Extending the Tractability Border for Closest Leaf Powers*

Michael Dom Jiong Guo Falk Hüffner Rolf Niedermeier

Abstract

The NP-complete Closest 4-Leaf Power problem asks, given an undirected graph, whether it can be modified by at most ℓ edge insertions or deletions such that it becomes a 4-leaf power. Herein, a 4-leaf power is a graph that can be constructed by considering an unrooted tree—the 4-leaf root—with leaves one-to-one labeled by the graph vertices, where we connect two graph vertices by an edge iff their corresponding leaves are at distance at most 4 in the tree. Complementing and "completing" previous work on Closest 2-Leaf Power and Closest 3-Leaf Power, we show that Closest 4-Leaf Power is fixed-parameter tractable with respect to parameter ℓ. This gives one of the first and so far deepest positive algorithmic results in the field of "approximate graph power recognition"—note that for 5-leaf powers even the complexity of the exact recognition problem is open.

1 Introduction

Graph powers form a classical concept in graph theory, and the rich literature dates back to the sixties of the previous century (see [2, Sect. 10.6] and [15] for surveys). The *k-power* of an undirected graph $G = (V, E)$ is the undirected graph $G^k = (V, E')$ with $(u, v) \in E'$ iff there is a path of length at most k between u and v in G. We say G is the *k-root* of G^k. It is NP-complete to decide whether a given graph is a k-power or not [18]. By way of contrast, one can decide in $O(|V|^3)$ time whether a graph is a k-power of a tree for any fixed k [12]. In particular, it can be decided in linear time whether a graph is a square of a tree [17, 14].

In this paper we concentrate on certain practically motivated variants of tree powers. Whereas Kearney and Corneil [12] study the problem where every tree node one-to-one corresponds to a graph vertex, Nishimura, Ragde, and Thilikos [21] introduce the notion of *leaf powers* where exclusively the tree leaves stand in one-to-one correspondence to the graph vertices. In addition, Lin, Kearney, and Jiang [16], Chen, Jiang, and Lin [4], and Chen and Tsukiji [5] examine the variant of leaf powers where all inner nodes of the root tree have degree at least three. Both problems find applications in computational evolutionary biology [21, 16, 4]. The corresponding recognition problems are called k-Leaf Power [21] and k-Phylogenetic Root [16], respectively.[1] For $k \leq 4$, both problems are solvable in polynomial time [21, 16]. The complexities of both recognition problems for $k \geq 5$ are still open.

Several groups of researchers [4, 12, 16] strongly advocate the consideration of a more relaxed or "approximate" version of the graph power recognition problem: Now, look for roots whose powers are *close* to the input graphs, thus turning the focus of study to the corresponding *graph modification* problems. Kearney and Corneil [12] were the first to formulate this problem setting when introducing the Closest k-Tree Power problem. In this "error correction setting" the question is whether a given graph can be modified by adding or deleting at most ℓ edges such that the resulting graph has a k-tree root. This problem turns out to be

*Institut für Informatik, Friedrich-Schiller-Universität Jena, Ernst-Abbe-Platz 2, D-07743 Jena, Germany, {dom,guo,hueffner,niedermr}@minet.uni-jena.de. Supported by the Deutsche Forschungsgemeinschaft (DFG), Emmy Noether research group PIAF (fixed-parameter algorithms), NI 369/4.

[1] Both problems k-Leaf Power and k-Phylogenetic Root ask if a given graph is a leaf power resp. a phylogenetic power. We find it more natural to use the term *power* instead of the term *root* here, although we used the term *root* in our previous considerations concerning the case $k = 3$ [6].

NP-complete for $k \geq 2$ [12, 11, 7]. One also obtains NP-completeness for the corresponding problems Closest k-Leaf Power [13, 6] and Closest k-Phylogenetic Root [4, 24].

All nontrivial ($k \geq 2$) "approximate recognition" problems in our context turn out to be NP-complete [1, 4, 6, 7, 11, 12, 13, 23, 24]. Hence, the pressing quest is to also show positive algorithmic tractability results such as polynomial-time approximation or non-trivial (exponential-time) exact algorithms. So far, only the most simple version of Closest Leaf Power, $k = 2$, has been algorithmically attacked with somewhat satisfactory success. In this context recently intricate polynomial-time constant-factor approximation algorithms have been developed [1, 3].[2] Moreover, it is fairly easy to show that the problem is fixed-parameter tractable with respect to the parameter ℓ denoting the number of allowed edge modifications [10]. At least with respect to this fixed-parameter tractability result, the success is surely due to the fact that there is a very simple characterization by a forbidden subgraph: a graph is a 2-leaf power iff it contains no induced 3-vertex subgraph forming a path. Observe that, in this way, also the recognition problem for 2-leaf powers is solvable in linear time by just checking whether the given graph is a disjoint union of cliques. By way of contrast, the recognition problem for 3-leaf and 4-leaf powers is much harder and only intricate cubic-time algorithms are known [21]. At first sight, this lowers the hope for obtaining positive algorithmic results for Closest k-Leaf Power for $k = 3, 4$. The key idea we put forward here and in a companion paper [6] is to again develop and to employ forbidden subgraph characterizations of the respective graph classes. Unlike for 2-leaf powers, these characterizations are not so obvious. In [6], we describe a forbidden subgraph characterization for 3-leaf powers, consisting of five graphs of small size. Here, we present a forbidden subgraph characterization for 4-leaf powers—it already requires numerous forbidden subgraphs.

Let us discuss the algorithmic use of these forbidden subgraph characterizations. First, both characterizations immediately imply polynomial-time recognition algorithms for 3- and 4-leaf powers which are conceptually simpler than those in [21]. However, they are of purely theoretical interest because the running times of these straightforward algorithms are much worse than that of the known cubic-time algorithms from [21]. More important, the characterizations open up the way to the first tractability results for the harder problems Closest k-Leaf Power for $k = 3, 4$. Using the forbidden subgraphs for 3-leaf powers, in [6] we show that Closest 3-Leaf Power is fixed-parameter tractable with respect to the parameter "number ℓ of edge modifications." Due to the significantly increased combinatorial complexity of 4-leaf powers (with numerous forbidden subgraphs instead of only a handful), analogous results for Closest 4-Leaf Power remained open in [6]. We close this gap here. We show that Closest 4-Leaf Power can be solved in polynomial time for $\ell = O(\log n / \log \log n)$; that is, it is fixed-parameter tractable with respect to parameter ℓ. Moreover, the variants of Closest 4-Leaf Power where only edge insertions or only edge deletions are allowed are fixed-parameter tractable as well. On the way to our result, we develop a "compressed form" of a forbidden subgraph characterization of 4-leaf powers that has been developed—independently and by different means—by Rautenbach [22]. Since we aim at algorithmic tractability results for Closest 4-Leaf Power, we employ a "more constructive" approach.

Due to the lack of space, we omit almost all proofs.

2 Preliminaries

We consider only undirected graphs $G = (V, E)$ with $n := |V|$ and $m := |E|$. Edges are denoted as tuples (u, v). For a graph $G = (V, E)$ and $u, v \in V$, let $d_G(u, v)$ denote the length of the shortest path between u and v in G. With $E(G)$, we denote the edge set E of a graph $G = (V, E)$. We call a graph $G' = (V', E')$ an *induced subgraph* of $G = (V, E)$ and denote G' with $G[V']$ if $V' \subseteq V$ and $E' = \{(u, v) \mid u, v \in V' \text{ and } (u, v) \in E\}$. For a non-empty collection of graphs $\mathcal{G}$, a graph is said to be $\mathcal{G}$*-free* if it does not contain any graph in $\mathcal{G}$ as

[2] Note that in the various papers (partially not referring to each other) Closest 2-Leaf Power appears under various names such as Cluster Editing [23] and Correlation Clustering [1, 3].

induced subgraph. A cycle with n vertices is denoted as C_n. An edge between two vertices of a cycle that is not part of the cycle is called *chord*. An induced cycle of length at least four is called *hole*. A *chordal graph* then is a hole-free graph. For two sets A and B, $A \triangle B$ denotes the *symmetric difference* $(A \setminus B) \cup (B \setminus A)$.

Definition 1 ([21]). Consider an unrooted tree T with leaves one-to-one labeled by the elements of a set V. The *k-leaf power* of T is a graph, denoted T^k, with $T^k := (V, E)$, where $E := \{(u, v) \mid u, v \in V \text{ and } d_T(u, v) \leq k\}$.

The leaf power version k-Leaf Power (LPk) of the graph power problem is: given a graph G, is there a tree T such that $T^k = G$?

One may view the leaf power concept as a "Steiner extension" of the standard notion of tree powers [4, 16]. The more general, *approximate version* of LPk we focus on in this work, called Closest k-Leaf Power (CLPk), then reads as follows. Consider a graph $G = (V, E)$ and a nonnegative integer ℓ, is there a tree T such that T^k and G differ by at most ℓ edges, that is, $|E(T^k) \triangle E(G)| \leq \ell$? CLP$k$ is NP-complete for $k \geq 2$ [13, 6].

In this paper we also study two variations of CLPk referring to only one-sided errors: CLPk Edge Insertion only allows insertion of edges and CLPk Edge Deletion only allows deletion of edges. CLPk Edge Deletion is NP-complete for $k \geq 2$ [19, 6], and CLPk Edge Insertion is NP-complete for $k \geq 3$ but polynomial-time solvable for $k = 2$ [6].

A central technical tool within this work are critical cliques and critical clique graphs as Lin et al. [16] introduce them:

Definition 2. A *critical clique* of a graph G is a clique K where the vertices of K all have the same set of neighbors in $G \setminus K$, and K is maximal under this property. Consider a graph $G = (V, E)$. Let C be the collection of its critical cliques. Then the *critical clique graph* $\mathrm{CC}(G)$ is a graph (C, E_C) with

$$(K_i, K_j) \in E_C \iff \forall u \in K_i, v \in K_j : (u, v) \in E.$$

That is, the critical clique graph has the critical cliques as nodes, and two nodes are connected iff the corresponding critical cliques together form a larger clique.

Eventually, for technical reasons we also need the concept of a k-Steiner root.

Definition 3. Consider a graph $G = (V, E)$. An unrooted tree $T = (A \,\dot\cup\, V, E')$ is called a *k-Steiner root* of G if $E = \{(u, v) \mid u, v \in V \text{ and } d_T(u, v) \leq k\}$.

Note that if $A = \emptyset$, then a k-Steiner root simply is a k-tree root. Similarly, if A is the set of inner nodes of T, then a k-Steiner root is the same as a k-leaf root. This means that the set of graphs that have k-Steiner roots is a superset of the set of graphs that have k-tree roots or k-leaf roots. The following lemma is easy to show (a similar statement was already made by Lin et al. [16]).

Lemma 1. *A graph G has a k-leaf root iff $\mathrm{CC}(G)$ has a $(k-2)$-Steiner root.*

We show that CLP4 and both its edge insertion and edge deletion variant are *fixed-parameter tractable* (FPT) with respect to parameter ℓ. That is, we show that CLP4 can be solved in $f(\ell) \cdot n^{O(1)}$ time, where f is a computable function only depending on ℓ, and n denotes the number of vertices of the input graph. Two recent surveys on fixed-parameter tractability can be found in [9, 20].

3 Forbidden Subgraph Characterization of 4-Leaf Powers

In this section we give a characterization of 4-leaf powers using a set of eight forbidden induced subgraphs for critical clique graphs of 4-leaf powers. This set can be extended to a larger set of forbidden subgraphs for the 4-leaf powers themselves by a simple iterative algorithm.

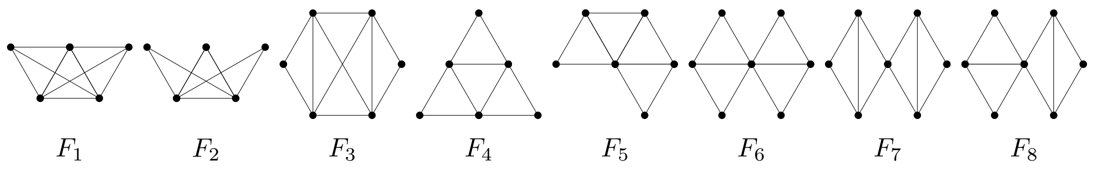

Figure 1: The "compressed" set of forbidden induced subgraphs: All forbidden subgraphs can be derived from these eight graphs.

Independently and by different proof techniques, Rautenbach [22] achieves the same results. Our approach, however, is tailored towards the algorithmic treatment following in the next section. The eight forbidden subgraphs for critical clique graphs of 4-leaf powers are shown in Fig. 1. Let $\mathcal{F} := \{F_1, F_2, \ldots, F_8\}$ as given there.

The main result of this section is as follows:

Theorem 1. *A graph G has a 4-leaf root iff G is chordal and its critical clique graph* $\mathrm{CC}(G)$ *is $\mathcal{F}$-free.*

The forbidden subgraph characterization of Theorem 1 refers to critical clique graphs. However, it directly implies a somewhat more extensive forbidden subgraph characterization for the original graphs.

Corollary 1. *All graphs that are 4-leaf powers are chordal and can be characterized by a finite set of forbidden subgraphs.*

It is relatively easy to see that graphs having a 4-leaf root must be chordal, and that a critical clique graph $\mathrm{CC}(G)$ containing a graph in $\mathcal{F}$ (Fig. 1) as an induced subgraph has no 2-Steiner root (and, according to Lemma 1, the graph G has no 4-leaf root). The reverse direction of Theorem 1 is technically far more difficult. We show constructively that every $\mathcal{F}$-free and chordal critical clique graph has indeed a 2-Steiner root by using Algorithm SRG (Fig. 2). This algorithm extends the method by Lin et. al. [16] for constructing 2-Steiner roots: While their algorithm only computes an output graph if the input graph has a 2-Steiner root and says "no" otherwise, our Algorithm SRG also generates an output graph with some guaranteed properties in case of inputs that are $(\mathcal{F}, C_4, C_5)$-free but nonchordal graphs. This will be of use for our fixed-parameter algorithms in Sect. 4.

For a given critical clique graph $\mathrm{CC}(G) = (C, E_C)$, Algorithm SRG constructs a *pseudo Steiner root graph* $S = (V', E')$ with $V' := A \dot\cup B$, where $B := \{b_c \mid c \in C\}$. The nodes in A and B are called *Steiner* and *non-Steiner* nodes, respectively. Each non-Steiner node one-to-one corresponds to a node in $\mathrm{CC}(G)$, whereas Steiner nodes do not have to correspond to nodes in $\mathrm{CC}(G)$. If $\mathrm{CC}(G)$ is $\mathcal{F}$-free and chordal, the graph S is a 2-Steiner root of $\mathrm{CC}(G)$. (The term "pseudo Steiner root graph" expresses that if the input graph is $(\mathcal{F}, C_4, C_5)$-free but nonchordal, the output S has some, but not all properties of a 2-Steiner root.)

The idea of the algorithm is to consider every maximal clique of the input graph $\mathrm{CC}(G)$ and to connect the corresponding nodes in the output graph to form a star. More specifically, if a maximal clique K in $\mathrm{CC}(G)$ has an edge e in common with another maximal clique, one of the two endpoints of e is selected and the corresponding node in the output graph is connected by edges with the other nodes corresponding to K. If otherwise K has no edge in common with another maximal clique, a Steiner node s_K is inserted into the output graph, and every node corresponding to a node of K is connected by an edge with s_K (see Fig. 3 for an example).

We can show that Algorithm SRG fulfills the following five claims, which implies that the constructed pseudo Steiner root graph of an $\mathcal{F}$-free, chordal critical clique graph $\mathrm{CC}(G)$ actually is a 2-Steiner root of $\mathrm{CC}(G)$, which together with Lemma 1 proves the missing direction of Theorem 1. Note that the first four claims do not require chordality of the input graph. We will make use of this fact in Sect. 4 when we have to modify a critical clique graph to make it chordal. The five claims are:

```
    SRG(CC($G$) = ($C, E_C$))
    **Input:**  ($\mathcal{F}, C_4, C_5$)-free critical clique graph CC($G$)
    **Output:** Pseudo Steiner root graph of CC($G$)
1   $S \leftarrow (\{b_c \mid c \in C\}, \emptyset)$
2   $L \leftarrow$ list of all maximal cliques of CC($G$)
3   while there is a $K$ in $L$ which shares edges $(c_1, c_2)$ and $(c_1, c_3)$ with two
            other maximal cliques $K'$ and $K''$ in CC($G$):
4       Delete $K$ from $L$
5       for $c \in K, c \neq c_1$:
6           Insert an edge between $b_{c_1}$ and $b_c$
7   while there is a $K$ in $L$ which shares only one edge $(c_1, c_2)$ with only one
            other maximal clique $K'$ in CC($G$):
8       Delete $K$ from $L$
9       if $K'$ is in $L$:
10          for $c \in K, c \neq c_1$:
11              Insert an edge between $b_{c_1}$ and $b_c$
12      else:
13          $c' \leftarrow$ a node in $K' \setminus K$
14          if there is an edge $(b_{c_1}, b_{c'})$ in $S$:
15              for $c \in K, c \neq c_2$:
16                  Insert an edge between $b_{c_2}$ and $b_c$
17          else:
18              for $c \in K, c \neq c_1$:
19                  Insert an edge between $b_{c_1}$ and $b_c$
20  while there is a $K$ in $L$:
21      Delete $K$ from $L$
22      Add a new node $s_K$ into $S$
23      for $c \in K$:
24          Insert an edge between $s_K$ and $b_c$
25  while there are at least two connected components $S_1$ and $S_2$ in $S$:
26      Add two edge-connected Steiner nodes $s_1$ and $s_2$ to $S$ and connect by an edge $s_1$ to
        an arbitrary node in $S_1$ and $s_2$ to an arbitrary node in $S_2$
```

Figure 2: Algorithm to construct the pseudo Steiner root graph S of a critical clique graph CC(G)

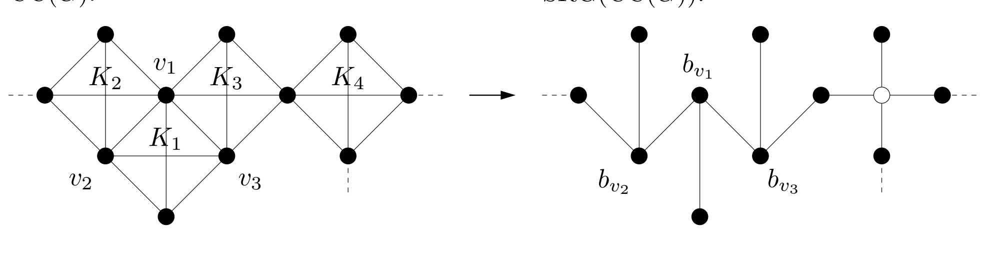

Figure 3: Example. A subgraph of a critical clique graph CC(G) and the pseudo Steiner root graph computed for this part of CC(G). Algorithm SRG first considers the maximal clique K_1 with $c_1 = v_1$ and inserts edges between b_{v_1} and the other nodes corresponding to K_1. Thereafter, the cliques K_2 and K_3 are considered. When considering K_4, Algorithm SRG inserts a Steiner node (drawn white).

1. There are at most $2\cdot|E|$ maximal cliques in an $(\mathcal{F}, C_4, C_5)$-free critical clique graph $\mathrm{CC}(G)$.

2. Every maximal clique K of an $(\mathcal{F}, C_4, C_5)$-free critical clique graph $\mathrm{CC}(G)$ is considered exactly once by the algorithm, and for every node pair u, v in K which is considered by the algorithm, a path of length at most two is generated between the corresponding nodes in the output graph.

3. For an $(\mathcal{F}, C_4, C_5)$-free critical clique graph $\mathrm{CC}(G) = (C, E_C)$ the algorithm outputs a graph with the following property: If two nodes $u, v \in C$ are not adjacent, the distance between the nodes corresponding to u and v in the output graph is at least three.

4. For an $(\mathcal{F}, C_4, C_5)$-free critical clique graph $\mathrm{CC}(G)$ the output graph of the algorithm contains no cycle of length at most seven.

5. For a chordal and $\mathcal{F}$-free critical clique graph $\mathrm{CC}(G)$ the output graph of the algorithm is a tree.

Note that Claims 2 and 3 together show the desired "distance property" of Definition 3 for Steiner roots.

4 Fixed-Parameter Tractability of CLP4

In this section we show the fixed-parameter tractability of CLP4 Edge Deletion, CLP4 Edge Insertion, and CLP4 with respect to the parameter "number of edge editing operations" ℓ. The basic approach somewhat resembles our previous work for CLP3 [6]; however, for the case of CLP4 Edge Deletion new, more intricate methods are necessary. Therefore, we focus on the CLP4 Edge Deletion case in this section.

Note that graphs that have 3-leaf roots have a characterization similar to that of Theorem 1: they are graphs that are chordal and contain none of the induced subgraphs "bull," "dart," and "gem" [6]. Therefore, the basic idea for CLP3 Edge Deletion as well as for CLP4 Edge Deletion is to use the forbidden subgraph characterization in a search tree algorithm: find a forbidden subgraph, and recursively branch into several cases according to the possible edge deletions that destroy the forbidden subgraph. If we can bound the number of branching cases by a function depending only on ℓ, we obtain a run time that proves fixed-parameter tractability.

Since the forbidden subgraph characterization from Theorem 1 for the critical clique graph $\mathrm{CC}(G)$ is much simpler than the implied characterization for G (Corollary 1), we would like to apply modifications directly on $\mathrm{CC}(G)$. This is possible by the following lemma, which is a straightforward extension of Lemma 6 in [6].

Lemma 2. *For a graph G, there is always an optimal solution for CLP4 that is represented by edge editing operations on $\mathrm{CC}(G)$. That is, one can find an optimal solution that does not delete any edges within a critical clique; furthermore, in this optimal solution, between two critical cliques either all or no edges are inserted or deleted.*

Now, working with $\mathrm{CC}(G)$ instead of G has two implications: First, a deletion of an edge e in $\mathrm{CC}(G)$ can represent several deletions in G. Consider an edge e in $\mathrm{CC}(G)$ between two nodes that represent critical cliques of size c_1 and c_2. Deleting e implies deleting all $c_1 \cdot c_2$ edges between the vertices of the critical cliques in G. Therefore, we give the edge e the weight $c_1 \cdot c_2$. Note that this means that an edge modification on $\mathrm{CC}(G)$ can decrease the parameter ℓ in the bounded search tree algorithm by more than one. Second, if two adjacent nodes in $\mathrm{CC}(G)$ obtain identical neighborhood after deleting edges in $\mathrm{CC}(G)$, then $\mathrm{CC}(G)$ needs to be updated, since each node in $\mathrm{CC}(G)$ has to represent a critical clique in G. In this situation a *merge* operation is needed, which replaces these nodes in $\mathrm{CC}(G)$ by a new node with the same neighborhood as the original nodes. In the following, we assume that after each modification of $\mathrm{CC}(G)$, all pairs of nodes in $\mathrm{CC}(G)$ are checked as to whether a merge operation between them is required, which can be done in $O(|C| \cdot |E_C|)$ time.

The main obstacle in obtaining an FPT algorithm for both CLP3 Edge Deletion and CLP4 Edge Deletion is that the holes in $\mathrm{CC}(G)$ can have arbitrary length, and, therefore, one cannot simply find some hole and branch for each edge of the hole that is to be deleted—the number of branching cases would not be a function depending on ℓ. For CLP3 Edge Deletion, the key observation was that a critical clique graph $\mathrm{CC}(G)$ containing neither a bull nor a dart nor a gem nor a C_4 contains no triangles and, therefore, no hole in $\mathrm{CC}(G)$ can be destroyed by merging its critical cliques without deleting at least one edge of the hole. In this way, a minimum weight set of edges to be deleted to make $\mathrm{CC}(G)$ chordal could be obtained in polynomial time by searching for a maximum weight spanning tree. Unfortunately, the observation is not valid for an $\mathcal{F}$-free (Fig. 1) $\mathrm{CC}(G)$ as we obtain it for CLP4 after deleting the forbidden subgraphs. Thus, the main technical contribution of this section is to show how to circumvent these difficulties by new, more sophisticated techniques than that required for CLP3 Edge Deletion.

The idea is to examine the output $\mathrm{SRG}(\mathrm{CC}(G))$ of Algorithm SRG (Fig. 2) for the critical clique graph $\mathrm{CC}(G)$. If it is a tree, we are done. Otherwise, the output is a pseudo Steiner root graph S that contains a cycle which corresponds to a hole in $\mathrm{CC}(G)$. By repeatedly deleting degree-1 nodes and contracting consecutive degree-2 nodes in S we get a graph S' in which every edge is part of at least one cycle and in which there is no path that consists of three or more consecutive degree-2 nodes. By finding the shortest cycle in this reduced graph S', we can obtain an "FPT hole" in $\mathrm{CC}(G)$, that is, a hole for which we can bound the possibilities to delete edges to get rid of the hole in an optimal way by a function only depending on ℓ.

For the pseudocode of this algorithm (Fig. 4), we introduce some notation for the relation between a graph and its pseudo Steiner Root graph.

Definition 4. Consider a critical clique graph $\mathrm{CC}(G) = (C, E_C)$ and a pseudo Steiner root graph $S = (V_S, E_S)$ constructed by Algorithm SRG for $\mathrm{CC}(G)$. For $v \in C$ we use $S(v)$ to denote the node from V_S that corresponds to v, and for $v_S \in V_S$, we define $S^{-1}(v_S)$ as the node in C corresponding to v_S if v_S is a non-Steiner node, or $\perp$ if v_S is a Steiner node. We extend this notation to sets: for $C' \subseteq C$, $S(C') := \{S(v) \mid v \in C'\}$, and for $V' \subseteq V_S$, $S^{-1}(V') := \{S^{-1}(v) \mid v \in V'\}$.

To show the correctness of Algorithm CLP4Del-Branch in Fig. 4, we first need the following lemma which is not very difficult to see using the Claims 1–4 in Sect. 3.

Lemma 3. *Consider a set of edges H as constructed in line 8 of Algorithm* CLP4Del-Branch *(Fig. 4). Then H induces at least one hole in* $\mathrm{CC}(G)$.

To define the branching set D in line 9 of Algorithm CLP4Del-Branch, we need some notation.

Definition 5. A *big node* is a node of a pseudo Steiner root graph S that is not deleted by the data reduction in line 5 of Algorithm CLP4Del-Branch and that has degree at least 3 in the constructed S'.

For a pseudo Steiner root graph S with two big nodes v_i and v_j connected by a path P containing only non-big nodes, the *border set* B_i^P is the set of non-Steiner nodes on P which have distance at most 2 to v_i. With P^+ we denote the maximal set of nodes in S such that P^+ contains the nodes of P and such that P^+ induces a connected component in $S \setminus \{v_i, v_j\}$.

Let $\textsc{MinCut}(G, V_1, V_2)$ be a minimum weight set of edges in $G = (V, E)$ that disconnects all vertices in $V_1 \subseteq V$ from those in $V_2 \subseteq V$.

The main observation that helps to bound the number of branching cases and, hence, leads to our FPT-algorithm is that for a cycle Q in S the number of branching cases is independent of the lengths of the paths in Q between the big nodes: If P is a path in Q between two big nodes v_i and v_j and if the two node sets $S^{-1}(B_i^P)$, $S^{-1}(B_j^P)$ have to be disconnected in $\mathrm{CC}(G)[S^{-1}(P^+)]$ then it is always optimal to take an edge set with a minimum weight whose removal disconnects the two sets. Such an edge set can be found in polynomial time; one can show that the stucture of an $(\mathcal{F}, C_4, C_5)$-free critical clique graph guarantees that

```
   CLP4Del-Branch($G, \ell$)
   Input:  A Graph $G = (V, E)$ and an integer $\ell$
   Output: A set of at most $\ell$ edges in $G$ whose removal makes $G$ a 4-leaf power,
           or nil if no such set exists
1  if $\ell < 0$: return nil
2  if CC($G$) contains a forbidden subgraph: branch accordingly
3  $S \leftarrow$ SRG(CC($G$))
4  if $S$ is a tree: return $\emptyset$
5  $S' \leftarrow S$ with degree-1 and degree-2 nodes reduced
6  $Q' \leftarrow$ shortest cycle in $S'$
7  $Q \leftarrow$ cycle corresponding to $Q'$ in $S$
8  $H \leftarrow S^{-1}(Q) \setminus \{\perp\}$
9  Determine a set $D$ of edge sets in CC($G$)[$H$] such that
   at least one $d \in D$ is a subset of an optimal solution
10 for $d \in D$:
11     $X \leftarrow$ CLP4Del-Branch (CC-Del($G, d$), $\ell-$CC-Weight($G, d$))
12     if $X \neq$ nil: return $X \cup d$
13 return nil
```

Figure 4: Algorithm for CLP4 Edge Deletion. The subroutine CC-Del(G, d) takes a graph G and a set d of edges in CC(G) as input. For every edge $(K_1, K_2) \in d$, all edges from G that have one endpoint in K_1 and the other endpoint in K_2 are deleted by CC-Del(G, d). CC-Weight(G, d) returns the sum of the weights of the edges in d.

no new forbidden subgraphs are generated by this step. The following lemma states this observation more precisely.

Lemma 4. *Consider a cycle Q in a pseudo Steiner root graph S as constructed by Algorithm* CLP4Del-Branch *(Fig. 4) in line 7. Let $v_0, \ldots, v_{j-1}$ be the big nodes in Q, ordered by their appearance in Q, and for every node v_i with $0 \leq i < j$ let P_i be the path in Q between v_i and $v_{(i+1) \bmod j}$.*

Then it is correct to choose the branching set D as follows: Either delete an edge (u, v) such that

$$u, v \in S^{-1}\big(\{v_i\} \cup B_i^{P_{(i-1) \bmod j}} \cup B_i^{P_i}\big) \setminus \{\perp\} \text{ with } 0 \leq i < j$$

(that is, delete an edge between nodes near a big node) or delete a set of edges

$$\textsc{MinCut}\big(\mathrm{CC}(G)[S^{-1}(P_i^+) \setminus \{\perp\}],\ S^{-1}(B_i^{P_i}),\ S^{-1}(B_{(i+1) \bmod j}^{P_i})\big) \text{ with } 0 \leq i < j$$

(that is, delete a minimum weight set of edges such that all paths in the subgraph induced by $S^{-1}(P_i^+) \setminus \{\perp\}$ between nodes in $S^{-1}(B_i^{P_i})$ and nodes in $S^{-1}(B_{(i+1) \bmod j}^{P_i})$ are destroyed).

Fig. 5 shows an illustration for Lemma 4. It remains to show the complexity of CLP4Del-Branch (Fig. 4). It is clear that S' can be constructed in $O(|S|)$ time and S' has a cycle iff S has a cycle. A well-known result by Erdős and Pósa [8] states that any graph with minimum vertex degree at least 3 has a cycle of length at most $2 \log n + 1$, where n denotes the number of graph nodes. Using this result we can give an upper bound on the size of the shortest cycle in S' and show the following lemma:

Lemma 5. *When choosing D in Algorithm* CLP4Del-Branch *in line 9 as described by Lemma 4, we can bound its size by $|D| \leq 48 \cdot O(\log(|V|)) + 24$.*

Using Lemma 5, we arrive at the main theorem of this section.

Theorem 2. CLP4 Edge Deletion *with ℓ edge deletions allowed is fixed-parameter tractable with respect to ℓ.*

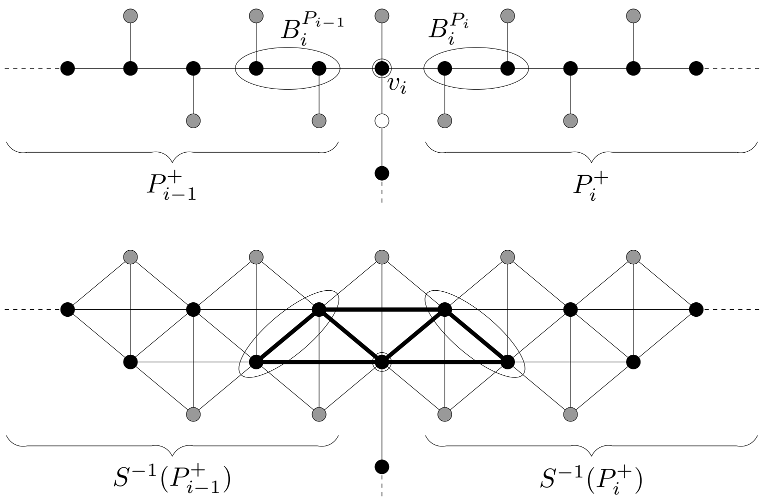

Figure 5: Illustration of Lemma 4. The upper picture shows a part of pseudo Steiner root graph S. The encircled node represents a big node; black nodes represent nodes that are part of a cycle in S. The grey nodes are not part of any cycle in S and, therefore, deleted by the data reduction in line 5 of Algorithm CLP4Del-Branch. In this example, all black and grey nodes are non-Steiner nodes; the only Steiner node is the white node. The lower picture shows the corresponding part of CC(G). The edges drawn with bold lines are those between nodes u, v with $u, v \in S^{-1}\big(\{v_i\} \cup B_i^{P_{(i-1) \bmod j}} \cup B_i^{P_i}\big) \setminus \{\perp\}$.

Proof. Using Lemma 4, it is easy to see that the Algorithm CLP4Del-Branch correctly solves CLP4 Edge Deletion. By Lemma 5 and the fact that the height of the search tree is bounded from above by ℓ, it runs in $(48 \cdot O(\log n) + 24)^\ell \cdot n^{O(1)} = c^\ell \cdot (\ell \log \ell)^\ell \cdot n^{O(1)}$ time for a constant c. ☐

With Theorem 2 and using the same techniques as applied for CLP3 Edge Insertion and CLP3 [6], we achieve the following result:

Corollary 2. CLP4 Edge Insertion *and* CLP4 *with ℓ edge modifications allowed are fixed-parameter tractable with respect to ℓ.*

5 Concluding Remarks

Our fixed-parameter algorithms for Closest 4-Leaf Power mean the first positive algorithmic results for the most difficult k-leaf power problem with known polynomial-time solvable recognition problem. To the best of our knowledge, so far results in this direction are only obtained for the simpler problems Closest 2-Leaf Power [1, 3, 10] and Closest 3-Leaf Power [6]. As long as it remains open to determine the complexity of k-Leaf Power for $k > 4$, it seems to make little sense to study the more general Closest k-Leaf Power for $k > 4$. Given our new results, it is of particular interest to attack the open problem of finding good polynomial-time approximation algorithms for Closest 3-Leaf Power and Closest 4-Leaf Power. The only known result in this direction is a factor-4 approximation algorithm for Closest 2-Leaf Power [1, 3], the by far simplest of these problems.

Also Closest k-Tree Power problems as introduced by Kearney and Corneil [12] deserve further investigations. Note that they only state a straightforward solution that calls the (exact) tree power recognition algorithm $O(n^\ell)$ times, thus exhaustively trying all possibilities. This clearly does not lead to fixed-parameter tractability since the parameter ℓ (number of edge modifications) appears in the exponent of the polynomial.

References

[1] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56(1–3):89–113, 2004. 2, 9

[2] A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph Classes: a Survey.* SIAM Monographs on Discrete Mathematics and Applications, 1999. 1

[3] M. Charikar, V. Guruswami, and A. Wirth. Clustering with qualitative information. In *Proc. 44th FOCS*, pages 524–533. IEEE Computer Society, 2003. 2, 9

[4] Z.-Z. Chen, T. Jiang, and G. Lin. Computing phylogenetic roots with bounded degrees and errors. *SIAM Journal on Computing*, 32(4):864–879, 2003. 1, 2, 3

[5] Z.-Z. Chen and T. Tsukiji. Computing bounded-degree phylogenetic roots of disconnected graphs. In *Proc. 30th WG*, volume 3353 of *LNCS*, pages 308–319. Springer, 2004. 1

[6] M. Dom, J. Guo, F. Hüffner, and R. Niedermeier. Error compensation in leaf root problems. In *Proc. 15th ISAAC*, volume 3341 of *LNCS*, pages 389–401. Springer, 2004. 1, 2, 3, 6, 9

[7] M. Dom, J. Guo, and R. Niedermeier. Bounded degree Closest k-Tree Power is NP-complete. Manuscript, 10 pages, Institut für Informatik, FSU Jena, Feb. 2005. 2

[8] P. Erdős and L. Pósa. On the maximal number of disjoint circuits of a graph. *Publicationes Mathematicae Debrecen*, 9:3–12, 1962. 8

[9] M. R. Fellows. Blow-ups, win/win's, and crown rules: Some new directions in FPT. In *Proc. 29th WG*, volume 2880 of *LNCS*, pages 1–12. Springer, 2003. 3

[10] J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Graph-modeled data clustering: Exact algorithms for clique generation. *Theory of Computing Systems*, 2005. 2, 9

[11] T. Jiang, G. Lin, and J. Xu. On the closest tree kth root problem. Manuscript, Department of Computer Science, University of Waterloo, 2000. 2

[12] P. E. Kearney and D. G. Corneil. Tree powers. *Journal of Algorithms*, 29(1):111–131, 1998. 1, 2, 9

[13] M. Křivánek and J. Morávek. NP-hard problems in hierarchical-tree clustering. *Acta Informatica*, 23(3):311–323, 1986. 2, 3

[14] L. C. Lau. Bipartite roots of graphs. In *Proc. 15th ACM-SIAM SODA*, pages 952–961. ACM/SIAM, 2004. 1

[15] L. C. Lau and D. G. Corneil. Recognizing powers of proper interval, split, and chordal graphs. *SIAM Journal on Discrete Mathematics*, 18(1):83–102, 2004. 1

[16] G. Lin, P. E. Kearney, and T. Jiang. Phylogenetic k-root and Steiner k-root. In *Proc. 11th ISAAC*, volume 1969 of *LNCS*, pages 539–551. Springer, 2000. 1, 3, 4

[17] Y. L. Lin and S. S. Skiena. Algorithms for square roots of graphs. *SIAM Journal on Discrete Mathematics*, 8(1):99–118, 1995. 1

[18] R. Motwani and M. Sudan. Computing roots of graphs is hard. *Discrete Applied Mathematics*, 54(1):81–88, 1994. 1

[19] A. Natanzon. Complexity and approximation of some graph modification problems. Master's thesis, Department of Computer Science, Tel Aviv University, 1999. 3

[20] R. Niedermeier. Ubiquitous parameterization—invitation to fixed-parameter algorithms. In *Proc. 29th MFCS*, volume 3153 of *LNCS*, pages 84–103. Springer, 2004. 3

[21] N. Nishimura, P. Ragde, and D. M. Thilikos. On graph powers for leaf-labeled trees. *Journal of Algorithms*, 42(1):69–108, 2002. 1, 2, 3

[22] D. Rautenbach. 4-leafroots. Manuscript, Forschungsinstitut für Diskrete Mathematik, Universität Bonn, June 2004. 2, 4

[23] R. Shamir, R. Sharan, and D. Tsur. Cluster graph modification problems. *Discrete Applied Mathematics*, 144:173–182, 2004. 2

[24] T. Tsukiji and Z.-Z. Chen. Computing phylogenetic roots with bounded degrees and errors is hard. In *Proc. 10th COCOON*, volume 3106 of *LNCS*, pages 450–461. Springer, 2004. 2