hat{L} = \bigcup_{c \in C(w)} L(T_j(c)) \quad \text{and} \quad \hat{I} = \{w\} \cup \bigcup_{c \in C(w)} V(T_j(c)) \setminus \hat{L}.
    \]
  - Then
    \[
    \sigma(w) = \min \left\{ \min_{w' \in I} \sigma_j(\mu(w')) + \text{dist}_{T_j}(\mu(w), \mu(w')), \min_{l \in \hat{L}} \text{dist}_{T_j}(\mu(w), l) \right\}.
    \]

- **Join node**: Let $B_i$ be a join node of $B$ with children $B_i$ and $B_r$, where $B_i = B_i = B_r$. Then $(T, \sigma) \in Q[B_i]$ if and only if there exists $(T_l, \sigma_l) \in Q[B_l]$ and $(T_r, \sigma_r) \in Q[B_r]$ that satisfy the following:
- \( T|\mathcal{N}[B_i] \approx_{L} T_l \). Let \( \mu_l \) be the leaf-isomorphism from \( T|\mathcal{N}[B_i] \) to \( T_l \).
- \( T|\mathcal{N}[B_r] \approx_{L} T_r \). Let \( \mu_r \) be the leaf-isomorphism from \( T|\mathcal{N}[B_r] \) to \( T_r \).
- For each \( w \in V(T) \setminus L(T) \), we have \( \sigma(w) = \min(\sigma_l(\mu_l(w)), \sigma_r(\mu_r(w))) \) if \( w \) is in both \( T|\mathcal{N}[B_i] \) and \( T|\mathcal{N}[B_r] \). \( \sigma(w) = \sigma_l(\mu_l(w)) \) if \( w \) is only in \( T|\mathcal{N}[B_i] \), and \( \sigma(w) = \sigma_r(\mu_r(w)) \) otherwise.
- For each distinct \( u, v \in L(T) \), \( uv \in E(G) \) if and only if \( \text{dist}_T(u, v) \leq k \).
- For each \( u \in \mathcal{N}[B_i] \setminus B_i \) and each internal node \( w \in V(T|\mathcal{N}[B_i]) \), \( \text{dist}_T(u, w) + \sigma_r(\mu_r(w)) > k \). Likewise, for each \( u \in \mathcal{N}[B_r] \setminus B_r \) and each \( w \in V(T|\mathcal{N}[B_r]) \), \( \text{dist}_T(u, w) + \sigma_l(\mu_l(w)) > k \).
- For each internal node \( w_l \in V(T|\mathcal{N}[B_i]) \) and each internal node \( w_r \in V(T|\mathcal{N}[B_r]) \), \( \sigma_l(\mu_l(w_l)) + \text{dist}_T(w_l, w_r) + \sigma_r(\mu_r(w_r)) > k \).

We will prove formally that these recurrences compute each \( Q[B_i] \) correctly as intended. Beforehand, we will make use of the following.

**Lemma 4.4.** Let \( G \) be a graph with \( k \)-leaf root \( T \). Let \( aa’ \) and \( bb’ \) be distinct edges of \( G \), where \( a \neq b \). Suppose that in \( T \), there exists a node \( w \) that lies on the path between \( a \) and \( a’ \), and on the path between \( b \) and \( b’ \). Then one of \( ab’ \) or \( a’b \) is an edge of \( G \).

**Proof.** If \( \text{dist}_T(a, w) \leq \text{dist}_T(b, w) \), then \( \text{dist}_T(a, b’) \leq \text{dist}_T(a, w) + \text{dist}_T(w, b’) \leq \text{dist}_T(b, w) + \text{dist}_T(w, b’) = \text{dist}_T(b, b’) \) since \( bb’ \) is an edge, which implies that \( ab’ \) is an edge of \( G \). If instead \( \text{dist}_T(b, w) < \text{dist}_T(a, w) \), then using an analogous chain of reasoning, the fact that \( aa’ \) is an edge implies that \( ba’ \) is an edge. \( \square \)

**Lemma 4.5.** Consider the recurrence given earlier. For any bag \( B_i \), \( (T, \sigma) \) is a valid valued restriction for \( B_i \) if and only if \( (T, \sigma) \in Q[B_i] \) (up to value-isomorphism).

**Proof.** The proof is by induction on the depth of \( B_i \). As a base case, assume that \( B_i \) is a leaf. Then \( B_i = \{v\}, \mathcal{N}[B_i] = \{v\} \), and \( (T, \sigma) \) with \( T \) containing \( v \) only is the only possible valid valued tree. Hence, the leaf case is correct.

Now consider the induction step. Let \( B_i \in V(B) \setminus L(B) \) and assume that the statement is correct for all descendants of \( B_i \). We prove the two directions of the statement separately.

\((\Rightarrow) \): suppose that \( (T, \sigma) \) is a valid valued restriction for \( B_i \). Then there exists a \( k \)-leaf power \( T^* \) of \( G[V_j] \) such that \( (T, \sigma) \) is the valued restriction of \( T^* \) to \( \mathcal{N}[B_i] \), and such that \( r(T) = r(T^*) \). We must argue that \( (T, \sigma) \) is in \( Q[B_i] \).

**Introduce Node.** Suppose that \( B_i \) is an introduce node, with child \( B_j = B_i \setminus \{v\} \). Notice that by the properties of tree decompositions, \( \mathcal{N}(v) \cap V_j = B_j \setminus \{v\} \) (since \( B_j \) is a clique).

Let \( T_j^* = T^*|V_j \), noting that \( V_j = V_i \setminus \{v\} \). Moreover, \( T_j^* \) is a \( k \)-leaf root of \( G[V_j] \), since distances between leaves do not change when taking a restriction. Let \( (T_j, \sigma_j) \) be the valued restriction of \( T_j^* \) to \( \mathcal{N}[B_j] \).

We would like to use induction and argue that \( (T_j, \sigma_j) \in Q[B_j] \), but for that we need \( r(T_j) = r(T_j^*) \), which is not immediate. Assume for the sake of contradiction that \( r(T_j) \neq r(T_j^*) \). This happens if and only if in \( T_j^* \), some ancestor of \( r(T_j) \) (other than itself) has descending leaves of \( V_j \) not in \( \mathcal{N}[B_j] \). Let \( L’ = V_j \setminus L(T_j^*(r(T_j))) \), and assume that \( L’ \) is non-empty. This is illustrated in Figure 7. For that to happen, observe that \( v \notin L(T^*(r(T_j))) \). This is because \( r(T) = r(T^*) \), and if \( v \) was a descendant of \( r(T_j) \) in \( T^* \), then the root of \( T = T^*|\mathcal{N}[B_i] \) would still be \( r(T_j) \neq r(T^*) \). Thus, in \( T^* \), the path from \( r(T_j) \) to \( v \) goes "above" \( r(T_j) \). Also note that no member of \( L’ \) has a neighbor in \( B_j \), as otherwise this member of \( L’ \) would be included in \( T_j = T_j^*|\mathcal{N}[B_j] \). Moreover, members of \( L’ \) are not neighbors of \( v \). Let \( u \in L’ \) be at minimum distance from \( r(T_j) \) in \( T_j^* \). Then \( u \) is also at minimum distance from \( r(T_j) \) in \( T^* \), among all leaves in \( L’ \). Note that \( u \) must have at
least one neighbor $u'$ in $L(T^* (r(T_j)))$ (otherwise, since $u$ is at minimum distance from $r(T_j)$, this would imply that no member of $L'$ has a neighbor outside of $L'$, and thus contradict that $G[V_j]$ is connected). Moreover, $u' \notin B_j$ since $u \notin N[B_j]$. Also note that since $B_j$ is non-empty and $B_i$ is a clique, $v$ has at least one neighbor $v'$ in $B_j$ (and thus $v'$ descends from $r(T_j)$). Thus, we have edges $uu'$ and $uv'$ in $E(G)$, and in $T^*$, both paths between $u, u'$ and $v, v'$ contain $r(T_j)$. By Lemma 4.4, one of $uv'$ or $vu'$ is an edge. But $uv' \in E(G)$ is not possible since $v' \in B_j$ and $u \notin N[B_j]$, and $uv'$ is not possible since $u' \notin B_j$. We deduce that $L'$ is empty. Hence, we can safely assume that $r(T^*_j) = r(T_j)$. Because $(T_j, \sigma_j)$ is also the valued restriction of $T^*_j$ to $N[B_j]$, it follows that $(T_j, \sigma_j)$ is valid for $B_j$ and, by induction, that $(T_j, \sigma_j) \in Q[B_j]$. It remains to show that $(T_j, \sigma_j)$ satisfies all conditions of introduce nodes.

Let us argue that $T|N[B_j] = T_j$ (which clearly implies leaf-isomorphism). Note that $V(T|N[B_j])$ is the set of all vertices of $T^*$ on the path between two leaves in $N[B_j]$, and $V(T_j)$ the set of all vertices of $T^*_j$ between two leaves in $N[B_j]$. This makes it clear that $T|N[B_j] = T_j$, but let us argue this for completeness. Consider two vertices $x, y \in N[B_j]$, and let $P_{xy}$ be the path of $T^*$ from $x$ to $y$. All vertices of $P_{xy}$ must be in $T$ since $x, y \in N[B_j]$, and all vertices of $P_{xy}$ are in $T|N[B_j]$ since $x, y \in N[B_j]$. All vertices of $P_{xy}$ must be in $T^*_j$ since $x, y \in V_j$, and must also all be in $T_j$ since $x, y \in N[B_j]$. Since this holds for every $x, y \in N[B_j]$, we obtain $T|N[B_j] = T_j$. We also have the leaf-isomorphism $\mu(w) = w$ for all $w \in V(T|N[B_j])$ since $T|N[B_j]$ and $T_j$ share the same set of vertices of $T^*$. We shall use $w$ and $\mu(w)$ interchangeably, since they represent the same node.

We next justify $\sigma(w) = \infty$ for every internal node $w \in V(T) \setminus V(T_j)$. There are two possible cases. First suppose that $r(T) \neq r(T_j)$. Recall that $L(T)$ and $L(T_j)$ differ only by the leaf $v$. Thus, $r(T) \neq r(T_j)$ happens if and only if $r(T)$ has exactly two distinct children in $T$ such that one leads to $v$ and the other leads to $r(T_j)$. In this situation, $w \in V(T) \setminus V(T_j)$ must be a node on one of these two paths (excluding $r(T_j)$ and $v$). As we have already argued, we have $r(T_j) = r(T^*_j)$ and thus all leaves of $T^*_j$ are descendants of $r(T_j)$ in $T^*$. Therefore, $w$ cannot have a child in $T^*$ that is not in $T_j$, as such a child would lead to a leaf in $V_j$ that is not a descendant of $r(T_j)$. It follows that $\sigma(w) = \infty$ represents the correct distance information for $w$.

The second case is when $r(T) = r(T_j)$. In this case, the only difference between $T_j$ and $T$ is that there is some vertex $x \in V(T_j)$ and a path $P = (x_1, \ldots, x_1 = v)$ such that $x_1 \in cht(x)$ and $x_1 \notin V(T_j)$. In other words, $T$ was obtained from $T_j$ by appending the path leading to $v$ under some node $x$. It follows that only the nodes $x_1, \ldots, x_{i-1}$ are internal nodes in $T$ but not in $T_j$. The recurrence assumes that $\sigma(x_i) = \infty$ for each $x_i$. Again, we must argue that in $T^*$, no such $x_i$ has a descending leaf in $L(T^*) \setminus N[B_j]$ (i.e., that $L(T^*(x_i)) = \{v\}$). Assume that this is not the case. Let $u \neq v$ be a leaf of $T^*(x_1)$ at minimum distance from $x$. This is similar to the previous situation. We
have that \( u \notin \mathcal{N}(B_j) \) since \( x_j \) is not in \( T_j \). There must exist \( u' \in \mathcal{N}(u) \) such that \( u' \notin L(T^*(x_1)) \) (otherwise, no leaf in \( L(T^*(x_1)) \setminus \{v\} \) has a neighbor in \( G \) outside of \( L(T^*(x_1)) \) and \( G[V_j] \) is not connected). In addition, \( u' \notin B_j \). Moreover, \( v \) has a neighbor \( v' \) in \( B_j \). The paths between \( u, u' \) and \( v, v' \) both go through \( x \), and by Lemma 4.4 one of \( uv' \) or \( vu' \) is an edge of \( G \). But \( uv' \) is not possible since \( v' \in B_j \), and \( vu' \) is not possible since \( u' \notin B_j \). Thus, in \( T^* \), each \( x_j \) only has \( v \) with a leaf descendant. This justifies putting \( \sigma(x_j) = \infty \) for each \( x_j \).

Next, let \( w \in V(T) \setminus \mathcal{N}(B_j) \) be an internal node that is in both \( T \) and \( T_j \). Let \( w' \in ch_{T^*}(w) \setminus ch_T(w) \). Then \( w' \) only leads to leaves in \( V_j \setminus \mathcal{N}(B_j) = V_j \setminus \mathcal{N}(B_j) \) and must also be in \( ch_{T^*}(w) \setminus ch_T(w) \). Conversely, let \( w'' \in ch_{T^*}(w) \setminus ch_T(w) \). Then in \( T_j \), \( w'' \) only leads to leaves in \( V_j \setminus \mathcal{N}(B_j) \). We have a problem if \( w'' \in ch_T(w) \). Again, since \( L(T) \) and \( L(T_j) \) differ only by \( v \), this is only possible if \( w'' \) has \( v \) as a descendant. But as we argued in the previous paragraph, \( w'' \) would not have descendants in \( V_j \setminus \mathcal{N}(B_j) \) (here, \( w'' \) plays the same role as \( x_j \)). Therefore, we may assume that \( w'' \in ch_{T^*}(w) \setminus ch_T(w) \). Since the set of children of \( w \in T^* \) not in \( T \) is the same as those in \( T_j \) not in \( T_j \), we deduce that \( \sigma(w) = \sigma(v) \) represents the correct distance information.

In addition, since \( T = T^* | \mathcal{N}(B_j) \) and \( T^* \) is a \( k \)-leaf root of \( G \), it is clear that for all \( w \in L(T) \setminus \{v\} \), \( v \in E(G) \Leftrightarrow dist_T(v, w) \leq k \). Moreover, since we have argued that \( \sigma(w) \) is correct for each \( w \in V(T) \setminus L(T) \), we must have \( dist_T(v, w) + \sigma(w) > k \), as otherwise \( v \) would have a neighbor outside of \( \mathcal{N}(B_j) \). This agrees with the recurrence, and it will therefore put \( (T, \sigma) \in Q[B_j] \).

**Forget Node.** Suppose that \( B_1 \) is a forget node, with child \( B_j = B_i \cup \{v\} \). Note that \( T^* \) is a \( k \)-leaf root of \( G[V_j] \). Let \( (T_j, \sigma_j) \) be the valued restriction of \( T^* \) to \( \mathcal{N}(B_j) \). Because \( \mathcal{N}(B_j) \subseteq \mathcal{N}(B_j) \), the root of \( T_j \) satisfies \( r(T_j) = r(T) = r(T^*) \). Then by induction, \( (T_j, \sigma_j) \in Q[B_j] \).

Note that \( r(T) \) must have two distinct children with descendants in \( \mathcal{N}(B_i) \), and thus the same holds for \( T_j \), as in the recurrence. Because \( T = T^* | \mathcal{N}(B_j) \) and \( T_j = T^* | \mathcal{N}(B_j) \), it is not hard to see that \( T = T_j | \mathcal{N}(B_j) \). The leaf-isomorphism is \( \mu(w) = w \) for all \( w \in V(T) \) because \( T \) and \( T_j | \mathcal{N}(B_j) \) use the same set of vertices.

Now let \( w \in V(T) \setminus L(T) \). Let \( L^* = \bigcup_{c \in ch_{T^*}(w) \setminus ch_T(w)} L(T^*(c)) \). To find the minimum distance from \( w \) to a leaf \( u \in L^* \), we must first consider all leaves of \( L(T_j) \) that descend from a child \( c \in ch_{T^*}(w) \setminus ch_T(w) \). The minimum distance to such a leaf is given by \( \min_{l \in L} dist_{T_j} (\mu(w), l) \) in the recurrence. We must also consider all leaves of \( V_j \setminus L(T_j) \) that descend from a child \( c \in ch_{T^*}(w) \setminus ch_T(w) \). Consider such a leaf \( u \) at minimum distance from \( w \), and let \( w' \) be the lowest ancestor of \( u \) that is in \( V(T_j) \) (which exists, since \( w \) is an ancestor of \( u \)). Then either \( w' = w \) if \( u \) descends from some child \( c \in ch_{T^*}(w) \setminus ch_T(w) \), in which case the distance is \( \sigma_j(w) \), or \( w' \) is a descendant of some child of \( w \) in \( ch_T(w) \setminus ch_T(w) \), in which case this distance is \( \sigma_j(w') + dist_{T_j}(\mu(v), \mu(w)) \). In any case, the distance to such a \( u \) is given by the expression \( \min_{v \in l} \sigma_j(\mu(v)) + dist_{T_j}(\mu(v), \mu(w)) \) in the recurrence. Since the recurrence takes the minimum over all possibilities, it will take the minimum possible distance, and thus \( \sigma(w) \) takes the same distance as in the recurrence.

**Join Node.** Suppose that \( B_i \) is a join node, with children \( B_i = B_r \). In this part of the proof, we will use \( B_i, B_i, \) and \( B_r \) interchangeably, but use \( B_i \) and \( B_r \) when we want to emphasize that we are dealing with the “left” or “right” side of the decomposition. Let \( T^*_i = T^* | V_i \) and \( T^*_r = T^* | V_r \). Let \( (T_i, \sigma_i) \) be the valued restriction of \( T^*_i \) to \( \mathcal{N}(B_i) \). Similarly, let \( (T_r, \sigma_r) \) be the valued restriction of \( T^*_r \) to \( \mathcal{N}(B_i) \).

As in the introduction case, we would like to argue that \( r(T_i) = r(T^*_i) \). Assume for contradiction that \( r(T_i) \neq r(T^*_i) \). Then there exists \( u \in L(T^*(r(T_i))) \setminus L(T^*(r(T_i))) \). Choose \( u \) among this set such that \( u \) has minimum distance to \( r(T_i) \). Notice that \( u \notin \mathcal{N}(B_i) \), and thus \( u \in V_i \setminus \mathcal{N}(B_i) \). This in turn implies that \( u \notin \mathcal{N}(B_i) \). Moreover, since \( r(T^*_i) = r(T)_r \), \( r(T^*_i) \) has at least two distinct children with a descendant in \( \mathcal{N}(B_i) \). Let \( w \) be such a child, chosen so that \( w \) does not have \( r(T) \) as a descendant. Let \( v \in L(T^*(w)) \setminus \mathcal{N}(B_i) \). Note that \( v \notin \mathcal{N}(B_i) \) by the choice of \( w \). Thus, \( v \notin \mathcal{N}(B_i) \setminus \mathcal{N}(B_i) \), which
implies that $v \in V_i \setminus B_i$. By the connectedness property of tree decompositions, this implies $u \neq v$. One can see that $u$ must have a neighbor $u'$ in $L(T^*(r(T_i)))$ (if not, by the choice of $u$, all the leaves in $L(T^*(r(T_i))) \setminus L(T(r(T_i)))$ have only neighbors in $L(T^*(r(T_i))) \setminus L(T(r(T_i)))$, contradicting that $G[V_i]$ is connected). In addition, $u' \in V_i \setminus B_i$ since $u \notin N[B_i]$. Moreover, $v \notin B_i$ implies that $v$ has a neighbor $v'$ in $B_i$, which descends from $r(T_i)$. The paths between $u, u', v', v''$ both go through $r(T_i)$, and by Lemma 4.4 one of $uv'$ or $v'u'$ is an edge of $G$. But $uv'$ is not possible since $u \notin N[B_i]$ and $v' \in B_i$, and $v'u'$ is not possible since $u' \in V_i \setminus B_i$ and $v \in V_i \setminus B_i$, which is not allowed by the properties of tree decompositions. We deduce that $r(T_i) = r(T_i^*)$. By a symmetric argument, $r(T_i) = r(T_i^*)$. By induction, $(T_i, \sigma_i) \in Q[B_i]$ and $(T_r, \sigma_r) \in Q[B_r]$.

Since $T = T^*|N[B_i]$ and $N[B_i] \subseteq N[B_r]$, it is not hard to see that $T|N[B_i] = T_i$, with the leaf-isomorphism $\mu_l(w) = w$ for $w \in V(T|N[B_i])$. Likewise, $T|N[B_r] = T_r$, with the leaf-isomorphism $\mu_r(w) = w$ for $w \in V(T|N[B_r])$. We shall not distinguish $w$ and $\mu_l(w), \mu_r(w)$ since they refer to the same node.

For the next condition of the recurrence, let $w \in V(T) \setminus L(T)$. We claim that $w$ is in at least one of $T|N[B_i]$ or $T|N[B_r]$. If there is some $v \in B_i$ that is not a descendant of $w$ in $T$, then $w$ is on the path between $v$ and some leaf in $N[B_i]$ or $N[B_r]$ that descends from $w$, in which case $w$ must be in at least one of $T|N[B_i]$ or $T|N[B_r]$. If $w$ has every element of $B_i$ in its descendants in $T$, then $w$ is on the path between $v \in B_i$ and some leaf $v' \in N[B_i] \cup N[B_r]$ such that the lowest common ancestor between $v$ and $v'$ in $T$ is $r(T)$ (such a leaf must exist). Thus, our claim holds. If $w$ is in both $T|N[B_i]$ and $T|N[B_r]$, then $\sigma(w) = \min(\sigma_l(w), \sigma_r(w))$ holds because all leaves of $V_i \setminus N[B_i]$ that have $w$ as their first ancestor in $T$ are either leaves of $V_i \setminus N[B_i]$ or $V_r \setminus N[B_r]$, and $\sigma_l(w), \sigma_r(w)$ cover both cases.

Suppose that $w$ is in $T|N[B_i]$ but not in $T|N[B_r]$. Let $C(w) = ch_{T^*}(w) \setminus ch_T(w)$. If, for all $w' \in C(w)$, $L(T^*(w'))$ is a subset of $V_i$, then having $\sigma(w) = \sigma(w)$ as in the recurrence is correct. So, assume that there is $w' \in C(w)$ such that $w'$ has a descending leaf $\tilde{u}$ with $\tilde{u} \notin V_i$. We infer that $\tilde{u} \in V_r \setminus N[B_r]$. We argue that $w$ does not have descendants in $N[B_r]$. If $w$ has all elements of $N[B_r]$ in its descendants, the fact that $w \notin V(T|N[B_r])$ and the existence of $\tilde{u}$ contradict that $r(T_i) = r(T_i^*)$. If $w$ has some but not all elements of $N[B_r]$ in its descendants, some path between elements of $N[B_r]$ use $w$ and it should be in $T|N[B_i]$. This $w$ has no descendant in $N[B_r]$.

Under that assumption, let $u \in V_r \setminus N[B_r]$ be a descendant of $w$ in $T^*$ whose distance to $w$ is minimum among the possibilities (such a $u$ exists since $\tilde{u}$ exists). We have that $u$ has a neighbor $u' \in V_r \setminus B_r$ that does not descend from $w$ in order for $G[V_r]$ to be connected (this is because $w$ has no descendant in $N[B_i]$). Let $v \in L(T(w))$, noting that $v \in N[B_i] \setminus N[B_r]$ (and thus $v \in V_i \setminus B_i$). Thus, $v$ has a neighbor $v' \in B_i = B_r$, and so the path from $u$ to $v'$ goes through $w$. By Lemma 4.4, one of $uv'$ or $v'u'$ is an edge. But $uv'$ is not possible since $v' \in B_r$ and $u \notin N[B_r]$, and $v'u'$ is not possible by the properties of tree decompositions since $v \in V_i \setminus B_i$ and $u' \in V_r \setminus B_r$. We deduce that $C(w)$ has no descending leaf outside of $V_i$, and that $\sigma(w) = \sigma_l(w)$ is correct.

Suppose that $w$ is in $T|N[B_i]$ but not in $T|N[B_i]$. A symmetric argument justifies that $\sigma_l(w) = \sigma_r(w)$.

The fact that for $u, v \in L(T), uv \in E(G) \Leftrightarrow \text{dist}_T(u, v) \leq k$ follows from the fact that $T = T^*|N[B_i]$ and that $T^*$ is a $k$-leaf root of $G[V_i]$.

Now let $u \in N[B_i] \setminus B_i$ and take an internal node $w \in V(T|N[B_i])$. We note that $u \in V_i \setminus B_i$. Moreover in $T^*(w)$, there is a leaf $v$ of $V_r \setminus N[B_r]$ at distance $\sigma_r(w)$ from $w$. It follows that $uv \notin E(G)$, and thus that $\text{dist}_{T^*}(u, w) + \sigma_r(w) > k$. This implies that $\text{dist}_{T^*}(u, w) + \sigma_r(w) > k$. A symmetric argument justifies the same property for $u \in N[B_r] \setminus B_r$.

Finally, let $w_l \in V(T|N[B_i])$ and $w_r \in V(T|N[B_r])$. Then any leaf of $T^*(w_l) \setminus N[B_i]$ must be at distance more than $k$ to any leaf of $T^*(w_r) \setminus N[B_r]$. This justifies $\sigma_l(w_l) + \text{dist}_{T^*}(w_l, w_r) + \sigma_r(w_r) > k$. 

ACM Transactions on Algorithms, Vol. 19, No. 4, Article 35. Publication date: September 2023.
(⇐) Suppose that \((T, \sigma) \in Q[B_j]\). We must argue that there is a \(k\)-leaf root \(T^*\) of \(G[V_j]\), with \(r(T^*) = r(T)\), such that \((T, \sigma)\) is the valued restriction of \(T^*\) to \(N[B_j]\).

**Introduce Node.** Let \(B_j\) be an introduce node with child \(B_j = B_i \cup \{v\}\). Then there exists \((T_j, \sigma_j) \in Q[B_j]\) that satisfies the properties of the recurrence. Let \(\mu\) be the leaf-isomorphism from \(T|N[B_j]\) to \(T_j\). By induction, there is a \(k\)-leaf power \(T^*_j\) of \(G[V_j] = G[V_i \setminus \{v\}]\) such that \((T_j, \sigma_j)\) is the valued restriction of \(T^*_j\) to \(N[B_j]\), and such that \(r(T^*_j) = r(T_j)\).

We can construct a \(k\)-leaf root \(T^*_j\) of \(G[V_j]\) as follows. For each \(v \in V(T|N[B_j])\), let \(U(\omega) = ch_{T^*_j}(\mu(\omega)) \setminus ch_{T_j}(\mu(\omega))\). Then for each \(u \in U(\omega)\), add the \(T^*_j(u)\) subtree as a child of \(T^*_j\) in \(T\). Because \(r(T^*_j) = r(T_j)\), every leaf in \(V_j \cap N[B_j]\) gets inserted. In essence, we simply add the subtrees of \(T^*_j\) that are missing at the locations indicated by \(T_j\).

We claim that \(T^*_j\) meets all the requirements for \((T, \sigma)\) to be valid. We note that \(r(T^*_j) = r(T)\) since in our construction, we started with \(T\), and only added subtrees under some of its internal nodes (thus, we have not accidentally “raised” the root in \(T^*_j\)). Let us argue that \((T, \sigma)\) is the valued restriction of \(T^*_j\) to \(N[B_j]\). Notice that \(T = T^*_j|N[B_j]\), since \(L(T) = N[B_j]\), and to obtain \(T^*\) we took \(T\) and only added extra subtrees under its existing nodes. Moreover, it is not hard to see that \(T^*_j|V_j \simeq T^*_j\). This is because we started with \(T\) such that \(T|N[B_j] \simeq T_j\), and we added the missing subtrees of \(T^*_j\) on the vertices of \(T|N[B_j]\) at the appropriate locations.

Consider \(w \in V(T) \setminus L(T)\) and let us argue that \(\sigma(w)\) represents the correct distance to a hidden leaf under \(w\). First assume that \(w \notin V(T|N[B_j])\). Then one can see from the construction that we never insert subtrees under \(w\), and hence \(\sigma(w) = \infty\) represents the correct distance. Assume instead that \(w \in V(T|N[B_j])\). Then our insertion procedure only adds the leaves descending from children in \(ch_{T^*_j}(\mu(\omega)) \setminus ch_{T_j}(\mu(\omega))\) under \(w\). The \(\sigma_j(\mu(\omega))\) quantity gives the minimum distance from \(\mu(\omega)\) to such an inserted leaf, and in this case the recurrence specifies that \(\sigma(w) = \sigma_j(\mu(\omega))\), which is correct.

Let us now argue that \(T^*_j\) is a \(k\)-leaf root of \(G[V_j]\). Because \(T^*_j|V_j \simeq L T^*_j\), for each \(x, y \in V_j, xy \in E(G) \Rightarrow dist_T(x, y) \leq k\). As for \(v\), we know by the recurrence that for each \(w \in N[B_i]\), \(vw \in E(G) \Rightarrow dist_T(v, w) = dist_T^*(v, w) \leq k\). For any \(w \in V_j \setminus N[B_i]\), let \(w^*\) be the lowest ancestor of \(w\) in \(T^*_j\) such that \(w^* \in V(T)\), which exists since \(r(T) = r(T^*_j)\). The recurrence ensures that \(dist_T(v, w^*) + \sigma(w^*) > k\), and thus that \(dist_T^*(v, w) = dist_T(v, w^*) + dist_T^*(w^*, w) > k\). It follows that \(T^*_j\) is a \(k\)-leaf root of \(G[V_j]\).

**Forget Node.** Let \(B_i\) be a forget node with child \(B_j = B_i \cup \{v\}\). Then there exists \((T_j, \sigma_j) \in Q[B_j]\) that satisfies the properties of the recurrence. Let \(\mu\) be the leaf-isomorphism from \(T\) to \(T_j|N[B_j]\).

By induction, there is a \(k\)-leaf power \(T^*_j\) of \(G[V_j]\) such that \((T_j, \sigma_j)\) is the valued restriction of \(T^*_j\) to \(N[B_j]\), and whose root is \(r(T^*_j) = r(T_j)\). Since \(V_j = V_i\), \(T^*_j\) is also a \(k\)-leaf root of \(G[V_i]\). Let us note that since the recurrence requires \(r(T)\) to have two children with descending leaves in \(N[B_i]\), we have \(\mu(r(T)) = r(T_j) = r(T^*_j)\). We want to argue that \(T\) is the valued restriction of \(T^*_j\) to \(N[B_i]\). Strictly speaking, \(V(T)\) is not a subset of \(V(T^*_j)\), so we cannot say that \(T\) is the valued restriction of \(T^*_j\) to \(N[B_i]\) (recall that valued restrictions require usage of same set of nodes, whereas here we are using an isomorphism \(\mu\)). The simplest solution is to construct a new \(k\)-leaf root \(T^*\) “around” \(T\).

For each \(w \in V(T)\), let \(C(\omega) = ch_{T^*_j}(\mu(\omega)) \setminus ch_{T_j}(\mu(\omega))\). Then for each \(u \in C(\omega) \cup C^*(\omega)\), add the \(T^*_j(u)\) subtree as a child of \(T^*_j\) in \(T\). Because \(\mu(r(T)) = r(T_j) = r(T^*_j)\), we know that every leaf of \(V_j \setminus N[B_j]\) is inserted by this procedure. Moreover, it is not hard to see that \(T^* \simeq L T^*_j\), since we add all the missing subtrees to \(T\) at the appropriate locations specified by \(T_j\). Hence, \(T^*\) is a \(k\)-leaf root of \(G[V_j]\). It remains to argue that \((T, \sigma)\) is the valued restriction of \(T^*\) to \(N[B_i]\).
Since $T^*$ is obtained from $T$ by inserting child subtrees under the nodes of $T$, we have $T = T^*|\mathcal{N}[B_i]$ and $r(T) = r(T^*)$. Now let $w \in V(T) \setminus L(T)$. Then in $T^*$, consider the minimum distance from $w$ to a leaf in $u \in V_1 \setminus \mathcal{N}[B_i]$ that descends from a node in $ch_{T^*}(w) \setminus ch_T(w)$. Because $T^* \equiv_L T_i^*$, this is identical to the $\Rightarrow$ direction of forget nodes. In other words, the $u$ leaf is either in $L(T_i) \setminus L(T)$, in which case $\text{dist}_{T^*}(w, u) = \min_{l \in L} \text{dist}_{T}(\mu_l(w), l)$ as in the recurrence, or this leaf is not in $T_i$.

In the latter case, $u$ has some $w'$ as its lowest ancestor in $T_i$, and the distance to $u$ is given by $\text{dist}_{T_i}(\mu_l(w), w') + \sigma_r(w')$. The value of $\sigma(w)$ should be the minimum of all possibilities, as in the recurrence. Thus, $(T, \sigma)$ is indeed the valued restriction of $T^*$ to $\mathcal{N}[B_i]$.

**Join Node.** Let $B_i$ be a join node with children $B_l, B_r$, with $B_l = B_i = B_r$. Then there exist $(I_l, \sigma_l) \in Q[B_l]$ and $(I_r, \sigma_r) \in Q[B_r]$ that satisfy the properties of the recurrence. Let $\mu_l$ and $\mu_r$ be the leaf-isomorphisms from $\mathcal{T}[\mathcal{N}[B_i]]$ to $T_l$, and from $\mathcal{T}[\mathcal{N}[B_r]]$ to $T_r$, respectively. By induction, there is a $k$-leaf root $T_i^*$ of $G[V_l]$ such that $(I_l, \sigma_l)$ is the valued restriction of $T_i^*$ to $\mathcal{N}[B_l]$, with $r(T_i^*) = r(T_l)$. Likewise, there is a $k$-leaf root $T_r^*$ of $G[V_r]$ such that $(I_r, \sigma_r)$ is the valued restriction of $T_r^*$ to $\mathcal{N}[B_r]$, with $r(T_r^*) = r(T_r)$.

We can construct $T^*$ as follows. For each $w \in V(T|\mathcal{N}[B_i])$, let $U_l(w) = ch_{T^*}(\mu_l(w)) \setminus ch_{T_l}(\mu_l(w))$. Then for each $u \in U_l(w)$, add the $\mathcal{T}_l^*(u)$ subtree as a child of $w$ in $T$. Because $r(T_l^*) = r(T_l)$, every leaf in $V_1 \setminus \mathcal{N}[B_i]$ gets inserted. Similarly, for each $w \in V(T|\mathcal{N}[B_r])$, let $U_r(w) = ch_{T^*}(\mu_r(w)) \setminus ch_{T_r}(\mu_r(w))$. Then for each $u \in U_r(w)$, add the $\mathcal{T}_r^*(u)$ subtree as a child of $w$ in $T$. Again, every leaf in $V_r \setminus \mathcal{N}[B_r]$ gets inserted.

We claim that $T^*$ meets all the requirements for $(T, \sigma)$ to be valid. We note that $r(T^*) = r(T)$ since in our construction, we started with $T$, and only added subtrees under some of its internal nodes. Let us argue that $(T, \sigma)$ is the valued restriction of $T^*$ to $\mathcal{N}[B_i]$. Notice that $T = T^*|\mathcal{N}[B_i]$, since $L(T) = \mathcal{N}[B_i]$, and to obtain $T^*$ we took $T$ and only added extra subtrees under its existing nodes. Moreover, it is not hard to see that $T^*|\mathcal{V}_l =_L T_l^*$. This is because we started with $T$ such that $\mathcal{T}[\mathcal{N}[B_i]] =_L T_l$, and we added the missing subtrees of $T_l^*$ on the vertices of $\mathcal{T}[\mathcal{N}[B_i]]$ at the appropriate locations. Similarly, $T^*|\mathcal{V}_r =_L T_r^*$.

Consider $w \in V(T) \setminus L(T)$ and let us argue that $\sigma(w)$ represents the correct distance to a hidden leaf under $w$. If $w \in V(T|\mathcal{N}[B_i])$ but is not in $V(T|\mathcal{N}[B_i])$, then our insertion procedure only adds the leaves descending from children in $ch_{T_l^*}(\mu_l(w)) \setminus ch_{T_l}(\mu_l(w))$ under $w$. The $\sigma_l(\mu_l(w))$ quantity gives the minimum distance from $\mu_l(w)$ to such an inserted leaf, and in this case the recurrence specifies that $\sigma(w) = \sigma_l(\mu_l(w))$, which is correct. The same argument applies if $w \in V(T|\mathcal{N}[B_r])$ but is not in $V(T|\mathcal{N}[B_r])$. If $w \in V(T|\mathcal{N}[B_l]) \cap V(T|\mathcal{N}[B_r])$, we have inserted both leaves under $\mu_l(w)$ from $T_l^*$ and leaves under $\mu_r(w)$ from $T_r^*$. In this case, $\sigma(w) = \min(\sigma_l(\mu_l(w)), \sigma_r(\mu_r(w)))$ correctly represent the minimum distance from $w$ to such a leaf. It follows that $(T, \sigma)$ is the valued restriction of $T^*$ to $\mathcal{N}[B_i]$.

Let us now argue that $T^*$ is a $k$-leaf root of $G[V_l]$. Because $T^*|\mathcal{V}_l =_L T_l^*$, for each $u, v \in V_l$, $uv \in E(G) \iff \text{dist}_{T^*}(u, v) \leq k$. Similarly, $T^*|\mathcal{V}_r =_L T_r^*$ implies that for each $u, v \in V_r, uv \in E(G) \iff \text{dist}_{T^*}(u, v) \leq k$.

Now, consider $u \in \mathcal{N}[B_i]$ and $v \in \mathcal{N}[B_i]$. The fact that $uv \in E(G) \iff \text{dist}_{T^*}(u, v) \leq k$ is explicitly stated in the recurrence, using the fact that $T^*|\mathcal{N}[B_i] = T$.

Next, consider $u \in \mathcal{N}[B_i]$ and $v \in V_r \setminus \mathcal{N}[B_r]$. Then $uv \notin E(G)$, by the properties of tree decompositions. Let $w$ be the lowest ancestor of $v$ in $T^*$ such that $w \in V(T|\mathcal{N}[B_r])$ (which must exist, by the construction of $T^*$, since the subtree containing $v$ was appended under a node of $T|\mathcal{N}[B_i]$ at some point). In $T^*$, the path from $u$ to $v$ must pass through $w$. We have $\text{dist}_{T^*}(u, v) = \text{dist}_{T_l}(u, w) + \text{dist}_{T_r}(w, v) \geq \text{dist}_{T}(u, w) + \sigma_r(w)$, which is strictly greater than $k$, by the properties of the recurrence. The symmetric argument holds for $u \in \mathcal{N}[B_r]$ and $v \in V_1 \setminus \mathcal{N}[B_i]$.
Finally, consider \(u \in V_I \setminus \mathcal{N}[B_I]\) and \(v \in V_r \setminus \mathcal{N}[B_I]\). Then \(uv \notin E(G)\). Let \(w_u\) (respectively, \(w_v\)) be the lowest ancestor of \(u\) (respectively, \(v\)) in \(T^*\) such that \(w_u \in V(T \setminus \mathcal{N}[B_I])\) (respectively, \(w_v \in V(T \setminus \mathcal{N}[B_I])\)). Then \(dist_T(u, v) = dist_T(u, w_u) + dist_T(w_u, w_v) + dist_T(w_v, v) \geq \sigma_T(w_u) + dist_T(w_u, w_v) + \sigma_T(w_v)\), which is strictly greater than \(k\), by the last property of the recurrence. We have handled every possible pair of leaves and deduce that \(T^*\) is indeed a \(k\)-leaf root of \(G[V_I]\). This concludes the proof. \(\square\)

We conclude this section with the proof of the main lemma.

**Proof of Lemma 4.1.** We first need to show that the preceding dynamic programming procedure can be used to enumerate \(T = \{T_1, \ldots, T_l\}\). Recall that \(T_i \in T\) if and only if there is a \(k\)-leaf root \(T^*\) of \(G\) such that the root of \(T^*\) is the parent of \(z\), and \(T_i = (T_i, \sigma_i)\) is the valued restriction of \(T^*\) to \(N[z]\). Let \(B_z = \{z\}\) be the root of the tree decomposition from above.

Let \(T_i = (T_i, \sigma_i) \in T\) and let \(T^*\) be a \(k\)-leaf root of \(G\) such that \(r(T_i) = r(T^*)\) and such that \(T_i\) is the valued restriction of \(T^*\) to \(N[z]\). Then \(L(T_i) = N[z] = N[B_z] = N[B_z] \cap V_z\) (since \(V_z = V(G)\)). Moreover, \((T_i, \sigma_i)\) is the valued restriction of \(T^*\) to \(T_i = (T_i, \sigma_i)\), \(N[z] = N[B_z] \cap V_z\), and \(r(T_i) = r(T^*)\) since both roots must be the parent of \(z\). Thus, \((T_i, \sigma_i)\) is valid for \(B_z\), and it follows from Lemma 4.5 that \((T_i, \sigma_i) \in Q[B_z]\).

Now let \((T, \sigma) \in Q[B_z]\) such that \(r(T)\) is the parent of \(z\). Then by Lemma 4.5, \((T, \sigma)\) is valid for \(B_z\) and there is a \(k\)-leaf root \(T^*\) of \(G[V_z] = G\) such that \((T, \sigma)\) is the valued restriction of \(T^*\) to \(N[B_z] \cap V_z = N_G[z]\). Moreover, \((T^*) = r(T)\) is the parent of \(z\). Therefore, \((T, \sigma)\) must belong to \(T\).

We have thus shown that by enumerating all the valued trees in the computed \(Q[B_z]\) and keeping only those whose roots are the parent of \(z\), we reconstruct exactly \(T\).

Let us discuss the complexity. Note that \(B\) has \(O(n)\) nodes. For each bag \(B_I \in V(B)\), by Lemma 4.3, we must enumerate \(O(f(k))\) possible valued trees and test each of them for membership in \(Q[B_I]\), where \(f(k) = d^{4k}d^{4k} \cdot (k+2)^d k\). Join nodes take the longest to test, since they require to check every combination of valued trees in \(Q[B_I]\) and \(Q[B_{r_j}]\), which amounts to \(f(k)^2\) tests. It is not hard to see that for each combination, checking whether the recurrence holds can be done in time \(O(d^{4k} \cdot d^{4k})\) (the longest condition to check is to test each \(w_I, w_r\) pairs). Therefore, each \(B_{r_j}\) can be computed in time \(O(d^{4k} \cdot f(k)^2)\). Since there are \(O(n)\) such \(B_I\) bags, the complexity is \(O(n \cdot d^{4k} \cdot f(k)^3)\). \(\square\)

## 5 Putting It All Together

The results accumulated previously lead to an immediate algorithm. First, we check whether \(G\) admits a \(k\)-leaf root of arity at most \(d\), where \(d = 3|S(k, 3k)|^2|S(k, 3k)|\). This can only happen if \(G\) has maximum degree at most \(d^k\), in which case we can use the algorithm of Eppstein and Havvai [19] (or even our algorithm from Section 4 would work). If there is no such \(k\)-leaf root but that \(G\) is a \(k\)-leaf power, by Lemma 3.7, we must be able to find a homogeneous similar structure \(S\) of size \(3|S(k, 3k)|\).

To find \(S\), we begin by searching for \(C\). We brute-force every \(3|S(k, 3k)|\) disjoint subsets of at most \(d^k\) vertices from \(G\), which is the only reason our algorithm takes time \(O(n^f(k))\) instead of \(O(f(k)n^c)\). Assuming a suitable \(C\) has been identified, we look at the connected components obtained after removing the \(C_i\)’s. At this point, it is easy to verify that the properties of similar structures hold and to find \(z\) and the \(Y_i\)’s. As for the layering functions, we brute-force them all, but since the size of the \(C_i\)’s is bounded, this adds little complexity compared to the gargantuan time taken to enumerate the possible \(C_i\)’s. Once a suitable set of layering functions is found, we use the algorithm from Section 4 to compute all the \(\text{accept}(S, C_i)\) sets, and it remains to check that they are equal. If so, we have found a redundant substructure of \(G\). By Theorem 3.8, we may remove \(C_i \cup Y_i\) from \(G\) to obtain an equivalent instance. We then repeat with \(G - (C_i \cup Y_i)\). The
algorithm ends when it either reaches a graph of maximum degree at most \(d^k\), which is “easy” to verify, or when no homogeneous similar structure can be found. In the latter case, we know by Lemma 3.7 that \(G\) cannot be a \(k\)-leaf power.

**ALGORITHM 2:** Deciding if a graph is a \(k\)-leaf power.

1. Function isLeafPower\((G, k)\)
2. if \(G\) is disconnected then
3. Call isLeafPower\((G[C], k)\) for each connected component \(C\) of \(G\);
4. return true if each call returned true, and return false otherwise;
5. \(d \leftarrow 3|S(k, 3k)|2^{|S(k, 3k)|}\);
6. if \(G\) has maximum degree at most \(d^k\) then
7. Check if \(G\) is a \(k\)-leaf power and return the result;
8. foreach collection \(C = \{C_1, \ldots, C_l\}\) of disjoint subsets of \(V(G)\), with \(l = 3|S(k, 3k)|\) and with each \(|C_i| \leq d^k\) do
9. \(G' = G - \bigcup_{i \in [l]} C_i\);
10. Let \(X = \{X_1, \ldots, X_l\}\) be the connected components of \(G'\);
11. Let \(z \in V(G')\) such that \(\bigcup_{i \in [l]} C_i \subseteq N_G(z)\);
12. if \(z\) does not exist then
13. continue to the next \(C\);
14. Let \(X_z \in X\) such that \(z \in X_z\);
15. if some \(X_j \in X \setminus \{X_z\}\) has neighbors in two distinct \(C_i, C_j\) then
16. continue to the next \(C\);
17. For \(i \in [l]\), let \(Y_i\) be the union of every \(X_j \in X \setminus X_z\) such that \(N_G(X_j) \subseteq C_i\);
18. if \(\exists i \in [l], G[C_i \cup Y_i \cup \{z\}]\) has maximum degree above \(d^k\) then
19. continue to the next \(C\);
20. foreach set of layering functions \(L = \{l_1, \ldots, l_l\}\) do
21. if \(S = (C, Y = \{Y_1, \ldots, Y_d\}, z, L)\) is a similar structure then
22. foreach \(i \in [l]\) do
23. Compute accept\((S, C_i)\);
24. end
25. if all the accept\((S, C_i)\) are equal and non-empty then
26. return isLeafPower\((G - (C_1 \cup Y_1), k)\);
27. end
28. end
29. return "Not a \(k\)-leaf power";
30. end

**Theorem 5.1.** Let \(k \geq 2\) be a fixed integer. Then Algorithm 2 correctly decides whether a graph \(G\) is a \(k\)-leaf power, and runs in time \(O(n(d^{k+1}|S(k, 3k)|+6))\), where \(d = 3|S(k, 3k)| \cdot 2^{|S(k, 3k)|}\).

**Proof.** We argue correctness and complexity separately.

**Correctness.** If \(G\) is disconnected, we have already argued that we can handle each connected component separately. So suppose that \(G\) is connected. Assume that \(G\) admits a \(k\)-leaf root of arity at most \(d\). Then it will be found on line 7, by Lemma 2.2. Otherwise, if \(G\) is a \(k\)-leaf power, all its \(k\)-leaf roots have arity at least \(d + 1\). By Lemma 3.7, \(G\) admits a homogeneous similar structure \(S = (C, Y, z, L)\) with \(|C| = 3|S(k, 3k)|\), with each \(|C_i| \leq d^k\) and \(G[C_i \cup Y_i \cup \{z\}]\) having maximum degree \(d^k\) or less.
We show that the algorithm finds such a structure, if one exists. By brute-force, the main for loop will find a $C$ that belongs to a desired homogeneous similar structure $S = (C, \mathcal{Y}, z, \mathcal{L})$. The $z$ vertex described on line 11 exists, by Property 2 of similar structures. By Property 3, only the connected component $X_z$ of $G'$ that contains $z$ can have neighbors in more than one $C_i$, and all the others have neighbors in exactly one $C_i$. Combined with the fact that $G$ is connected, lines 14 through 17 correctly build the $Y_i$ subsets. Moreover, by the properties described in Lemma 3.7, each $G[C_i \cup Y_i \cup \{z\}]$ must have maximum degree at most $d^k$, and thus we will not enter the if’ on line 18. After that, since we brute-force every possible set of layering functions, we will eventually find the correct $\mathcal{L}$ for $S$. We then explicitly check whether $S$ is a similar structure and compute all the accept sets to verify homogeneity. Assuming that such a $S$ exists, it follows that it will be found, and that we will eventually reach line 26.

We can also argue the converse—that is, when line 26 is reached, $S$ is homogeneous and satisfies all the requirements of Lemma 3.7. When this line is reached, $S$ is a similar structure (this is checked explicitly) and is homogeneous since we compute every accept set. We have $|C| = 3|S(k, 3k)|$ and each $|C_i| \leq d^k$, since this is what we enumerate. Moreover, it is checked that each $G[C_i \cup Y_i \cup \{z\}]$ has degree bounded by $d^k$. Therefore, when line 26 is reached, $S$ meets all the requirements of Lemma 3.7. We can thus apply Theorem 3.8 and state that $G$ is a $k$-leaf power if and only if $G - (C_i \cup Y_i)$ is a $k$-leaf power. Thus, the recursive call on line 26 is correct.

If the algorithm never reaches line 26, then by the preceding, $G$ does not admit an homogeneous structure with all the desired properties. By contraposition of Lemma 3.7, $G$ cannot be a $k$-leaf power. This proves the correctness of the algorithm.

**Complexity.** We consider the time required to handle one recursion. We can handle the case where $G$ has maximum degree at most $d^k$ in time $O(n(d^k k)^d)$ for some constant $c$, by Lemma 2.2. Otherwise, the enumeration of the possible $C$’s requires choosing $l = 3|S(k, 3k)|$ subsets of $V(G)$ of size at most $d^k$. We can asymptotically bound the number of $C$’s to enumerate by

$$
\left(\sum_{i=1}^{d^k} \binom{n}{i}\right)^{3|S(k, 3k)|} \leq \left(\sum_{i=1}^{d^k} n^i\right)^{3|S(k, 3k)|} \leq \left(n^{d^k + 1}\right)^{3|S(k, 3k)|}
$$

(for large enough $n$). For each such $C$, the construction and verifications for $G', X, z$ and the $Y_i$’s can be done in time $O(n + |E(G)|) = O(n^2)$, until we must enumerate every set of layering functions. Such a $\mathcal{L}$ must assign each vertex in $C$ an integer between 0 and $k$. The total number of vertices in $C$ is at most $3|S(k, 3k)| \cdot d^k$, and so the number of layering functions is at most $(k + 1)^{3|S(k, 3k)|} \cdot d^k$.

Then we must check whether $S$ is truly a similar structure. At this point, we must only check that $\mathcal{L}$ satisfies all the requirements of similar structures, which can be done in time $O(n^3)$, since if suffices to compare pairs of vertices of $C$ and their neighborhoods.

One can see that for a $C_i \in C$, we can compute accept($S, C_i$) in time $O(|S(k, 3k)| \cdot n^2 d^k f(k)^3)$. To see this, recall that by using Lemma 4.1 we can enumerate every valued restriction to $C_i \cup \{z\}$ of $k$-leaf roots for $G[C_i \cup \{z\} \cup Y_i]$ in time $n d^k f(k)^3$. We need to compute the signature of each of those valued trees. Such a signature can be computed by traversing each node of the valued tree in post-order. Each node requires filling a vector with at most $|S(k, 3k)|$ entries, and so computing the signature of one $k$-leaf root takes time $O(|S(k, 3k)| \cdot n)$. We multiply this with the complexity from Lemma 4.1 to get the time needed to compute the accept($S, C_i$) set. Since there are $3|S(k, 3k)| C_i$’s to consider, computing every accept set takes time $O(|S(k, 3k)|^2 \cdot n^2 d^k f(k)^3)$. Verifying that all accept sets all equal does not add complexity to this.

Finally, we note that at each recursion, $G$ becomes smaller since $C_i$ is non-empty, so this whole procedure is repeated at most $n$ times.
Let us mention that the complexity of the case of maximum degree at most $d^k$ is dominated by the main loop, so we may omit it. To sum up, we have an asymptotic complexity of

$$n \cdot (n^{d^k+1})^3 |S(k,3k)| \cdot (k + 1)^3 |S(k,3k)|^d \cdot n^3 \cdot |S(k,3k)|^2 \cdot n^2 d^k f(k)^3,$$

where $d$, $f(k)$, and $|S(k,3k)|$ depend only on $k$. Assuming that $k \in O(1)$, this amounts to

$$O \left( n^{(d^k+1)3 |S(k,3k)|+6} \right).$$

\hfill \Box

6 CONCLUSION

Although this work answers a long-standing open question, there is still much to do on the topic of leaf powers and $k$-leaf powers:

- **Is recognizing $k$-leaf powers FPT in $k$? In other words, can it be done in time $O(f(k)n^c)$ for some function $f$ and some constant $c$?**

  Using the techniques of this work would require finding an homogeneous similar structure in FPT time, thereby avoiding brute-force enumeration. Although this appears difficult, it is possible that such structures have graph-theoretical properties that can be exploited for fast identification. For instance, we have not used the fact that $k$-leaf powers are strongly chordal, which may help finding similar structures. Moreover, $k$-leaf powers have bounded clique-width, and perhaps a clique-tree expression or an MSO$_1$ expression for similar structures could help finding them more efficiently.

  It is also worth mentioning that Fomin et al. [21, Chapter 16] provide a solution for a similar problem. For the purpose of kernelization, an algorithm for finding a $t$-protrusion of a certain size in a graph $G$ is needed, where a $t$-protrusion is a vertex set $X$ such that $tw(G[X]) \leq t$ and such that the number of vertices of $X$ with neighbors outside of $X$ is at most $t$. One can see from the definitions that the $C_i \cup Y_i$ sets are $d^k$-protrusions. The first version of the authors’ algorithm finds a $t$-protrusion of a desired size in time $n^{O(t)}$, and later shows how to find them in time $O(f(t)poly(n))$. It would be worth investigating whether the techniques developed in this work could be modified to find the relevant $C_i \cup Y_i$ sets quickly. The main challenge is that we may need to explore around $n^{f(k)}$ protrusions to find the correct $C_i \cup Y_i$ sets.

- **Can $k$-leaf powers be recognized in time $O(n^{f(k)})$, where $f(k)$ is more reasonable than in this work?**

  *In particular, can a power tower function be avoided?*

  It may be possible to find a better type of signature that is more succinct or admits less possibilities, but that still preserves enough information to prove Theorem 3.8.

- **Can the techniques used here be used to recognize other graph classes to $k$-leaf powers? In particular, can leaf powers be recognized easily?**

  This appears difficult to achieve, since our techniques are heavily dependent on $k$. However, our approach might be adaptable to leaf powers whose leaf root has bounded arity. We also mention that the complexity of recognizing PCGs is still open (recall that PCGs are graphs for which there is an interval $I$ such that $uv$ is an edge if and only if the corresponding leaves have distance in $I$). Although our techniques might not be suited for this problem, they may be useful for some restricted subclasses, such as PCGs whose interval $I$ satisfies $\max(I) \leq k$, or even $|I| \leq k$.  

ACM Transactions on Algorithms, Vol. 19, No. 4, Article 35. Publication date: September 2023.
ACKNOWLEDGMENTS
The author thanks the anonymous reviewers of the SODA 2022 conference and those of the ACM Transactions on Algorithms journal for their extremely useful suggestions. The author also thanks Michael Blondin and François Ladouceur for discussions on leaf powers and related problems.

REFERENCES
[1] Benjamin Bergougnoux, Svein Høgemo, Martin Vatshelle, and Jan Arne Telle. 2022. Recognition of linear and star variants of leaf powers is in P. In Proceedings of the International Workshop on Graph-Theoretic Concepts in Computer Science. 70–83.
[2] Jean R. S. Blair and Barry Peyton. 1993. An introduction to chordal graphs and clique trees. In Graph Theory and Sparse Matrix Computation. Springer, 1–29.
[3] Hans L. Bodlaender. 2009. Kernelization: New upper and lower bound techniques. In Proceedings of the International Workshop on Parameterized and Exact Computation. 17–37.
[4] Andreas Brandstädt and Van Bang Le. 2006. Structure and linear time recognition of 3-leaf powers. Information Processing Letters 98, 4 (2006), 133–138.
[5] Andreas Brandstädt, Van Bang Le, and Dieter Rautenbach. 2009. A forbidden induced subgraph characterization of distance-hereditary 5-leaf powers. Discrete Mathematics 309, 12 (2009), 3843–3852.
[6] Andreas Brandstädt and Christian Hundt. 2008. Ptolemaic graphs and interval graphs are leaf powers. In Proceedings of the Latin American Symposium on Theoretical Informatics. 479–491.
[7] Andreas Brandstädt, Christian Hundt, Federico Mancini, and Peter Wagner. 2010. Rooted directed path graphs are leaf powers. Discrete Mathematics 310, 4 (2010), 897–910.
[8] Andreas Brandstädt, Van Bang Le, and R. Sritharan. 2008. Structure and linear-time recognition of 4-leaf powers. ACM Transactions on Algorithms 5, 1 (2008), 1–22.
[9] Andreas Brandstädt and Peter Wagner. 2007. On \((k, l)\)-leaf powers. In Proceedings of the International Symposium on Mathematical Foundations of Computer Science. 525–535.
[10] Andreas Brandstädt and Peter Wagner. 2008. On \(k\)-versus \((k + 1)\)-leaf powers. In Proceedings of the International Conference on Combinatorial Optimization and Applications. 171–179.
[11] Tiziana Calamoneri, Manuel Lafond, Angelo Monti, and Blerina Sinaimeri. 2022. On generalizations of pairwise compatibility graphs. arXiv preprint arXiv:2112.08503 (2022).
[12] Tiziana Calamoneri and Blerina Sinaimeri. 2016. Pairwise compatibility graphs: A survey. SIAM Review 58, 3 (2016), 445–460.
[13] Maw-Shang Chang and Ming-Tat Ko. 2007. The 3-Steiner root problem. In Proceedings of the International Workshop on Graph-Theoretic Concepts in Computer Science. 109–120.
[14] Maw-Shang Chang, Ming-Tat Ko, and Hsueh-I. Lu. 2006. Linear-time algorithms for tree root problems. In Proceedings of the Scandinavian Workshop on Algorithm Theory. 411–422.
[15] Zhi-Zhong Chen, Tao Jiang, and Guohui Lin. 2003. Computing phylogenetic roots with bounded degrees and errors. SIAM Journal on Computing 32, 4 (2003), 864–879.
[16] Michael Dom, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. 2004. Error compensation in leaf root problems. In Proceedings of the International Symposium on Algorithms and Computation. 389–401.
[17] Michael Dom, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. 2005. Extending the tractability border for closest leaf powers. In Proceedings of the International Workshop on Graph-Theoretic Concepts in Computer Science. 397–408.
[18] Guillaume Ducoffe. 2019. The 4-Steiner root problem. In Proceedings of the International Workshop on Graph-Theoretic Concepts in Computer Science. 14–26.
[19] David Eppstein and Elham Havvaei. 2020. Parameterized leaf power recognition via embedding into graph products. Algorithmica 82 (2020), 2337–2359.
[20] Joseph Felsenstein. 2004. Inferring Phylogenies. Vol. 2. Sinauer Associates, Sunderland, MA.
[21] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. 2019. Kernelization: Theory of Parameterized Preprocessing. Cambridge University Press.
[22] Frank Gurski and Egon Wanke. 2009. The NLC-width and clique-width for powers of graphs of bounded tree-width. Discrete Applied Mathematics 157, 4 (2009), 583–595.
[23] Lars Jaffke, O-Joung Kwon, Torstein J. F. Strømme, and Jan Arne Telle. 2019. Mim-width III. Graph powers and generalized distance domination problems. Theoretical Computer Science 796 (2019), 216–236.
[24] William Kennedy, Guohui Lin, and Guiying Yan. 2006. Strictly chordal graphs are leaf powers. Journal of Discrete Algorithms 4, 4 (2006), 511–525.
[25] Manuel Lafond. 2017. On strongly chordal graphs that are not leaf powers. In Proceedings of the International Workshop on Graph-Theoretic Concepts in Computer Science. 386–398.
[26] Ragnar Nevries and Christian Rosenke. 2016. Towards a characterization of leaf powers by clique arrangements. *Graphs and Combinatorics* 32, 5 (2016), 2053–2077.

[27] Naomi Nishimura, Prabhakar Ragde, and Dimitrios M. Thilikos. 2002. On graph powers for leaf-labeled trees. *Journal of Algorithms* 42, 1 (2002), 69–108.

[28] Hervé Philippe, Henner Brinkmann, Dennis V. Lavrov, D. Timothy J. Littlewood, Michael Manuel, Gert Wörheide, and Denis Baurain. 2011. Resolving difficult phylogenetic questions: Why more sequences are not enough. *PLoS Biology* 9, 3 (2011), e1000602.

[29] Md. Saidur Rahman and Shareef Ahmed. 2020. A survey on pairwise compatibility graphs. *AKCE International Journal of Graphs and Combinatorics* 17, 3 (2020), 788–795.

[30] Dieter Rautenbach. 2006. Some remarks about leaf roots. *Discrete Mathematics* 306, 13 (2006), 1456–1461.

Received 4 November 2022; revised 4 July 2023; accepted 2 August 2023