Originally published in *Algorithmica*, 44(4):363–381. Springer, 2006.

Error Compensation in Leaf Power Problems*

Michael Dom† Jiong Guo† Falk Hüffner†
Rolf Niedermeier†

April 6, 2005

Abstract

The k-Leaf Power recognition problem is a particular case of graph power problems: For a given graph it asks whether there exists an unrooted tree—the k-leaf root—with leaves one-to-one labeled by the graph vertices and where the leaves have distance at most k iff their corresponding vertices in the graph are connected by an edge. Here we study "error correction" versions of k-Leaf Power recognition—that is, adding or deleting at most l edges to generate a graph that has a k-leaf root. We provide several NP-completeness results in this context, and we show that the NP-complete Closest 3-Leaf Power problem (the error correction version of 3-Leaf Power) is fixed-parameter tractable with respect to the number of edge modifications or vertex deletions in the given graph. Thus, we provide the seemingly first nontrivial positive algorithmic results in the field of error compensation for leaf power problems with $k > 2$. To this end, as a result of independent interest, we develop a forbidden subgraph characterization of graphs with 3-leaf roots.

Key Words. NP-completeness, Fixed-parameter tractability, Graph algorithms, Graph modification, Graph power, Leaf power, Forbidden subgraph characterization

1 Introduction

Graph powers are a classical concept in graph theory [2, Section 10.6] with recently increased interest from an algorithmic point of view. The *k-power* of a graph $G = (V, E)$ is the graph $G^k = (V, E')$ with $(u, v) \in E'$ iff there is a path

*A preliminary version of this paper appears under the title "Error Compensation in Leaf Root Problems" in the proceedings of the 15th Annual International Symposium on Algorithms and Computation (ISAAC '04), held in Hong Kong, China, December 20–22, 2004 [8]. Supported by the Deutsche Forschungsgemeinschaft (DFG), Emmy Noether research group PIAF (fixed-parameter algorithms), NI 369/4. Significant parts of this work have been done while the authors were at the Universität Tübingen.

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

Originally published in *Algorithmica*, 44(4):363–381. Springer, 2006.

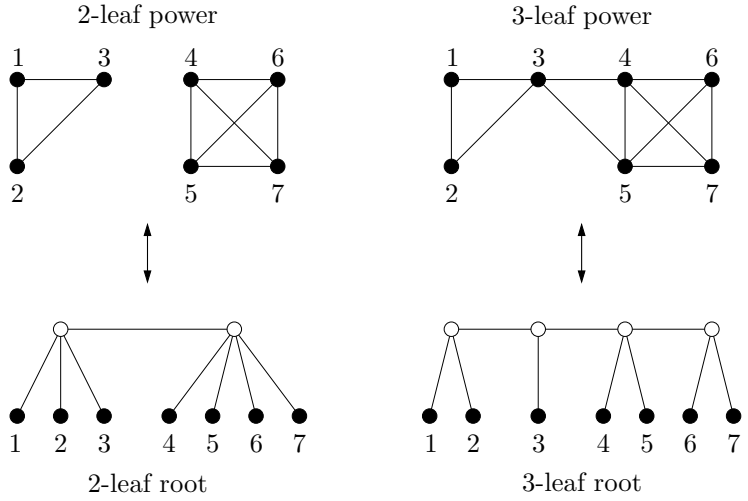

Figure 1: Leaf powers and leaf roots. The leaves of a leaf root stand in one-to-one correspondence to the vertices of its leaf power.

of length at most k between u and v in G. We say G is the *k-root* of G^k and G^k is the *k-power* of G; deciding whether a given graph is a power of some other graph is called the *graph power* problem. It is NP-complete in general [27], but one can decide in $O(|V|^3)$ time whether a graph is a k-power of a tree for any fixed k [20]. In particular, it can be decided in linear time whether a graph is a square (that is, a 2-power) of a tree [26, 22]. Lau [22] shows that it can be found in polynomial time whether a graph is a square of a bipartite graph, but it is NP-complete to decide whether a graph is a cube of a bipartite graph. Moreover, Lau and Corneil [23] give a polynomial-time algorithm for recognizing k-powers of proper interval graphs for every k and show that, contrariwise, recognizing squares of chordal graphs and split graphs is NP-complete.

Here we concentrate on certain variants of tree powers. Whereas Kearney and Corneil [20] study the problem where every tree node one-to-one corresponds to a graph vertex, Nishimura, Ragde, and Thilikos [31] introduced the notion of *leaf powers* where exclusively the tree leaves stand in one-to-one correspondence to the graph vertices (see Figure 1). Motivated by applications in computational biology, Lin, Kearney, and Jiang [25] and Chen, Jiang, and Lin [5] examine the variant of leaf powers where all inner nodes of the root tree have degree at least three. The corresponding algorithmic problems to decide whether a graph has such a k-root are called k-Leaf Power recognition [31] and k-Phylogenetic Root [25], respectively. For $k \leq 4$, both problems are solvable in polynomial time [31, 25]. The complexities of both problems for $k \geq 5$ are still open. Moreover, Chen, Jiang, and Lin [5] and Chen and Tsukiji [6] show that, under the assumption that the maximum degree of the phylogenetic root is bounded from above by a constant, there is a linear-time algorithm that determines

whether a graph has a k-phylogenetic root (that is, a k-leaf root with minimum degree three) for arbitrary k.

What to do if the given input graph has no k-leaf root? In particular, the input graph might be "close" to having a k-root but due to certain "errors" (as occur in many practical applications), the graph structure would need some "correction" first before the computation of a k-leaf root is doable. This problem was already recognized by Kearney and Corneil [20], and they introduce 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. Unfortunately, this problem turns out to be NP-complete for $k \geq 2$ [20, 18]. One also obtains NP-completeness for the corresponding problems Closest k-Phylogenetic Root [5, 35] and, as we point out here, Closest k-Leaf Power.[1] In addition, for Closest k-Leaf Power we study other edge modification problems—namely, only to allow edge deletions or only to allow edge insertions—together with the vertex deletion variant, and we show NP-completeness. See Table 1 in Section 4 for an overview concerning classical complexity and hardness results for Closest k-Leaf Power and its variants.

To the best of our knowledge, except for the "simpler" case $k = 2$ the above error correction scenario so far only led to results showing hardness of complexity no matter whether concerning closest tree powers, closest phylogenetic roots, or closest leaf powers. We are not aware of any results concerning approximation or nontrivial exact algorithms. In contrast, we show the seemingly first positive algorithmic results in this context, proving fixed-parameter tractability with respect to the number l of edge modifications for Closest 3-Leaf Power and all its variants mentioned above. To achieve our fixed-parameter results, we develop a novel forbidden subgraph characterization of graphs that are 3-leaf powers—a result that may be of interest on its own: A graph is a 3-leaf power iff it is chordal and it contains none of the 5-vertex graphs bull, dart, and gem as an induced subgraph (see Section 3 for details). A much simpler characterization of graphs that are 2-leaf powers is already known by forbidding an induced path of three vertices [34]. This characterization finds direct applications in corresponding fixed-parameter algorithms [15] (fixed-parameter tractability is also implied by a more general result of Leizhen Cai [3]), whereas our new characterization of 3-leaf powers requires a more tricky approach. We mention in passing that in a companion paper we show that Closest 4-Leaf Power is fixed-parameter tractable [9]. The approach there is similar but the corresponding forbidden subgraph characterization becomes much more complicated and, therefore, causes a much increased algorithmic complexity.

[1] Note that both problems ask for the closest graph which 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 previous works [7, 8].

Originally published in *Algorithmica*, 44(4):363–381. Springer, 2006.

2 Preliminaries, Basic Definitions, and Previous Work

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 $G = (V, E)$ if $V' \subseteq V$ and $E' = \{(u, v) \mid u, v \in V'$ and $(u, v) \in E\}$. An edge between two vertices of a cycle that is not part of the cycle is called *chord*. An induced, chordless cycle of length at least four is called *hole*. A *chordal graph* is a graph that contains no hole. For two sets A and B, $A \triangle B$ denotes the *symmetric difference* $(A \setminus B) \cup (B \setminus A)$.

We use the standard notations for special graphs: P_n for a path of n vertices and C_n for a cycle of n vertices.

Closely related to the well-known graph power concept [2, Section 10.6] is the notion of a *k-leaf power* of a tree, introduced by Nishimura, Ragde, and Thilikos [31]:

Definition 1. For 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 tree T is called a *k-leaf root* of T^k.

The following problem is inspired by forming a phylogenetic tree[2] based on a binary similarity measure.

> k-Leaf Power recognition (LPk)
>
> **Instance:** A graph G.
>
> **Question:** Is there a tree T such that $T^k = G$?

Nishimura et al. [31] show that k-Leaf Power recognition can be solved in polynomial time for $k \leq 4$. As already Nishimura et al. point out, in practice phylogenetic problems involve errors in similarity estimators. This motivates the following problem.

> Closest k-Leaf Power (CLPk)
>
> **Instance:** A graph $G = (V, E)$ and a nonnegative integer l.
>
> **Question:** Is there a tree T such that T^k and G differ by at most l edges, that is, $|E(T^k) \triangle E(G)| \leq l$?

More precisely, this problem is denoted as CLPk Edge Editing. In this paper we also study two variations, where the distance estimator is assumed to have only one-sided errors:

[2] That is, a tree where leaves correspond to species and internal nodes represent evolutionary events.

- CLPk Edge Insertion: Only inserting edges into G is allowed to obtain T^k (that is, $E(T^k) \supseteq E(G)$);
- CLPk Edge Deletion: Only deleting edges from G is allowed to obtain T^k (that is, $E(T^k) \subseteq E(G)$).

In addition, we examine the problem that considers deleting vertices instead of edges.

- CLPk Vertex Deletion: Is there a tree T such that we can get T^k by deleting at most l vertices from G?

CLP2 Edge Editing has been studied under various names in the literature. The first proof of its NP-completeness is due to Křivánek and Morávek [21], where it is called Hierarchical-Tree Clustering, later usually referenced as Fitting Ultrametric Trees.

Independently, the problem is studied by Shamir, Sharan, and Tsur [34] as Cluster Editing and by Bansal, Blum, and Chawla [1] as Correlation Clustering. Charikar, Guruswami, and Wirth [4] show that there exists some constant $\epsilon > 0$ such that it is NP-hard to approximate Correlation Clustering to within a factor of $1 + \epsilon$ (that is, Correlation Clustering is APX-hard).

CLP2 Edge Deletion (also known as Cluster Deletion) was shown to be NP-complete by Natanzon [28]. Moreover, Shamir et al. [34] show that Cluster Deletion is APX-hard.

Lin, Kearney, and Jiang [25] consider a variant of k-Leaf Power recognition where the inner nodes of the output tree are not allowed to have degree 2. They call this problem k-Phylogenetic Root (PRk), and show that PRk can be solved in linear time for $k \leq 4$.[3] As for leaf roots, the generalization that allows for the input graph to contain errors is a better match for the biological motivation, and one can ask for the Closest k-Phylogenetic Root (CPRk), defined analogous to Closest k-Leaf Power. CPRk is examined by Chen, Jiang, and Lin [5], who show that it is NP-complete for $k \geq 2$. Even if the maximum degree of the k-root is bounded above by a constant, Closest k-Phylogenetic Root is NP-complete [35].

Our positive algorithmic results appear in the context of parameterized complexity [11]. The core concept herein is that of fixed-parameter tractable problems which we concretely define here in terms of the CLP3 problem: CLP3 is *fixed-parameter tractable* with respect to parameter l; that is, we show that CLP3 can be solved in $f(l) \cdot n^{O(1)}$ time, where f is an (exponential) function only depending on l. Thus, we can confine the "combinatorial explosion" exclusively to the parameter l. In this way for small l, as might be naturally expected since l refers to the number of errors, efficient (polynomial-time) algorithms are possible that provide *optimal* solutions. We refer to the literature with its by now numerous surveys for more motivation and background concerning fixed-parameter complexity [10, 13, 14, 30].

[3] For $k = 4$, they show this only for connected graphs.

Originally published in *Algorithmica*, 44(4):363–381. Springer, 2006.

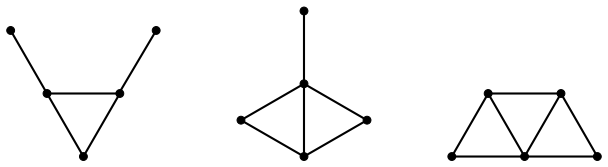

Figure 2: 5-vertex graphs that occur as forbidden induced subgraphs

3 Forbidden Subgraph Characterization for 3-Leaf Powers

It is not hard to see that graphs that are 2-leaf powers are exactly the graphs where every connected component is a clique. Shamir et al. [34] note that these graphs are characterized by a forbidden induced subgraph, namely a path of three vertices (P_3). In this section we derive a similar, but far less evident forbidden subgraph characterization of 3-leaf powers: they are chordal graphs that contain no induced bull, dart, or gem (see Figure 2).

Forbidden subgraph characterizations can be valuable in various ways. We describe three here.

First, they help with the recognition problem and lead to its polynomial-time solvability in a straightforward way. Second, we also make use of them for proving our NP-completeness results in the next section. Third, they can lead to fixed-parameter algorithms for the corresponding graph modification problems. Leizhen Cai [3] shows that with a finite set of forbidden subgraphs, finding the l edges to be modified is fixed-parameter tractable with respect to l. Using the single forbidden subgraph P_3, this immediately applies to the case of 2-leaf powers; for 3-leaf powers, exploiting the subsequent forbidden subgraph characterization is one of the decisive ingredients of the fixed-parameter algorithms presented in Section 5. Note, however, that here Cai's result does not apply directly, since chordal graphs do not admit a characterization by a *finite* set of forbidden subgraphs.

As we will see, 3-leaf powers are closely connected to the concept of a *critical clique*, which was introduced by Lin et al. [25].

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.

Since between the vertices of two critical cliques either all pairwise or no connections are present, the concept of a *critical clique graph* [25] comes up naturally. As we will see, the structure of the critical clique graph is already close to the structure of the 3-leaf roots we are looking for. For easier distinction from the elements of G, we use the term *nodes* for vertices in the critical clique graph.

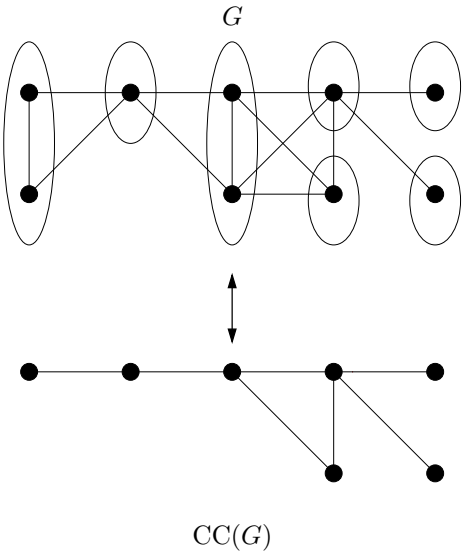

Figure 3: A graph G and its critical clique graph $\mathrm{CC}(G)$. *Ovals* denote the critical cliques of G.

Definition 3. Given 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.

An example of a graph and its critical clique graph is shown in Figure 3. Since every vertex of G belongs to exactly one critical clique, the critical clique graph of G can be constructed in $O(n \cdot m)$ time by iterating through the vertices and constructing the critical clique they are part of by comparing their neighborhood to that of all adjacent vertices.

The following connection to 3-leaf powers can be shown:

Lemma 1. *If a graph G is a 3-leaf power, then every clique in G contains vertices of at most two critical cliques.*

Proof. If two vertices of G are adjacent to the same inner node in a 3-leaf root T of G, then they have identical neighborhood in G, since their distances to other leaves in T are identical. Therefore, vertices from different critical cliques cannot be adjacent to the same inner node of T. Thus, if a clique K in G contains vertices of at least three critical cliques, the vertices of K must be adjacent to at least three different inner nodes of T. Two of three inner nodes in a tree have distance at least 2, which already yields a distance of 4 between

Originally published in *Algorithmica*, 44(4):363–381. Springer, 2006.

their leaf children, contradicting the assumption that their leaf children are part of K. □

The following lemma shows that C_4-free graphs with the critical clique structure that we just derived for 3-leaf powers can be characterized by a set of forbidden subgraphs.

Lemma 2. *For a C_4-free graph G, the following are equivalent:*

(1) There is a clique K in G that contains vertices from at least three critical cliques.

(2) $\mathrm{CC}(G)$ contains a triangle.

(3) G contains a bull, dart, or gem (see Figure 2) as an induced subgraph.

Proof. **(1) ⇒ (2):** This follows directly from the definition of critical clique graph.

(2) ⇒ (3): Consider three nodes in $\mathrm{CC}(G)$ that form a triangle and take one vertex of each of the corresponding critical cliques in G. These three vertices are pairwise connected by an edge and must have pairwise distinct neighborhoods. We make a case distinction based on whether there exists a non-common neighbor that is connected to exactly one of the three vertices or not. For each case, we can get bull, dart, or gem in G.

(3) ⇒ (1): Assume that G contains a forbidden subgraph. Let u, v, w be the vertices of a triangle in the forbidden subgraph (in the case of the gem, the triangle which contains both degree-3 vertices). Then u, v, w form a clique. Let x and y be the remaining two vertices in the subgraph. Since each of u, v, w is adjacent to a different combination of x and y, they belong to three different critical cliques. □

Utilizing Lemmas 1 and 2, we can obtain the main theorem of this section.

Theorem 1. *For a graph G, the following are equivalent:*

(1) G has a 3-leaf root.

(2) $\mathrm{CC}(G)$ is a forest.

(3) G is chordal and contains no bull, dart, or gem as an induced subgraph.

Proof. **(1) ⇒ (3):** If G is a leaf power, then G must be chordal [25]. Then, by Lemma 1 and Lemma 2, it does not contain any of the forbidden subgraphs.

(3) ⇒ (2): If G is chordal, then so is $\mathrm{CC}(G)$, since if $\mathrm{CC}(G)$ contained a hole, we could also find a hole in G by taking one arbitrary vertex from each critical clique on the cycle. With Lemma 2, it follows that $\mathrm{CC}(G)$ is a forest.

Originally published in *Algorithmica*, 44(4):363–381. Springer, 2006.

Table 1: Complexity of Closest k-Leaf Power. The polynomial-time solvability of CLP2 Edge Insertion is trivial; the results for $k \geq 3$ are discussed in Section 4. The main new result is NP-completeness of CLPk Edge Insertion for $k \geq 3$.

	$k = 2$	$k \geq 3$
Edge editing	NP-complete [21]	NP-complete
Edge deletion	NP-complete [28]	NP-complete
Edge insertion	P	NP-complete
Vertex deletion	NP-complete [24]	NP-complete [24]

(2) ⇒ (1): For each connected component of CC(G), construct a leaf root by attaching to each node a new leaf node for each vertex of the corresponding critical clique. Finally, create a new node and connect this node to an arbitrary inner node of each newly constructed tree. Then the resulting tree T is a 3-leaf root of G. To see this, consider two vertices u, v, $u \neq v$ of G. They are connected in G iff they are in the same critical clique, or they are in two adjacent critical cliques. This is equivalent to the distance of u and v in T being 2 and 3, respectively. □

4 NP-Completeness Results

In Table 1 we summarize known and new results on the classical complexity (P vs. NP) of Closest k-Leaf Power problems.

The NP-completeness of CLPk Edge Editing and CLPk Edge Deletion for $k \geq 3$ can be shown by a straightforward adaption of the NP-completeness proof for CPRk by Chen et al. [5].

Theorem 2. CLPk Edge Editing *and* CLPk Edge Deletion *are NP-complete for* $k \geq 3$.

Note that this reduction also implies APX-hardness for CLPk Edge Editing and CLPk Edge Deletion.

CLP2 Edge Insertion can be trivially solved in linear time, since it is exactly the problem of adding edges to a graph so that each connected component becomes a clique.

We show in the following that for $k = 3$ CLPk Edge Insertion becomes NP-complete by giving a reduction from Maximum Edge Biclique.

> Maximum Edge Biclique
>
> **Instance:** A bipartite graph $G = (V_1 \dot\cup V_2, E)$, and a nonnegative integer l.
>
> **Question:** Does G contain a *biclique* with at least l edges, that is, are there two sets $A_1 \subseteq V_1, A_2 \subseteq V_2$ with $(v_1, v_2) \in E$ for all $v_1 \in A_1, v_2 \in A_2$ and $|A_1| \cdot |A_2| \geq l$?

Originally published in *Algorithmica*, 44(4):363–381. Springer, 2006.

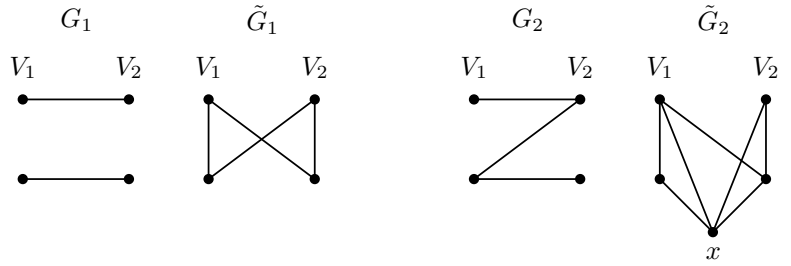

Figure 4: Forbidden subgraphs for bicliques (G_1 and G_2) and the results of their transformation as described in the proof of Theorem 3 ($\tilde{G}_1$ and $\tilde{G}_2$). Note that $\tilde{G}_1$ and $\tilde{G}_2$ are forbidden subgraphs for 3-leaf powers (Theorem 1).

Maximum Edge Biclique has been shown to be NP-complete by Peeters [32]. Consider the problem Biclique Deletion which asks for the least number of edges to delete to make a bipartite graph a biclique (plus isolated vertices). Biclique Deletion is clearly also NP-complete, since a solution for Maximum Edge Biclique with l edges can be found by solving Biclique Deletion with $|E| - l$ deletions. The NP-completeness of CLP3 Edge Insertion can now be shown by a reduction from Biclique Deletion. The key idea is to exploit forbidden subgraph characterizations for both problems. The following lemma is easy to observe.

Lemma 3. *A bipartite graph $G = (V_1 \dot\cup V_2, E)$ without isolated vertices is a biclique iff it does not contain a $2K_2$ (G_1 in Figure 4) or a P_4 (G_2 in Figure 4) as an induced subgraph.*

With Lemma 3, we show the NP-completeness of CLPk Edge Insertion by giving a reduction from Biclique Deletion.

Theorem 3. CLPk Edge Insertion *is NP-complete for $k \geq 3$.*

Proof. We show here only the NP-completeness of CLP3 Edge Insertion. Using the same technique as employed in [5] for CPRk, we can also show the NP-completeness of CLPk Edge Insertion with $k > 3$.

CLP3 Edge Insertion is clearly in NP. For a bipartite graph $G = (V_1 \dot\cup V_2, E)$ which is an instance of Biclique Deletion, we assume, without loss of generality, that G has no isolated vertices. Then, we construct $\tilde{G}$ by taking the complement of G and adding a special vertex x which is connected to all other vertices. Formally, $\tilde{G} = (V_1 \dot\cup V_2 \dot\cup \{x\}, \tilde{E})$ with

$$\begin{aligned}\tilde{E} =& \{(u,v) \mid u,v \in V_1 \cup V_2, (u,v) \notin E\} \\ &\cup \{(x,v) \mid v \in V_1 \cup V_2\}.\end{aligned}$$

Observe that G contains a $2K_2$ (P_4) iff $\tilde{G}$ contains a C_4 (gem), see Figure 4 for an illustration. Furthermore, $\tilde{G}$ cannot contain a bull or dart, since it contains no three independent vertices. Moreover, there is no hole of length greater than four

in $\tilde{G}$, because such a hole would contain at least three nodes from either V_1 or V_2, and these three nodes induce a clique. Based on this observation, the equivalence between the solution of Biclique Deletion on G and the solution of CLP3 Edge Insertion on $\tilde{G}$ follows directly from Theorem 1 and Lemma 3. □

The NP-completeness of CLPk Vertex Deletion for $k \geq 2$ follows directly from a result by Lewis and Yannakakis [24], who show that the vertex deletion problem is NP-complete for any nontrivial hereditary graph property.

5 Fixed-Parameter Tractability Results for CLP3

In this section we show fixed-parameter tractability with respect to the number of editing operations l for CLP3 Edge Insertion, CLP3 Edge Deletion, CLP3 Edge Editing, and CLP3 Vertex Deletion. According to the characterization of 3-leaf powers from Theorem 1, the algorithms have two tasks to fulfill:

(1) Edit the input graph G to get rid of the forbidden subgraphs bull, dart, and gem.

(2) Edit G to make it chordal.

Lin et al. [25] show the usefulness of the critical clique graph for the construction of the 3-leaf root (see also Section 3). The following lemma demonstrates that the critical clique graph is also of crucial importance for our algorithms solving CLP3: our algorithms modify the critical clique graph $\mathrm{CC}(G)$ instead of G.

Lemma 4. *There is always an optimal solution for* CLP3 Edge Editing, CLP3 Edge Deletion, *or* CLP3 Edge Insertion *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.*

Proof. We consider here only CLP3 Edge Editing, but the same argumentation holds for CLP3 Edge Deletion and CLP3 Edge Insertion. Observe that deleting an edge within a critical clique or inserting or deleting some but not all edges between two critical cliques results in splitting one critical clique in G into at least two critical cliques in the resulting graph G', that is, the vertices of a critical clique of G belong to different critical cliques of G'. Thus, if we can construct an optimal solution for CLP3 Edge Editing on G which splits no critical clique of G, then we have shown the lemma. Let F_{opt} be an arbitrary optimal solution for CLP3 Edge Editing on $G = (V, E)$, and $G_{\mathrm{opt}} := (V, E \triangle F_{\mathrm{opt}})$. If F_{opt} splits no critical clique of G, then we are done; otherwise, for a critical clique K of G, there are at least two critical cliques K_1 and K_2 in G_{opt} with $K_1^A := K \cap K_1 \neq \emptyset$ and $K_2^A := K \cap K_2 \neq \emptyset$.

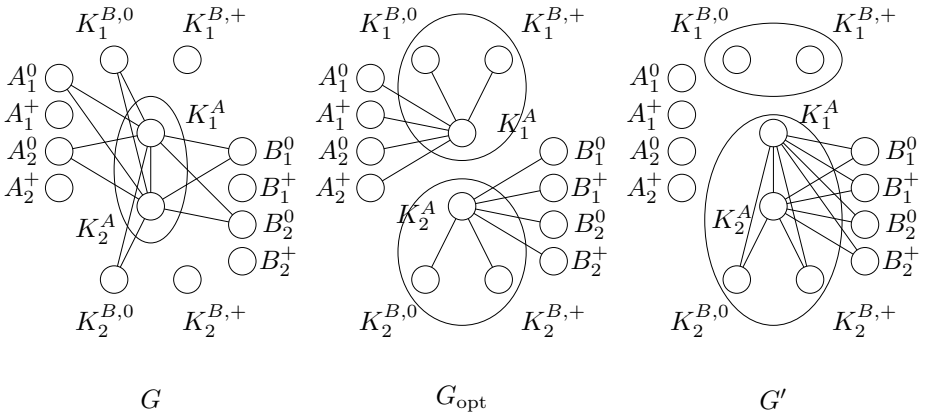

Figure 5: Illustration of the proof of Lemma 4. *Lines* denote pairwise connections between vertex sets. Small *circles* denote vertex sets, *ovals* denote critical cliques. Only connections having one end in K_1^A or K_2^A are displayed.

Let $K_1^B := K_1 \setminus K_1^A$ and $K_2^B := K_2 \setminus K_2^A$. Note that K_1^B and K_2^B can be empty. In the following we show the claim that there exists a solution which does not split K and is at least as good as F_{opt}.

We use $A_1, \ldots, A_r$ to denote the critical cliques in G_{opt} which are adjacent to K_1 but not to K_2 and $B_1, \ldots, B_s$ to denote the critical cliques which are adjacent to K_2 but not to K_1. Furthermore, we partition the vertices of K_1^B into two sets of vertices: $K_1^{B,0}$ whose vertices are adjacent to the vertices of K in G, and $K_1^{B,+} := K_1^B \setminus K_1^{B,0}$. The vertices of K_2^B, $A_1, \ldots, A_r$, and $B_1, \ldots, B_s$ are partitioned in the same way, see Figure 5. (Figure 5 shows the case that there are no edges between K_1 and K_2 in G_{opt}.)

In order to show the claim, we transform at first G_{opt} to a graph G' where K is not split, see Figure 5. Due to the forest structure of $\text{CC}(G_{\text{opt}})$, it is easy to observe that $\text{CC}(G')$ is also a forest and, therefore, G' is a 3-leaf power. Hence, the set of edge modifications F' which transforms G to G' is a solution of CLP3 Edge Editing on G. Observe that the roles of K_1 and K_2 in Figure 5 can be exchanged: we can also apply modifications such that K_2^A, K_1^A, $K_1^{B,0}$, and $K_1^{B,+}$ form a critical clique in G'. Then, by simply counting the neighbors of K_1^A and K_2^A in G and G_{opt} and taking the solution with the smaller number of required edge modifications, we can always find an F' which is at least as good as F_{opt}. This proves the claims. Repeatedly applying the claim to each critical clique in G which is split by F_{opt}, we show the lemma. □

Based on Lemma 4 and Theorem 1, we can now work on $\text{CC}(G)$ instead of G. Note that a modification on $\text{CC}(G)$ can decrease the parameter l by more than one. Then, our algorithm scheme is as follows:

(1) Construct $\text{CC}(G)$ from G.

Originally published in *Algorithmica*, 44(4):363–381. Springer, 2006.

(2) Edit CC(G) to make it a forest.

Note that after modifying CC(G), two or more nodes in CC(G) might obtain identical neighborhoods. Since each node in CC(G) has to represent a critical clique in G, a *merge* operation is needed, which replaces these nodes in CC(G) by a new node with the same neighborhood as the original nodes. Therefore, in the following we assume that after each modification operation, we check for every pair of nodes whether a merge operation between them is possible, which can be done in $O(n \cdot m)$ time.

We now examine the running time of the respective steps. As mentioned in Section 3, CC(G) can be constructed in $O(n \cdot m)$ time. In order to make CC(G) a forest, we have to destroy all cycles in it. A shortest cycle can be found in $O(n \cdot m)$ time [17]. This cycle can be destroyed by either deleting at least one edge of this cycle or triggering a merge operation for two nodes on this cycle. Recall that two nodes can be merged iff they are adjacent and they have the same neighborhood. Thus, in order to merge two nodes K_i and K_j, we have to insert an edge between them if they are not already adjacent; furthermore, we need to delete or insert edges such that K_i and K_j have the same neighborhood. We employ a search tree which finds a shortest cycle and branches into several cases corresponding to each editing operation that destroys it. Note that each case decreases the parameter l by at least one. In the descriptions of the respective algorithms, we now only need to show that the number of the cases corresponding to the editing operations is bounded by a function of l. This implies the fixed-parameter tractability.

As shown in the proof of Theorem 1, if CC(G) has more than one connected component, we can solve the problem for each component independently, and then connect the generated leaf roots by adding a new inner node and connecting it to an arbitrary inner node of each leaf root. This allows us in the following without loss of generality to only consider connected graphs. Note that this property does not hold for Closest k-Phylogenetic Root, which makes it considerably harder to obtain analogous results.

5.1 Edge Deletion

As stated above, the task of Step (2) is to transform CC(G) into a forest by edge deletions. The following lemma enables us to consider only triangles.

Lemma 5. *For a triangle-free critical clique graph* CC(G)*, we can find an optimal solution for* CLP3 Edge Deletion *by finding a maximum weight spanning tree for* CC(G)*, where edges are weighted by the product of the sizes of the critical cliques corresponding to their two endpoints.*

Proof. By Lemma 4, there is an optimal solution which can be described by edge deletions in CC(G). Disconnecting two critical cliques of sizes s_1 and s_2 requires $s_1 \cdot s_2$ edge deletions. Furthermore, since CC(G) contains no triangle, a merge operation between two nodes in CC(G) can only be triggered if the two nodes form an isolated component. However, for a connected CC(G) with more

Originally published in *Algorithmica*, 44(4):363–381. Springer, 2006.

than two nodes, no optimal solution of CLP3 Edge Deletion can produce an isolated component of two nodes. Thus, no merge operation is needed in Step (2). Therefore, a spanning tree with maximum edge weights implies a minimum number of edge deletions to obtain a tree from CC(G). □

With Lemma 5, we can show that each inner node of the search tree has at most six cases to branch.

Theorem 4. CLP3 Edge Deletion *with l edge deletions allowed is fixed-parameter tractable with respect to l.*

Proof. We employ a search tree of height bounded by l. In each inner node of the search tree, we find a triangle in $O(nm)$ time and branch into at most six cases corresponding to the deletions of the three edges of the triangle and the merging of the three pairs of nodes of the triangle. Note that since only edge deletions are allowed, we can trigger a merge between two nodes K_i and K_j only by deleting edges to make all non-common neighbors of K_i and K_j nonadjacent to both of K_i and K_j. At each leaf of the search tree, we find a maximum weight spanning tree in $O(m \log n)$ time. In summary, we have a running time of $O(6^l \cdot nm)$. □

5.2 Edge Insertion

To show that CLP3 Edge Insertion is fixed-parameter tractable, we again use a bounded search tree that modifies CC(G) to be a tree. If CC(G) is already a tree, then no edge insertion is required. For a CC(G) containing at least one cycle, the only possible way to make it a tree by edge insertions is to trigger a merge operation for two nodes on this cycle. Our search tree algorithm tries all possible pairs of nodes to merge in a cycle. To obtain the desired runtime bound, it suffices to determine an upper bound for the length of a cycle in CC(G) that depends only on l. We achieve this by giving a connection between the triangulation of a hole and the merge operations that turn a cycle into a tree.

A *triangulation* of a hole $C = (V_C, E_C)$, where V_C denotes the set of the vertices on this cycle and E_C the set of the edges, is a set D of chords of C such that there is no hole in $C' = (V_C, E_C \cup D)$. A triangulation F of a graph G is *minimal* if no proper subset of F triangulates G.

Lemma 6. *Each set of edges inserted into a cycle C of a critical clique graph to transform C into a tree is a triangulation of C.*

Proof. We prove the lemma by contradiction. Suppose that we have a set of edges D with which cycle $C = (V_C, E_C)$ can be made a tree by merge operations, but there is a hole C' in $C = (V_C, E_C \cup D)$. Since all pairs of the nodes on C' are either not connected by an edge or have at least two neighbors not in common, none of them can be merged with other nodes on C'. This is a contradiction to the assumption that C can be made a tree by inserting the edges in D. □

Kaplan, Shamir, and Tarjan [19] show that a minimal triangulation of an n-cycle consists of $n-3$ chords, which implies that a graph that can be triangulated by at most l edge insertions cannot have a chordless cycle of length more than $l+3$. This is also the key idea of one of their fixed-parameter algorithms for Minimum Fill-In, which is the problem to make a graph chordal by edge insertions. With Lemma 6, we conclude that the maximum cycle length of $\mathrm{CC}(G)$ is bounded above by $l+3$; otherwise, there is no solution to CLP3 Edge Insertion using only l insertion operations.

Altogether, we get the following theorem.

Theorem 5. CLP3 Edge Insertion *on a graph G with l edge insertions allowed is fixed-parameter tractable with respect to l.*

Proof. As already mentioned, we can easily find a shortest cycle (if there is any) in $O(m \cdot n)$ time [17]. If this cycle has length greater than $l+3$, then there is no solution; otherwise, we make an $O((l+3)^2)$-branching: each branch represents the merge of a pair of nodes on this cycle, and there are at most $(l+3)^2$ such node pairs. After each merging of two nodes, there can be at most two new shorter cycles. Since there are at most l edge insertions allowed, such an $O((l+3)^2)$-branching can be made at most l times. We obtain an $O((l+3)^{2l} \cdot nm)$ running time. □

Note that with a modified case distinction, we can improve the branching for getting rid of cycles with length at most $l+3$ to a $(l+3)$-branching such that the overall running time is $O((l+3)^l \cdot nm)$:

Instead of trying all possibilities to merge two nodes of a cycle, we choose an arbitrary node K_i of the cycle. In order to triangulate the cycle, we either have to merge K_i with at least one of the $l+2$ other nodes of the cycle, or we have to merge the two neighbors of K_i. The correctness of this claim is given by the following lemma.

Lemma 7. *Let v be a vertex on a hole C and let u and w be its neighbors in C. Each triangulation of C either connects v to at least one vertex of C, or it connects u and w.*

Proof. Suppose there is a triangulation of C which neither connects v to a vertex of C nor connects u and w.

Let P be the shortest path between u and w after the triangulation that consists only of vertices of C, but does not contain v. Since u and w are not connected by an edge after the triangulation, at least one vertex lies on P between u and w. Because P is the shortest path between u and w and because no vertex of P except u and w is connected with v, there is a hole consisting of v and the vertices of P, which contradicts the assumption that C is a triangulation. □

5.3 Edge Editing

In this section we extend the algorithm for CLP3 Edge Insertion from Section 5.2 to solve CLP3 Edge Editing by additionally taking edge deletions

into account. We distinguish two types of cycles: the *short* cycles having length at most $l+3$, and the *long* cycles having length greater than $l+3$.

We can destroy a short cycle in $\mathrm{CC}(G)$ by deleting at least one edge from it, or by merging some critical cliques. This means we have at most $l+3$ possible edge deletions and at most $(l+3)^2$ possible merge operations. However, merge operations with both edge deletion and edge insertion are more complicated than merge operations with only edge insertion. Suppose that we merge a pair of critical cliques K_i and K_j on a cycle. As with only edge insertions allowed, we insert an edge between K_i and K_j if they are not adjacent. There may be some critical cliques which are neighbors of K_i but not of K_j or vice versa. To satisfy the neighborhood condition of a critical clique, for each of these neighbors which are not common to K_i and K_j, we have to either insert at least one edge to make it a common neighbor of both critical cliques, or delete at least one edge to make it nonadjacent to both critical cliques. However, there may be at most l such non-common neighbors, since there are at most l edge editing operations allowed. A merge operation between K_i and K_j is then possible only if they have at most l non-common neighbors. Thus, we have at most 2^l different ways to merge these two critical cliques. Altogether, we now have $(l+3)+(l+3)^2 \cdot 2^l$ branchings to transform a short cycle into a tree.

For long cycles, with the discussion in Section 5.2, the following observation is easy to prove:

Observation 1. *A long cycle in* $\mathrm{CC}(G)$ *cannot be transformed into a tree solely by edge insertions.*

Therefore, we have to delete at least one edge on a long cycle. Furthermore, we can show that for such long cycles no edge insertion is necessary.

Lemma 8. *In order to solve* CLP3 Edge Editing *on a graph with only long cycles, there is no need to insert any edges.*

Proof. Suppose that we have an optimal solution F, that is, a set of some edge editing operations, for a graph G with only long cycles, and that there is at least one edge insertion in F. With Observation 1, we know that F also contains at least one edge deletion. We perform only the edge deletions from F in G. If the resulting graph has no cycles, then there is a contradiction to the optimality of F; otherwise, the cycles in the resulting graph are long cycles. By Observation 1, we cannot make the resulting graph a tree solely by edge insertions, which is a contradiction to the fact that F is a solution. Then, there is no edge insertion in F. □

Theorem 6. CLP3 Edge Editing *on a graph with* l *edge editing operations allowed is fixed-parameter tractable with respect to* l*.*

Proof. As already stated in the introduction of Section 5, the algorithm uses the $l+3+(l+3)^2 \cdot 2^l$-branching to get all solutions which destroy the short cycles by at most l edge editing operations. Finally, a maximum weight spanning forest is constructed for each resulting graph which has no short cycle. □

Again, with a slight modification of the algorithm we can improve the branching for getting rid of short cycles. More precisely, we get a $(l+3+l+5)$-branching such that the overall running time is $O((2l+8)^l \cdot nm)$.

First, we use the same idea as in Section 5.2: Instead of considering every pair of nodes of a short cycle for a merging operation, we choose an arbitrary node of the cycle and try the $l+3$ possibilities to merge this node with another node of the cycle or to merge its two neighbors on the cycle.

The second idea is rather a refined analysis than a modification of the algorithm: Instead of branching into up to 2^l cases for merging two nodes K_i, K_j (and decreasing the parameter l by the number of their non-common neighbors), we only connect K_i and K_j by an edge (without merging them immediately) if they are not already connected. If, however, K_i and K_j are adjacent, we consider one non-common neighbor of K_i and K_j. In order to merge K_i and K_j, we have to either insert or delete an edge between this neighbor and one of K_i and K_j.

Altogether, in order to destroy a short cycle we have at most $l+3$ possibilities to delete an edge of the cycle and at most $l+5$ possibilities to modify an edge in order to merge two nodes of the cycle.

5.4 Vertex Deletion

We first show that the property from Lemma 4 holds analogously for the vertex deletion problem.

Lemma 9. *All optimal solutions for* CLP3 Vertex Deletion *can be represented by node deletions on* $\mathrm{CC}(G)$. *That is, if one vertex in a critical clique is deleted, then all vertices in the critical clique are deleted.*

Proof. This lemma is proven by contradiction. Assume that we have an optimal solution $F_{\mathrm{opt}} \subseteq V$ on $G = (V, E)$ that deletes some but not all vertices from a critical clique K in G. Let G' be G modified by F_{opt}, that is, $G' = (V', E')$ where $V' := V \setminus F_{\mathrm{opt}}$ and $E' := E \cap (V' \times V')$. We construct G'' from G' by putting back a deleted vertex from K. That is, we choose $v \in (K \setminus V')$ and $u \in (K \cap V')$, which exist by the assumption, and define $G'' = (V'', E'')$ where $V'' := V' \cup \{v\}$ and $E'' := E \cap (V'' \times V'')$. Since two vertices having the same neighbors in G have also the same neighbors in any induced subgraph of G, u and v must belong to the same critical clique in G''. Thus, $\mathrm{CC}(G'')$ has the same structure as $\mathrm{CC}(G')$, i.e., it is a forest, which implies that $F' := F_{\mathrm{opt}} \setminus \{v\}$ is a solution. This is a contradiction to the optimality of F_{opt}. □

We use a similar approach to solve CLP3 Vertex Deletion as for the edge deletion variant: For an input graph G, if its critical clique graph $\mathrm{CC}(G)$ is a forest, then we can easily construct a 3-leaf root as described in Section 3. Otherwise, $\mathrm{CC}(G)$ contains some cycles. Due to Lemma 9, for each cycle in $\mathrm{CC}(G)$, we have to delete at least one critical clique on this cycle completely, that is, delete all vertices in this critical clique from G. As in the edge deletion case, the search tree algorithm first considers the triangles in $\mathrm{CC}(G)$, for which there

Originally published in *Algorithmica*, 44(4):363–381. Springer, 2006.

are clearly only six cases to branch. Furthermore, solely with vertex deletions, we cannot create any triangle in a triangle-free CC(G). Hence, if CC(G) is triangle-free, a merge operation can only occur for two critical cliques which form an isolated component, and can be done after we convert CC(G) into a forest. If we give each critical clique a weight equal to its size, the problem to transform a triangle-free CC(G) into a forest by node deletions can be stated as a weighted version of the Undirected Feedback Vertex Set problem, which seeks in an undirected graph for a set of at most l vertices whose removal destroys all cycles in the graph. Here we want a set of vertices with the sum of weights at most l. Raman et al. [33] give a fixed-parameter algorithm solving the unweighted version, which is based on the following classical result of Erdős and Pósa [12].

Lemma 10. *Any graph with minimum degree at least* 3 *has a cycle of length at most* $2 \log n + 1$.

While in the unweighted case we can simply ignore the degree-2 nodes, a degree-2 node with low weight can be relevant in the weighted case. However, we can apply the following data reduction rule to CC(G): We omit from two consecutive degree-2 nodes the one with higher weight, and connect its two neighbors. This data reduction can be clearly done in linear time. Then between two nodes with degree at least 3 there is at most one degree-2 node. By Lemma 10, there is a cycle in CC(G) of length at most $4 \log n + 2$. We then use the greedy strategy described by Raman et al. [33] to destroy this cycle, that is, we make an $O(4 \log n)$-branching; each branch represents the deletion of one node on this cycle. Since this branching can be done at most l times, we obtain a bounded search tree of size $O((4 \log n)^l)$. Since $(\log n)^l \leq (3l \log l)^l + n$ for all n and l, we have then a fixed-parameter algorithm for the weighted case.

Theorem 7. CLP3 Vertex Deletion *with l vertex deletions is fixed-parameter tractable with respect to l.*

6 Concluding Remarks

Our algorithmic results fall into the broad category of complexity of graph modification problems [3, 24, 28, 29].

The line of research initiated in our work offers several future challenges. We only mention four points.

- It remains open to provide a non-trivial reduction to a problem kernel [14, 30] for Closest 3-Leaf Power.
- Also open is the problem of finding good polynomial-time approximation algorithms for Closest 3-Leaf Power.
- One challenge is to investigate whether similar fixed-parameter tractability results can be achieved for the closely related phylogenetic root problems

studied in [5, 25]. Forbidding degree-2 nodes there in the output trees seems to make things more elusive, though.

- From a more applied point of view, it would be interesting to see how small the combinatorial explosion in the number of modifications can be made for CLP3 and its variants. Encouraging results for the "simpler" but still NP-complete Closest 2-Leaf Power problem are obtained in [15, 16] (where the problem is referred to as Cluster Editing).

  Moreover, our algorithms for the edge editing, edge insertion, and vertex deletion variants require to know the number of modifications l in advance, whereas the algorithm for the edge deletion variant finds an optimal solution even without this knowledge. It would be interesting to see whether we can find fixed-parameter algorithms for the first three variants with this desirable property.

We recently obtained similar positive results for the most difficult k-leaf power problem with known polynomial-time solvable recognition problem: CLP4. More specifically, we found a forbidden subgraph characterization for 4-leaf powers and fixed-parameter algorithms for CLP4 [9]. As long as it remains open to determine the complexity of k-Leaf Power recognition for $k > 4$, it seems to make little sense to study the more general Closest k-Leaf Power for $k > 4$.

Acknowledgements. The authors would like to thank the anonymous referees for their valuable comments and suggestions.

References

[1] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56(1):89–113, 2004. 5

[2] A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph Classes: a Survey*. SIAM Monographs on Discrete Mathematics and Applications, 1999. 1, 4

[3] L. Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58:171–176, 1996. 3, 6, 18

[4] M. Charikar, V. Guruswami, and A. Wirth. Clustering with qualitative information. In *Proc. 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS '03)*, pages 524–533. IEEE Computer Society Press, Los Alamitos, CA, 2003. 5

[5] 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. 2, 3, 5, 9, 10, 19

[6] Z.-Z. Chen and T. Tsukiji. Computing bounded-degree phylogenetic roots of disconnected graphs. In *Proc. 30th International Workshop on Graph-Theoretic Concepts in Computer Science (WG '04)*, volume 3353 of *LNCS*, pages 308–319. Springer-Verlag, Berlin, 2004. Long version to appear in *Journal of Algorithms*. 2

[7] M. Dom. Fehler-Korrektur bei Leaf-Root-Problemen (error compensation in leaf root problems). Diploma thesis (in German), Universität Tübingen, 2004. 3

[8] M. Dom, J. Guo, F. Hüffner, and R. Niedermeier. Error compensation in leaf root problems. In *Proc. 15th Annual International Symposium on Algorithms and Computation (ISAAC '04)*, volume 3341 of *LNCS*, pages 389–401. Springer-Verlag, Berlin, 2004. 1, 3

[9] M. Dom, J. Guo, F. Hüffner, and R. Niedermeier. Extending the tractability border for closest leaf powers. Manuscript, Institut für Informatik, Friedrich-Schiller-Universität Jena, Mar. 2005. 3, 19

[10] R. G. Downey. Parameterized complexity for the skeptic. In *Proc. 18th IEEE Annual Conference on Computational Complexity (CCC '03)*, pages 147–169, 2003. 5

[11] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer-Verlag, Berlin, 1999. 5

[12] P. Erdős and L. Pósa. On the maximal number of disjoint circuits of a graph. *Publicationes Mathematicae Debrecen*, 9:3–12, 1962. 18

[13] M. R. Fellows. Blow-ups, win/win's, and crown rules: Some new directions in FPT. In *Proc. 29th International Workshop on Graph-Theoretic Concepts in Computer Science (WG '03)*, volume 2880 of *LNCS*, pages 1–12. Springer-Verlag, Berlin, 2003. 5

[14] M. R. Fellows. New directions and new challenges in algorithm design and complexity, parameterized. In *Proc. 8th International Workshop on Algorithms and Data Structures (WADS '03)*, volume 2748 of *LNCS*, pages 505–520. Springer-Verlag, Berlin, 2003. 5, 18

[15] J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Graph-modeled data clustering: Fixed-parameter algorithms for clique generation. In *Proc. 5th Italian Conference on Algorithms and Complexity (CIAC '03)*, volume 2653 of *LNCS*, pages 108–119. Springer-Verlag, Berlin, 2003. Long version to appear in *Theory of Computing Systems*. 3, 19

[16] J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Automated generation of search tree algorithms for hard graph modification problems. *Algorithmica*, 39(4):321–347, 2004. 19

[17] A. Itai and M. Rodeh. Finding a minimum circuit in a graph. *SIAM Journal on Computing*, 7(4):413–423, 1978. 13, 15

[18] T. Jiang, G. Lin, and J. Xu. On the closest tree kth root problem. Manuscript, Department of Computer Science, University of Waterloo, 2000. 3

[19] H. Kaplan, R. Shamir, and R. E. Tarjan. Tractability of parameterized completion problems on chordal, strongly chordal, and proper interval graphs. *SIAM Journal on Computing*, 28(5):1906–1922, 1999. 15

[20] P. E. Kearney and D. G. Corneil. Tree powers. *Journal of Algorithms*, 29(1):111–131, 1998. 2, 3

[21] M. Křivánek and J. Morávek. NP-hard problems in hierarchical-tree clustering. *Acta Informatica*, 23(3):311–323, 1986. 5, 9

[22] L. C. Lau. Bipartite roots of graphs. In *Proc. 15th ACM-SIAM Symposium on Discrete Algorithms (SODA '04)*, pages 952–961. ACM/SIAM, 2004. 2

[23] 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. 2

[24] J. M. Lewis and M. Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, 20(2):219–230, 1980. 9, 11, 18

[25] G. Lin, P. E. Kearney, and T. Jiang. Phylogenetic k-root and Steiner k-root. In *Proc. 11th Annual International Symposium on Algorithms and Computation (ISAAC '00)*, volume 1969 of *LNCS*, pages 539–551. Springer-Verlag, Berlin, 2000. 2, 5, 6, 8, 11, 19

[26] Y. L. Lin and S. S. Skiena. Algorithms for square roots of graphs. *SIAM Journal on Discrete Mathematics*, 8(1):99–118, 1995. 2

[27] R. Motwani and M. Sudan. Computing roots of graphs is hard. *Discrete Applied Mathematics*, 54(1):81–88, 1994. 2

[28] A. Natanzon. Complexity and approximation of some graph modification problems. Master's thesis, Department of Computer Science, Tel Aviv University, 1999. 5, 9, 18

[29] A. Natanzon, R. Shamir, and R. Sharan. Complexity classification of some edge modification problems. *Discrete Applied Mathematics*, 113:109–128, 2001. 18

[30] R. Niedermeier. Ubiquitous parameterization—invitation to fixed-parameter algorithms. In *Proc. 29th International Symposium on Mathematical Foundations of Computer Science (MFCS '04)*, volume 3153 of *LNCS*, pages 84–103. Springer-Verlag, Berlin, 2004. 5, 18

[31] N. Nishimura, P. Ragde, and D. M. Thilikos. On graph powers for leaf-labeled trees. *Journal of Algorithms*, 42(1):69–108, 2002. 2, 4

[32] R. Peeters. The maximum edge biclique problem is NP-complete. *Discrete Applied Mathematics*, 131(3):651–654, 2003. 10

[33] V. Raman, S. Saurabh, and C. R. Subramanian. Faster fixed parameter tractable algorithms for undirected feedback vertex set. In *Proc. 13th Annual International Symposium on Algorithms and Computation (ISAAC '02)*, volume 2518 of *LNCS*, pages 241–248. Springer-Verlag, Berlin, 2002. 18

[34] R. Shamir, R. Sharan, and D. Tsur. Cluster graph modification problems. *Discrete Applied Mathematics*, 144(1–2):173–182, 2004. 3, 5, 6

[35] T. Tsukiji and Z.-Z. Chen. Computing phylogenetic roots with bounded degrees and errors is hard. In *Proc. 10th International Computing and Combinatorics Conference (COCOON '04)*, volume 3106 of *LNCS*, pages 450–461. Springer-Verlag, Berlin, 2004. 3, 5