# Error Compensation in Leaf Root Problems*

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

Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Sand 13,
D-72076 Tübingen, Fed. Rep. of Germany
{dom,guo,hueffner,niedermr}@informatik.uni-tuebingen.de

**Abstract.** The $k$-Leaf Root problem is a particular case of graph power problems. Here, we study "error correction" versions of $k$-Leaf Root—that is, for instance, 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 Root problem (the error correction version of 3-Leaf Root) is fixed-parameter tractable with respect to the number of edge modifications in the given graph. Thus, we provide the seemingly first nontrivial positive algorithmic results in the field of error compensation for leaf root 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.

## 1 Introduction

Graph powers are a classical concept in graph theory (cf. [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 of length at most $k$ between $u$ and $v$ in $G$. We say $G$ is the *$k$-root* of $G^k$; deciding whether a graph is a power of some other graph is called the *graph root* problem. It is NP-complete in general [18], but 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]. Very recently, Lau [14] 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 [15] 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 [12] study the problem where every tree node one-to-one corresponds to a graph vertex, Nishimura, Ragde, and Thilikos [21] introduced the notion of *leaf powers* where exclusively the tree leaves stand in one-to-one correspondence to the graph vertices. Motivated by applications in computational biology, Lin, Kearney, and Jiang [16] and Chen,

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

Jiang, and Lin [4] 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 a $k$-root are called $k$-Leaf Root [21] and $k$-Phylogenetic Root [16], respectively. For $k \leq 4$, both problems are solvable in polynomial time [21,16]. The complexities of both problems for $k \geq 5$ are still open. Moreover, Chen et al. [4] 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 $k$-phylogeny 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 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 [12], and they introduced 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 $k \geq 2$ [12,10]. One also obtains NP-completeness for the corresponding problems Closest $k$-Phylogenetic Root [4] and, as we point out here, Closest $k$-Leaf Root. In addition, for Closest $k$-Leaf Root we study other edge modification problems—namely, only to allow edge deletions or only to allow edge insertions—and we show NP-completeness. See Table 1 in Section 4 for an overview concerning complexity results for Closest $k$-Leaf Root and its variants.

To the best of our knowledge, the above error correction scenario so far only led to results showing hardness of complexity. We are not aware of any results concerning approximation or non-trivial 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 Root 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 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 [23]. This characterization finds direct applications in corresponding fixed-parameter algorithms [7] (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. Due to the lack of space, many proofs are deferred to the full version of this paper.

## 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 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* $A \setminus B \cup B \setminus A$.

Closely related to the well-known graph power concept (cf. [2, Section 10.6]) is the notion of a *$k$-leaf power* of a tree, introduced by Nishimura, Ragde, and Thilikos [21]:

**Definition 1.** *Given 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 $E := \{(u, v) \mid u, v \in V$ and $d_T(u, v) \leq k\}$.*

The following problem is inspired by the problem of forming a phylogenetic tree[1] based on a binary similarity measure.

$k$-Leaf Root (LR$k$)

**Instance:** A graph $G$.

**Question:** Is there a tree $T$ such that $T^k = G$?

Nishimura et al. [21] show that $k$-Leaf Root can be solved in polynomial time for $k \leq 4$. As already Nishimura et al. point out, in practice phylogenetic problems involve errors in distance estimators. This motivates the following.

Closest $k$-Leaf Root (CLR$k$)

**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, $|E(T^k) \triangle E(G)| \leq l$?

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

- CLR$k$ Edge Insertion: Only inserting edges into $G$ is allowed to obtain $T^k$;
- CLR$k$ Edge Deletion: Only deleting edges from $G$ is allowed to obtain $T^k$.

CLR2 Edge Editing has been studied under various names in the literature. The first proof of its NP-completeness is due to Křivánek and Morávek [13], where it is called Hierarchical-Tree Clustering. Independently, the problem was studied by Shamir, Sharan, and Tsur as Cluster Editing [23] and by Bansal, Blum, and Chawla as Correlation Clustering [1]. CLR2 Edge Deletion (also known as Cluster Deletion) was shown to be NP-complete by Natanzon [19].

Lin, Kearney, and Jiang [16] consider a variant of $k$-Leaf Root where the inner nodes of the output tree are not allowed to have degree 2. They call this problem $k$-Phylogenetic Root (PR$k$), and show that PR$k$ can be solved in

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

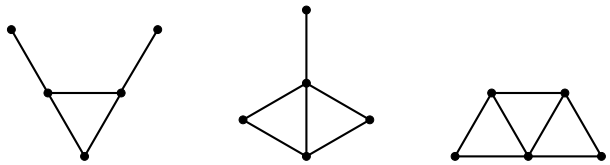


**Fig. 1.** 5-vertex graphs that occur as forbidden induced subgraphs

linear time for $k \leq 4$.[2] 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 (CPR$k$), defined analogous to Closest $k$-Leaf Root. CPR$k$ is examined by Chen, Jiang, and Lin [4], who show that it is NP-complete for $k \geq 2$.

Among other things, we show that CLR3 is *fixed-parameter tractable* with respect to parameter $l$. That is, we show that CLR3 can be solved in $f(l) \cdot n^{O(1)}$ time, where $f$ is an (exponential) function only depending on $l$. For small $l$, as might be naturally expected since $l$ refers to the number of errors, efficient (polynomial-time) algorithms are possible. Two recent surveys on fixed-parameter tractability can be found in [6,20].

## 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. [23] 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 1).

Forbidden subgraph characterizations can be valuable in various ways. For instance, 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. [16].

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

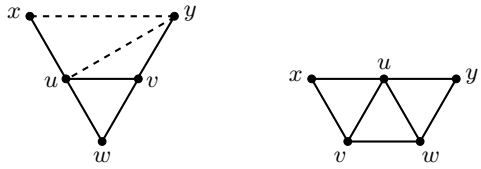


**Fig. 2.** Neighborhood of 3 vertices $u, v, w$ from 3 different critical cliques

**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.*

In other words, a critical clique is a module that is maximal and a clique. 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 $G$ consists of at most two critical cliques.*

**Lemma 2.** *For a chordal graph $G$, the following are equivalent:*

*(1) There is a clique $K$ in $G$ that consists of at least three critical cliques.*
*(2) $G$ contains a bull, dart, or gem (see Figure 1) as induced subgraph.*

*Proof.* **(1) ⇒ (2):** Let $u, v, w$ be three vertices from $K$ that belong to different critical cliques. We distinguish two cases.

(a) There is a vertex $x$ which is connected to exactly one of $u, v, w$, say to $u$ (see Figure 2 (a)). Since $v$ and $w$ belong to different critical cliques, there must be a vertex $y$ which is connected to only one of them, say to $v$. The edges $(y, u)$ and $(y, x)$ can be present or not, except that if $(y, x)$ is present, then $(y, u)$ must also be present, because otherwise $x, y, v, u$ induce a hole. This leaves 3 possibilities, where we get the induced subgraphs bull, dart, and gem, respectively.

(b) There is no vertex which is connected to exactly one of $u, v, w$ (see Figure 2 (b)). Then there is a vertex $x$ which is connected to exactly two of $u, v, w$, say to $u$ and $v$ (otherwise, $u, v, w$ would have identical neighborhood, and would be in the same critical clique). Since $u$ and $v$ belong to different critical cliques, there is a vertex $y$ which is adjacent to only one of them, say to $u$. By the precondition of this case, $y$ is connected to $w$. The vertices $x$ and $y$ cannot be connected, since otherwise $x, y, w, v$ induce a hole. We get an induced gem.

**(2) ⇒ (1):** Assume $G$ 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 3 different critical cliques. □

Since between the vertices of two critical cliques either all pairwise or no connections are present, the concept of a *critical clique graph* [16] 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.

**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.*

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 lemma reveals details of the structure of a critical clique graph.

**Lemma 3.** *If every clique of a graph $G$ consists of at most two critical cliques, then $\mathrm{CC}(G)$ does not contain a clique of three or more vertices.*

Utilizing Lemmas 1, 2, and 3, 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) $G$ is chordal and contains no bull, dart, or gem as induced subgraph.*

*Proof.* **(1) ⇒ (2):** If $G$ is a leaf power, then $G$ must be chordal [16]. Then, by Lemma 1 and Lemma 2, it does not contain any of the forbidden subgraphs.

**(2) ⇒ (1):** 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 3, it follows that $\mathrm{CC}(G)$ is a forest. For each connected component of $\mathrm{CC}(G)$, 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. □

**Table 1.** Complexity of Closest $k$-Leaf Root. The polynomial-time solvability of CLR2 Edge Insertion is trivial; the results for $k \geq 3$ are discussed in Section 4. The main new result is NP-completeness of CLR$k$ Edge Insertion for $k \geq 3$.

| | $k = 2$ | $k \geq 3$ |
|---|---|---|
| Edge editing | NP-complete [13] | NP-complete |
| Edge deletion | NP-complete [19] | NP-complete |
| Edge insertion | P | NP-complete |

## 4 NP-Completeness Results

In Table 1 we summarize known and new results on the classical complexity (P vs. NP) of Closest $k$-Leaf Root problems. The NP-completeness of CLR$k$ Edge Editing and CLR$k$ Edge Deletion for $k \geq 3$ can be shown by an adaption of the NP-completeness proof for CPR$k$ by Chen et al. [4]. We refer to the full version of this paper for details on how to adapt their proof.

CLR2 Edge Insertion can be trivially solved in polynomial time, since it is exactly the problem of adding edges to a graph so that each connected component becomes a clique. However, it can be shown that for $k \geq 3$ this problem becomes NP-complete by giving a reduction from Maximum Edge Biclique [22]; we defer the proof to the full version of this paper.

**Theorem 2.** CLR$k$ Edge Insertion *is NP-complete for* $k \geq 3$.

## 5 Fixed-Parameter Tractability Results for CLR3

In this section we show fixed-parameter tractability with respect to the number of editing operations $l$ for CLR3 Edge Insertion, CLR3 Edge Deletion, and CLR3 Edge Editing. 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. [16] 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 CLR3: our algorithms work with the critical clique graph $\mathrm{CC}(G)$ instead of $G$. We defer the proof of the lemma to the full version of this paper since it is similar to the proof of Lemma 2. We use $C_4$ to denote a chordless cycle of four vertices.

**Lemma 4.** *If a graph $G$ contains no bull, dart, gem, or $C_4$ as induced subgraph, then there is no triangle in its critical clique graph.*

Following from Lemma 4, if we can show that there is an optimal solution for CLR3 problems which can be represented as editing operations on the critical clique graph, then the two above tasks can be fulfilled in two independent phases: First, eliminate each induced bull, dart, gem, and $C_4$ in the critical clique graph using a search tree of bounded depth. Then, edit each of the resulting critical clique graphs to make it a forest. The optimal solution is then the solution with minimum total number of editing operations in the two steps. The following two lemmas show that it is indeed correct to work only with $\mathrm{CC}(G)$ instead of $G$.

**Lemma 5.** *Graph $G$ contains a bull, dart, gem, or $C_4$ iff its critical clique graph $\mathrm{CC}(G)$ contains a bull, dart, gem, or $C_4$, respectively.*

**Lemma 6.** *Given a graph $G$. Then there is always an optimal solution for* CLR3 Edge Editing *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.*

Based on Lemmas 5 and 6, we can now consider a critical clique of $G$ as a single vertex and work on $\mathrm{CC}(G)$ instead of $G$. One more benefit of this approach is that we can eliminate several forbidden subgraphs in $G$ by only one modification operation in $\mathrm{CC}(G)$. A modification operation on $\mathrm{CC}(G)$ can decrease the parameter $l$ by more than one since it can correspond to more than one modification operation on $G$. Then, our algorithm scheme is as follows:

(0) Construct $\mathrm{CC}(G)$ from $G$.
(1) Edit $\mathrm{CC}(G)$ to get rid of the forbidden subgraphs bull, dart, gem, and $C_4$.
(2) Edit $\mathrm{CC}(G)$ to make it a forest.

Note that after modifying $\mathrm{CC}(G)$, two or more nodes in $\mathrm{CC}(G)$ might obtain identical neighborhoods. Since each node in $\mathrm{CC}(G)$ has to represent a critical clique in $G$, 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. 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, $\mathrm{CC}(G)$ can be constructed in $O(n \cdot m)$ time. By Cai's result [3], there is a fixed-parameter algorithm for Step (1). More specifically, because of Lemma 4, we know that each triangle in $\mathrm{CC}(G)$ is either part of a bull, dart, or gem, or it has one edge in common with a $C_4$. We can then determine a forbidden subgraph by first finding a triangle in $O(n \cdot m)$ time, and then partitioning the $n - 3$ nodes not in the triangle into eight sets depending on to which of the three nodes in the triangle they are connected. This allows to find a bull, dart, gem, or $C_4$ in additional $O(n)$ time. If we do not find a triangle, we can find a $C_4$ by determining the shortest cycle in $O(n \cdot m)$ time [9]. In summary, we find a forbidden subgraph in $O(n \cdot m)$ time.[3]

[3] Note that using algorithms based on matrix multiplication, we can alternatively do this in $O(n^{2.38})$ time [5].

Therefore, we can employ a search tree which finds a forbidden subgraph and branches into several cases corresponding to each editing operation that destroys it. As an example, for edge deletion, this will lead to an $O(7^l \cdot nm)$ time algorithm, since the highest number of edges occurring in any forbidden subgraph is seven (gem). We mention in passing that the techniques applied by Gramm et al. [7,8] for CLR2 could be adapted to improve the base 7 of the exponential component of the running time. In the descriptions of the respective algorithms, we now only need to deal with Step (2) to show fixed-parameter tractability.

As shown in the proof of Theorem 1, if $\mathrm{CC}(G)$ 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 a bull-, dart-, gem-, and $C_4$-free $\mathrm{CC}(G)$, which does not contain a triangle as an induced subgraph, into a forest by edge deletions. Observe that after getting rid of all forbidden subgraphs in Step (1), throughout Step (2) $\mathrm{CC}(G)$ always remains bull-, dart-, gem-, and $C_4$-free when only edge deletions are allowed. Hence, Step (2) of our algorithm scheme can be handled with a polynomial-time algorithm, as stated in the following lemma.

**Lemma 7.** *Given a critical clique graph* $\mathrm{CC}(G)$ *that contains no induced bull, dart, gem, or* $C_4$*. Then we can find an optimal solution for* CLR3 Edge Deletion *by finding a maximum weight spanning tree for* $\mathrm{CC}(G)$*, where edges are weighted by the product of the sizes of the critical cliques corresponding to their two endpoints.*

**Theorem 3.** CLR3 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 forbidden subgraph and branch into at most seven cases corresponding to the edges of the forbidden subgraph. 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 $O(7^l \cdot nm)$. □

### 5.2 Edge Insertion

If $\mathrm{CC}(G)$ after Step (1) contains no cycle, i.e., it is a tree, then there is no edge insertion required. For a $\mathrm{CC}(G)$ containing at least one cycle, the only possible way to make it a tree by edge insertions is to trigger a merge operation for

some nodes on this cycle such that the remaining nodes induce no 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 connect $K_i$ to all neighbors of $K_j$ and connect $K_j$ to all neighbors of $K_i$. Since each cycle of $\mathrm{CC}(G)$ has length at least four, two nodes $K_i$ and $K_j$ on the same cycle either are not adjacent or there are at least two neighbors which are not common to $K_i$ and $K_j$. Hence, we need at least one edge insertion to merge two nodes on a cycle.

We show the fixed-parameter tractability of CLR3 Edge Insertion with respect to $l$ by giving a simple search tree algorithm that tries all possible pairs of nodes to merge in a cycle. For this, it suffices to determine an upper bound for the length of a cycle in $\mathrm{CC}(G)$ 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 8.** *Each set of edges inserted into a cycle $C$ of a critical clique graph to transform $C$ into a tree is a triangulation of $C$.*

Kaplan, Shamir, and Tarjan [11] show that a minimal triangulation $D$ of an $n$-cycle $C$ consists of $n - 3$ chords, which implies that a graph $G$ 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 insertion. With Lemma 8, we conclude that the maximum cycle length of $\mathrm{CC}(G)$ is bounded above by $l + 3$; otherwise, there is no solution to CLR3 Edge Insertion using only $l$ insertion operations.

Altogether, we get the following theorem.

**Theorem 4.** CLR3 Edge Insertion *on a graph $G = (V, E)$ with $l$ edge insertions allowed is fixed-parameter tractable with respect to $l$.*

### 5.3 Edge Editing

In this section we extend the algorithm for CLR3 Edge Insertion from Section 5.2 to solve CLR3 Edge Editing by additionally taking edge deletions into account. We distinguish two types of cycles: the *long* cycles having length greater than $l + 3$, and the *short* cycles having length at most $l + 3$.

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 an edge to make it a common neighbor of both critical cliques, or delete an edge to make it nonadjacent to both critical cliques. However, there may be at most $l$ such noncommon 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 $l$ noncommon 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.

**Theorem 5.** CLR3 Edge Editing *on a graph $G = (V, E)$ with $l$ edge editing operations allowed is fixed-parameter tractable with respect to $l$.*

## 6 Concluding Remarks

Our algorithmic results fall into the broad category of complexity for graph modification problems. In addition, we recently obtained a fixed-parameter tractability result for CLR3 Vertex Deletion, the NP-complete problem that asks for the least number of vertices to delete to make a graph a 3-leaf root. The line of research initiated in our work offers several future challenges. We only mention four points.

- In ongoing work we examine the generalization of our fixed-parameter tractability results for Closest 3-Leaf Root to Closest 4-Leaf Root. For $k \geq 5$ the question is completely open. The difficulty here lies in the more complicated structure of the critical clique graph; for example, it is no longer required to be a tree.
- It remains open to provide a problem kernel for 3-Leaf Root [6,20].
- One challenge is to investigate whether similar fixed-parameter tractability results can be achieved for the closely related phylogenetic root problems studied in [4,16]. 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 for CLR3 and its variants in the parameter $l$ (denoting the number of modifications) can be made. Encouraging results for the "simpler" but still NP-complete Closest 2-Leaf Root problem are obtained in [7,8] (where the problem is referred to as Cluster Editing).

## References


1. N. Bansal, A. Blum, and S. Chawla. Correlation clustering. In *Proc. 43rd FOCS*, pages 238–247. IEEE Computer Society, 2002. 3
2. A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph Classes: a Survey.* SIAM Monographs on Discrete Mathematics and Applications, 1999. 1, 3

3. L. Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58:171–176, 1996. 2, 4, 8
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. 2, 4, 7, 11
5. D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9:251–280, 1990. 8
6. M. R. Fellows. New directions and new challenges in algorithm design and complexity, parameterized. In *Proc. 8th WADS*, volume 2748 of *LNCS*, pages 505–520. Springer, 2003. 4, 11
7. J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Graph-modeled data clustering: Fixed-parameter algorithms for clique generation. In *Proc. 5th CIAC*, volume 2653 of *LNCS*, pages 108–119. Springer, 2003. Long version to appear in *Theory of Computing Systems*. 2, 9, 11
8. 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. 9, 11
9. A. Itai and M. Rodeh. Finding a minimum circuit in a graph. *SIAM Journal on Computing*, 7(4):413–423, 1978. 8
10. T. Jiang, G. Lin, and J. Xu. On the closest tree $k$th root problem. Manuscript, Department of Computer Science, University of Waterloo, 2000. 2
11. 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. 10
12. P. E. Kearney and D. G. Corneil. Tree powers. *Journal of Algorithms*, 29(1):111–131, 1998. 1, 2
13. M. Křivánek and J. Morávek. NP-hard problems in hierarchical-tree clustering. *Acta Informatica*, 23(3):311–323, 1986. 3, 7
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, 2, 3, 4, 6, 7, 11
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, 7
20. R. Niedermeier. Ubiquitous parameterization—invitation to fixed-parameter algorithms. In *Proc. 29th MFCS*, volume 3153 of *LNCS*, pages 84–103. Springer, 2004. 4, 11
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. R. Peeters. The maximum edge biclique problem is NP-complete. *Discrete Applied Mathematics*, 131(3):651–654, 2003. 7
23. R. Shamir, R. Sharan, and D. Tsur. Cluster graph modification problems. In *Proc. 28th WG*, volume 2573 of *LNCS*, pages 379–390. Springer, 2002. Long version to appear in *Discrete Applied Mathematics*. 2, 3, 4