Proc. 31st WG-05, Vol. 3787 in LNCS, pp. 397-408, Springer, 2005

Extending the Tractability Border for Closest Leaf Powers*

Michael Dom, Jiong Guo, Falk Hüffner, and Rolf Niedermeier

Institut für Informatik, Friedrich-Schiller-Universität Jena,
Ernst-Abbe-Platz 2, D-07743 Jena, Germany,
{dom,guo,hueffner,niedermr}@minet.uni-jena.de

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.

1 Introduction

Graph powers form a classical concept in graph theory, and the rich literature dates back to the sixties of the previous century. 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. While it is NP-complete to decide whether a given graph is a k-power [10], one can decide in $O(|V|^3)$ time whether a graph is a k-power of a tree for any fixed k [6], and it can be decided in linear time whether a graph is a square of a tree [9].

Here, we concentrate on certain practically motivated variants of tree powers. Whereas Kearney and Corneil [6] study the problem where every tree node one-to-one corresponds to a graph vertex, Nishimura, Ragde, and Thilikos [12] introduce the notion of *leaf powers* where exclusively the tree leaves stand in one-to-one correspondence with the graph vertices. In addition, Lin, Kearney, and Jiang [8] and Chen, Jiang, and Lin [2] 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 [12, 8, 2]. The corresponding recognition problems are called k-Leaf Power [12] and k-Phylogenetic

* Research supported by the Deutsche Forschungsgemeinschaft (DFG), Emmy Noether research group PIAF (fixed-parameter algorithms), NI 369/4.

Root [8], respectively.[1] For $k \leq 4$, both problems are solvable in polynomial time [12, 8]. The complexities of both recognition problems for $k \geq 5$ are open.

Several groups of researchers [2, 6, 8] 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. 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 NP-complete for $k \geq 2$ [6, 5]. One also obtains NP-completeness for the corresponding problems Closest k-Leaf Power [7, 3] and Closest k-Phylogenetic Root [2].

All nontrivial ($k \geq 2$) "approximate recognition" problems in our context turn out to be NP-complete [2, 3, 5–7, 14]. 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 k-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].[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. 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 [12]. The key idea we put forward here and in a companion paper [3] is to again develop and employ forbidden subgraph characterizations of the respective graph classes. In [3], we describe a forbidden subgraph characterization for 3-leaf powers, consisting of five graphs of small size. Here, we employ 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 [12]. However, they are of purely theoretical interest because the running times of these straightforward algorithms are much worse than that of the

[1] Both problems k-Leaf Power and k-Phylogenetic Root ask whether 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*, although we used the term *root* in the conference version of our previous considerations concerning the case $k = 3$ [3].

[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 [14] and Correlation Clustering [1].

known cubic-time algorithms from [12]. 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 [3] 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, analogous results for Closest 4-Leaf Power remained open in [3]. 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.

Due to the lack of space, we omit 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), ignoring any ordering. 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 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 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*—note that a hole is chordless. A *chordal graph* then is a hole-free graph. Let a *minimum edge cut*, denoted $\textsc{MinCut}(G, V_1, V_2)$, be a minimum weight set of edges in $G = (V, E)$ that disconnects all vertices in $V_1 \subseteq V$ from those in $V_2 \subseteq V$. We say a set is *maximal* with respect to some property if it is not a proper subset of another set with that property. For two sets A and B, $A \triangle B$ denotes the *symmetric difference* $(A \setminus B) \cup (B \setminus A)$.

Definition 1 ([12]). *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 $E := \{(u, v) \mid u, v \in V \text{ and } d_T(u, v) \leq k\}$. We call T a* k-leaf root *of T^k.*

The k-Leaf Power (LPk) problem then is to decide, given a graph G, whether there is 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 [2, 8]. 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 $k \geq 2$ [7, 3].

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 to obtain T^k. CLPk Edge Deletion is NP-complete for $k \geq 2$ [11, 3], and CLPk Edge Insertion is NP-complete for $k \geq 3$ but trivially polynomial-time solvable for $k = 2$.

A central technical tool within this work are critical cliques and critical clique graphs as Lin et al. [8] 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 $G \backslash K$, and K is maximal under this property. Consider a graph $G = (V, E)$. Let V_C be the collection of its critical cliques. Then the* critical clique graph $\mathrm{CC}(G)$ *is a graph (V_C, E_C) (we use the term* nodes *for its vertices) 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.*

Definition 3. *Consider a graph $G = (V, E)$ and an arbitrary set of vertices A with $A \cap V = \emptyset$. An unrooted tree $T = (A \cup V, E')$ is called a* k-Steiner root *of G if $E = \{(u, v) \mid u, v \in V$ 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. [8]).

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.

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 the 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. Independently and by different proof techniques, Rautenbach [13] 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.

Theorem 1. *For a graph G, the following are equivalent: (1) G is a 4-leaf power. (2) G is chordal and its critical clique graph $\mathrm{CC}(G)$ is $\mathcal{F}$-free.*

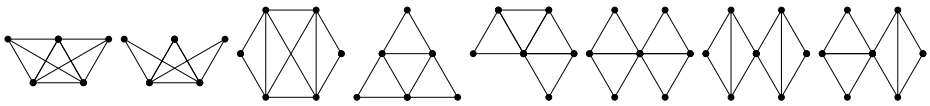

Fig. 1. The eight forbidden subgraphs for critical clique graphs of 4-leaf powers

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 indeed has a 2-Steiner root by using Algorithm SRG (Fig. 2). This algorithm extends a method by Lin et al. [8] 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 (Fig. 2) also generates an output graph with some guaranteed properties for inputs that are $(\mathcal{F} \cup \{C_4, C_5\})$-free but nonchordal graphs. This will be of use for our fixed-parameter algorithms in Sect. 4.

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

The idea of Algorithm SRG 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 K', then the node in the output graph corresponding to one of the endpoints of e is connected by edges with the other nodes corresponding to K and the node in the output graph corresponding to the other endpoint of e 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 to s_K (see Fig. 3 for an example).

We can show that Algorithm SRG fulfills the following 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 two claims do not require chordality of the input graph. We will make use of this

SRG(CC(G) $= (V_C, E_C)$)
Input: $(\mathcal{F} \cup \{C_4, C_5\})$-free critical clique graph CC(G) $= (V_C, E_C)$
Output: Pseudo Steiner root graph of CC(G)

1 $S \leftarrow (\{b_c \mid c \in V_C\}, \emptyset)$
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 each** $c \in K, c \neq c_1$**:**
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 each** $c \in K, c \neq c_1$**:**
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 each** $c \in K, c \neq c_2$**:**
16 Insert an edge between b_{c_2} and b_c
17 **else:**
18 **for each** $c \in K, c \neq c_1$**:**
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 each** $c \in K$**:**
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 new edge-connected Steiner nodes s_1 and s_2 to S and connect s_1 by an edge to an arbitrary node in S_1 and s_2 to an arbitrary node in S_2
27 **return** S

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

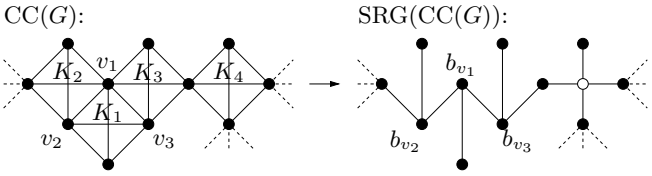

Fig. 3. Example of a subgraph of a critical clique graph CC(G) and the pseudo Steiner root graph computed for this subgraph. Algorithm SRG first considers the maximal clique K_1 with $c_1 = v_1$ (see Fig. 2) and inserts edges between b_{v_1} and the other nodes corresponding to K_1. Thereafter, the cliques K_2 and K_3 are considered. When considering K_4, Algorithm SRG inserts a Steiner node (drawn white).

fact in Sect. 4 when we have to modify a critical clique graph to make it chordal. The claims are:

1. Every maximal clique K of an $(\mathcal{F} \cup \{C_4, C_5\})$-free critical clique graph $\mathrm{CC}(G)$ is considered at least once by Algorithm SRG, and for every node pair u, v in K, a path of length at most two is generated between the corresponding nodes of u and v in the output graph.
2. For an $(\mathcal{F} \cup \{C_4, C_5\})$-free critical clique graph $\mathrm{CC}(G) = (V_C, E_C)$ Algorithm SRG outputs a graph with the following property: If two nodes $u, v \in V_C$ are not adjacent in $\mathrm{CC}(G)$, then the distance between the nodes corresponding to u and v in the output graph is at least three.
3. For a chordal and $\mathcal{F}$-free critical clique graph $\mathrm{CC}(G)$ the output graph of Algorithm SRG is a tree.

Note that Algorithm SRG runs in polynomial time, as there are at most $2 \cdot |E_C|$ maximal cliques in an $(\mathcal{F} \cup \{C_4, C_5\})$-free critical clique graph $\mathrm{CC}(G) = (V_C, E_C)$.

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 resembles our previous work for CLP3 [3]; 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" [3]. Therefore, the basic idea for CLP3 Edge Deletion as well as for CLP4 Edge Deletion is to use the forbidden subgraph characterization in a depth-bounded 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 upper-bound the number of branching cases by a function depending only on ℓ, since the depth can be bounded from above by 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 [3].

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 $\mathrm{CC}(G) = (V_C, E_C)$ instead of G has two consequences: First, a deletion of an edge in $\mathrm{CC}(G)$ can represent several edge deletions in G. Consider an edge e in $\mathrm{CC}(G)$ between two nodes that represent critical cliques of sizes 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 depth-bounded search tree algorithm by more than one. Second, if two adjacent nodes in $\mathrm{CC}(G)$ obtain an 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. Subsequently, 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. This can be done in $O(|V_C| \cdot |E_C|)$ time.

The main obstacle in obtaining fixed-parameter tractability 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 only depending on ℓ. For CLP3 Edge Deletion, the key observation is that the critical clique graph $\mathrm{CC}(G)$ of a graph G containing neither a bull nor a dart nor a gem nor a C_4 contains no triangles. This allows to show that, after destroying the forbidden subgraphs bull, dart, gem, and C_4 in G, no hole in $\mathrm{CC}(G)$ can be "accidentally" destroyed by merge operations between its nodes and, therefore, one has to delete at least one edge of every hole. Since moreover making a triangle-free graph chordal means to make it a forest, a minimum weight set of edges to be deleted to make $\mathrm{CC}(G)$ chordal can be obtained in polynomial time by searching for a maximum weight spanning tree. Unfortunately, there *can* be triangles in 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.

The idea is to examine the output graph $\mathrm{SRG}(\mathrm{CC}(G))$ 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 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 (see Fig. 4).

For the pseudocode of this algorithm, which is presented in Fig. 5, we introduce some notation for the mapping between the nodes of a critical clique graph and the nodes of its pseudo Steiner root graph.

Definition 4. *Consider a critical clique graph* $\mathrm{CC}(G) = (V_C, E_C)$ *and a pseudo Steiner root graph* $S = (V_S, E_S)$ *constructed by Algorithm SRG for* $\mathrm{CC}(G)$. *For* $v \in V_C$ *we use* $S(v)$ *to denote the node from* V_S *that corresponds to* v, *and*

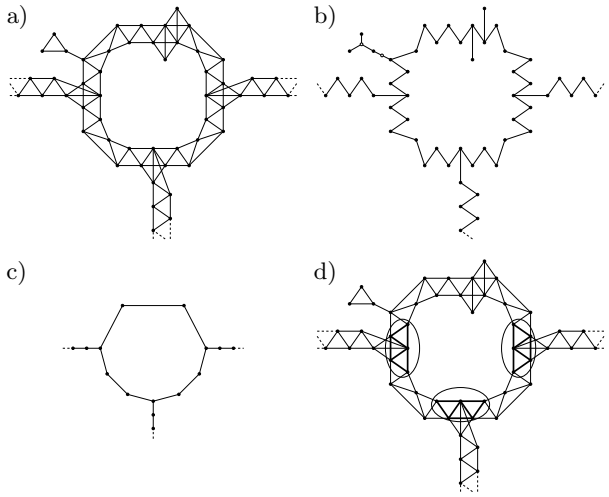

Fig. 4. Illustration of finding and destroying holes in an $(\mathcal{F} \cup \{C_4, C_5\})$-free critical clique graph: a) A nonchordal critical clique graph $\mathrm{CC}(G)$. b) The pseudo Steiner root graph S constructed by Algorithm SRG for $\mathrm{CC}(G)$. c) The reduced pseudo Steiner root graph S'. d) The sets marked with an ellipsis correspond to the degree-3 nodes in S'. Our algorithm for CLP4 Edge Deletion either deletes one of the bold edges or it deletes a minimum weight set of edges between two of the node sets marked with an ellipsis (these sets are called "big node areas" in Def. 5).

for $v_S \in V_S$, we define $S^{-1}(v_S)$ as the node in V_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 $V'_C \subseteq V_C$, $S(V'_C) := \{S(v) \mid v \in V'_C\}$, and for $V'_S \subseteq V_S$, $S^{-1}(V'_S) := \{S^{-1}(v) \mid v \in V'_S\}$.

To define the branching set D in line 18 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 lines 11–19 of Algorithm* CLP4Del-Branch *(Fig. 5) and that has degree at least* 3 *in the constructed pseudo Steiner root graph S' (see Fig. 6).*

For a cycle Q in a pseudo Steiner root graph S as constructed by Algorithm CLP4Del-Branch *in line 16, let $v_0, \ldots, v_{q-1}$ be the big nodes in Q, ordered by their appearance in Q, and for every node v_i with $0 \leq i < q$ let P_i be the path in Q between v_i and $v_{(i+1) \bmod q}$.*

With P_i^+ we denote the path P_i plus its attached trees, that is, the maximal set of nodes in S such that P_i^+ contains the nodes of P_i and such that P_i^+ induces a connected component in $S \setminus \{v_i, v_{(i+1) \bmod q}\}$.

We further denote with A_i, $0 \leq i < q$, the big node areas *that are defined as*

$$A_i := S^{-1}(\{v \in Q \mid d_S(v_i, v) \leq 2\}) \setminus \{\perp\}.$$

CLP4Del-Branch(G, ℓ)
Input: A graph $G = (V, E)$ and an integer ℓ
Output: A set of at most ℓ 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 Compute CC(G)
3 **if** CC(G) contains an induced forbidden subgraph $F \in \mathcal{F} \cup \{C_4, C_5\}$**:**
4 **for each** edge e in F**:**
5 $X \leftarrow$ CLP4Del-Branch(CC-Del$(G, \{e\})$, $\ell-$CC-Weight$(G, \{e\})$)
6 **if** $X \neq$ **nil: return** $X \cup \{e\}$
7 **return nil**
8 $S \leftarrow$ SRG(CC(G))
9 **if** S is a tree**: return** $\emptyset$
10 $S' \leftarrow S$
11 **while** there is a degree-1-node u in S'**:**
12 delete u
13 **while** there is a path (u, v, w) of three degree-2-nodes in S'**:**
14 delete v and insert an edge between u and w
15 $Q' \leftarrow$ shortest cycle in S'
16 $Q \leftarrow$ cycle in S corresponding to Q'
17 $H \leftarrow S^{-1}(Q) \setminus \{\perp\}$
18 Determine a set D (see Lemma 4) of edge sets in CC$(G)[H]$ such that at least one edge set $d \in D$ is a subset of an optimal solution
19 **for each** $d \in D$**:**
20 $X \leftarrow$ CLP4Del-Branch(CC-Del(G, d), $\ell-$CC-Weight(G, d))
21 **if** $X \neq$ **nil: return** $X \cup d$
22 **return nil**

Fig. 5. 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 the clique represented by K_1 and the other endpoint in the clique represented by K_2 are deleted by CC-Del(G, d). The function CC-Weight(G, d) returns the sum of the weights of the edges in d.

The following lemma will help us to show that the cycle Q determined by Algorithm CLP4Del-Branch (Fig. 5) in line 16 indeed induces at least one hole in CC(G).

Lemma 3. *Consider a cycle Q in a pseudo Steiner root graph S as constructed by Algorithm* CLP4Del-Branch *(Fig. 5) in line 16. Let $v_0, \ldots, v_{p-1}$ be the nodes of Q, ordered by their appearance in Q. Then there is no edge $(S^{-1}(v_i), S^{-1}(v_j))$ with $0 \leq i, j < p$ in* CC(G) *such that v_i and v_j have a distance of more than 2 on Q.*

The main observation that helps to bound the number of branching cases and, hence, leads to our fixed-parameter algorithm is that for a cycle Q in a pseudo Steiner root graph S the number of branching cases is independent of

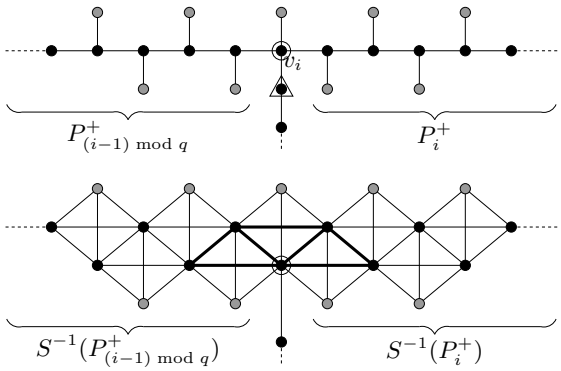

Fig. 6. Illustration for Definition 5. The upper picture shows a part of the pseudo Steiner root graph S. The encircled node v_i is a big node; black nodes are part of a cycle in S. The grey nodes are deleted by the data reduction in lines 11–14 of Algorithm CLP4Del-Branch. The only Steiner node in this example is the node marked with a triangle. The lower picture shows the corresponding part of CC(G). The bold edges are those between vertices of the big node area A_i.

the lengths of the paths in Q between the big nodes: If we want to disconnect two big node areas, then it is always optimal to take an edge set with minimum weight whose removal disconnects the two big node areas. Such an edge set can be found in polynomial time by maximum flow techniques.

Lemma 4. *In* CLP4Del-Branch*, the branching set D chosen as follows contains at least one subset of an optimal solution: Either delete an edge in a big node area, that is, an edge (u, v) with $u, v \in A_i$ for some $0 \leq i < q$, or delete a set of edges*

$$\textsc{MinCut}(\mathrm{CC}(G)[S^{-1}(P_i^+) \setminus \{\perp\}], A_i, A_{(i+1) \bmod q}),$$

that is, delete a minimum weight set of edges such that all paths between two neighboring big node areas are destroyed.

It remains to show the complexity of CLP4Del-Branch. All steps within a single invocation of CLP4Del-Branch can be done in polynomial time. We therefore focus on the number of recursive calls. In line 4, there can be at most 10 recursive calls corresponding to at most 10 edges to delete in a forbidden subgraph (for example F_3 in Fig. 1); as we will see, this is dominated by the number of recursive calls in line 20 for destroying a long cycle.

A well-known result by Erdős and Pósa [4] 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 vertices. 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 line 18 of Algorithm* CLP4Del-Branch *as described by Lemma 4, we can upper-bound its size by $|D| \leq 96 \cdot \log |V| + 24$.*

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

Proof. By Lemma 5 and the fact that the height of the search tree is bounded from above by ℓ, CLP4Del-Branch runs in $(96 \cdot \log |V| + 24)^\ell \cdot |V|^{O(1)} \leq c^\ell \cdot (\ell \log \ell)^\ell \cdot n^{O(1)}$ time for a constant c (the inequality holds because $(\log n)^\ell \leq (3\ell \log \ell)^\ell + n$ for all values of n and ℓ). □

With Theorem 2 and using the same techniques as applied for CLP3 Edge Insertion and CLP3 [3], we achieve the following result:

Theorem 3. *1.* CLP4 Edge Insertion *with ℓ edge insertions allowed is fixed-parameter tractable with respect to ℓ.*
2. CLP4 *with ℓ edge insertions and deletions is fixed-parameter tractable with respect to ℓ.*

References

1. M. Charikar, V. Guruswami, and A. Wirth. Clustering with qualitative information. In *Proc. 44th FOCS*, pages 524–533. IEEE Computer Society, 2003. To appear in *J. Comput. System Sci.*
2. Z.-Z. Chen, T. Jiang, and G. Lin. Computing phylogenetic roots with bounded degrees and errors. *SIAM J. Comput.*, 32(4):864–879, 2003.
3. 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. Long version to appear under the title *Error compensation in leaf power problems* in *Algorithmica*.
4. P. Erdős and L. Pósa. On the maximal number of disjoint circuits of a graph. *Publ. Math. Debrecen*, 9:3–12, 1962.
5. T. Jiang, G. Lin, and J. Xu. On the closest tree kth root problem. Manuscript, Department of Computer Science, University of Waterloo, 2000.
6. P. E. Kearney and D. G. Corneil. Tree powers. *J. Algorithms*, 29(1):111–131, 1998.
7. M. Křivánek and J. Morávek. NP-hard problems in hierarchical-tree clustering. *Acta Inform.*, 23(3):311–323, 1986.
8. 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.
9. Y.-L. Lin and S. S. Skiena. Algorithms for square roots of graphs. *SIAM J. Discrete Math.*, 8(1):99–118, 1995.
10. R. Motwani and M. Sudan. Computing roots of graphs is hard. *Discrete Appl. Math.*, 54(1):81–88, 1994.
11. A. Natanzon. Complexity and approximation of some graph modification problems. Master's thesis, Department of Computer Science, Tel Aviv University, 1999.
12. N. Nishimura, P. Ragde, and D. M. Thilikos. On graph powers for leaf-labeled trees. *J. Algorithms*, 42(1):69–108, 2002.
13. D. Rautenbach. 4-leafroots. Manuscript, Forschungsinstitut für Diskrete Mathematik, Universität Bonn, June 2004.
14. R. Shamir, R. Sharan, and D. Tsur. Cluster graph modification problems. *Discrete Appl. Math.*, 144:173–182, 2004.