To appear in *Algorithmica*

A Cubic-Vertex Kernel for Flip Consensus Tree

Christian Komusiewicz · Johannes Uhlmann

Received: date / Accepted: date

Abstract Given a bipartite graph $G = (V_c, V_t, E)$ and a nonnegative integer k, the NP-complete Minimum-Flip Consensus Tree problem asks whether G can be transformed, using up to k edge insertions and deletions, into a graph that does not contain an induced P_5 with its first vertex in V_t (a so-called M-graph or Σ-graph). This problem plays an important role in computational phylogenetics, V_c standing for the characters and V_t standing for taxa. Chen et al. [IEEE/ACM TCBB 2006] showed that Minimum-Flip Consensus Tree is NP-complete and presented a parameterized algorithm with running time $O(6^k \cdot |V_t| \cdot |V_c|)$. Subsequently, Böcker et al. [ACM TALG, 2012] presented a refined search tree algorithm with running time $O(4.42^k(|V_t| + |V_c|) + |V_t| \cdot |V_c|)$. We continue the study of Minimum-Flip Consensus Tree parameterized by k. Our main contribution are polynomial-time executable data reduction rules yielding a problem kernel with $O(k^3)$ vertices. In addition, we present an improved search tree algorithm with running time $O(3.68^k \cdot |V_c|^2|V_t|)$.

Keywords computational phylogenetics · perfect phylogeny · NP-hard problem · edge-modification problem · fixed-parameter tractability · polynomial-time preprocessing

A preliminary version appeared in Proceedings of the IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2008), volume 2 in Leibniz International Proceedings in Informatics (LIPIcs), pages 280–291, Schloss Dagstuhl—Leibniz-Zentrum für Informatik, 2008.

Supported by a PhD fellowship of the Carl-Zeiss-Stiftung and the DFG, research project PABI, NI 369/7.

Supported by the DFG, research project PABI, NI 369/7.

C. Komusiewicz and J. Uhlmann
Institut für Softwaretechnik und Theoretische Informatik, TU Berlin,
D-10587 Berlin, Germany
E-mail: christian.komusiewicz@tu-berlin.de, johannes.uhlmann@campus.tu-berlin.de

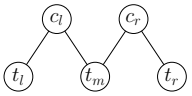


Fig. 1 An M-subgraph with $t_l, t_m, t_r \in V_t$ and $c_l, c_r \in V_c$.

1 Introduction

The Minimum-Flip Consensus Tree problem arises in computational phylogenetics in the context of supertree construction. Given a binary matrix, the task is to "flip" a minimum number of entries of the matrix in order to obtain a binary matrix that admits what is called a *perfect phylogeny*. These are matrices from which a rooted phylogenetic tree can be inferred [25, 33].

In this work, we employ a graph-theoretic formulation of the problem, which was introduced by Chen et al. [9]: the binary input matrix A is represented by a bipartite graph $G = (V_c, V_t, E)$ where an edge between two vertices $i \in V_c$ and $j \in V_t$ is drawn if and only if $A_{i,j} = 1$. The matrix then admits a perfect phylogeny if and only if the graph does not contain an *M-graph* as an induced subgraph. An M-graph is a path of five vertices with the first vertex belonging to V_t (see Figure 1 for an example). The flipping of a matrix entry $A_{i,j}$ from 0 to 1 corresponds to the insertion of the edge $\{i, j\}$, and the flipping of $A_{i,j}$ from 1 to 0 corresponds to the deletion of the edge $\{i, j\}$.

The Minimum-Flip Consensus Tree problem (MFCT) is then defined as follows.

Instance: A bipartite graph $G = (V_c, V_t, E)$ and an integer $k \geq 0$.
Question: Can G be changed by up to k edge modifications into an M-free graph, that is, a graph without an induced M-graph?

Chen et al. [9] showed that MFCT is NP-complete, which motivates the study of MFCT in the context of parameterized algorithmics [13, 17, 31]. We present new and improved results for MFCT parameterized by the number k of edge modifications. The main focus of this work is on polynomial-time data reduction with provable performance guarantee, that is, kernelization. In addition, we also present an improved search tree algorithm for MFCT. In the following, we first give a survey of previous work on MFCT and related problems and then describe our contributions.

Known results. The MFCT problem was introduced by Chen et al. [9] who also proved its NP-completeness and described a factor-$2d$ approximation algorithm for graphs with maximum degree d. Furthermore, they showed fixed-parameter tractability with respect to the number k of flips by describing a simple $O(6^k \cdot |V_t||V_c|)$-time search tree algorithm that is based on the forbidden induced subgraph characterization with M-graphs. Subsequently, Böcker et al. [5] improved the running time to $O(4.42^k(|V_c|+|V_t|)+|V_c|\cdot|V_t|)$ by employing a refined branching strategy. This theoretically proven running time acceleration was also confirmed by computational experiments [5].

MFCT is a special case of the Flip Supertree problem, where the input matrix is allowed to have "uncertain" entries [9]. Experiments have shown that Flip Supertree compares favorably with other supertree-construction methods [8]. However, Flip Supertree is W[2]-hard with respect to the parameter "number of

flips of certain entries" [4]. Chimani, Rahmann, and Böcker [11] proposed an ILP-formulation for Flip Supertree and showed that optimal solutions can be found for instances with up to 100 taxa. For a survey on fixed-parameter algorithms in phylogenetics we refer to [19]. An introduction into the topic of supertree construction methods is given by [3].

From a graph-theoretic point of view, MFCT is related to the class of so-called Π-Edge Modification problems: Given a graph G, a graph property Π, and an integer $k \geq 0$, the question is whether G can be transformed by at most k edge modifications into a graph with property Π. A lot of work has been put into classifying Π-Edge Modification problems with respect to their classical complexity [7, 30, 36]. Recently, parameterized algorithmics —in particular kernelizations—for Π-Edge Modification problems have attracted special attention. For instance, there is a series of papers studying the kernelizability of Cluster Editing and some of its variations [10, 15, 16, 18, 22, 23, 34]. Moreover, Kratsch and Wahlström [28] showed that there is a graph H on seven vertices such that H-free Editing does not admit a polynomial-size problem kernel, unless an unexpected complexity-theoretic collapse takes place. This contrasts the case of vertex deletion where for every fixed forbidden subgraph H the problem to destroy all occurrences of H by deleting at most k vertices admits a polynomial-size problem kernel [1]. Hence, for every forbidden subgraph it is a challenging task to prove the (non)existence of a polynomial-time problem kernel for the corresponding edge modification problem.

Damaschke [12] investigated kernelization in the context of enumerating all inclusion-minimal solutions of size at most k. In this scenario, when designing reduction rules one has to guarantee that all inclusion-minimal solutions of size at most k are preserved. Kernels that fulfill these additional constraints are called *full kernels*. In this setting, Damaschke [12] presents a full kernel consisting of $O(6^k)$ matrix entries for the following problem closely related to MFCT: Given a binary matrix and a nonnegative integer k, enumerate all inclusion-minimal sets of at most k flips that transform the matrix into a matrix that admits an unrooted perfect phylogeny.

Our contributions. In this work, we provide several polynomial-time data reduction rules for MFCT that lead to a problem kernel containing $O(k^3)$ vertices. This is the first nontrivial kernelization result for MFCT. Moreover, we present a search-tree algorithm with running time $O(3.68^k \cdot |V_c|^2|V_t|)$. Combining our kernelization algorithm with our search tree algorithm we achieve a running time of $O(3.68^k+|V_c|^2\cdot|V_t|\cdot|E|)$ instead of the previous $O(4.42^k\cdot(|V_c|+|V_t|)+|V_c|\cdot|V_t|)$ [5]. Furthermore, we describe one of the data reduction rules in a fairly abstract and general way, making it applicable to a wide range of Π-Edge Modification problems.

This work is organized as follows. In Section 2, we introduce basic notation and the most important definitions that we make use of. In Section 3, we present a data reduction rule that is correct for all graph properties that can be characterized by forbidden induced subgraphs that do not have two nonadjacent vertices with the same neighborhood. Note that these subgraphs include all paths of length at least 4, and cycles of length at least 5. In Section 4, we present a decomposition property which yields the basis for one of the data reduction rules and for the identification of a polynomial-time solvable special case needed for the improved

search tree strategy. In Section 5, we present one easy and two more intricate data reduction rules specific to MFCT. Based on these rules, in Section 6 we can show that MFCT admits a problem kernel with $O(k^3)$ vertices. Finally, in Section 7 we present an $O(3.68^k)$-size search tree for MFCT.

2 Preliminaries

Throughout this work, we use the following notation and definitions. For an undirected graph $G = (V, E)$, we use $V(G)$ and $E(G)$ to denote its vertex and edge set, respectively. The *open neighborhood* $N_G(v)$ of a vertex $v \in V$ is the set of vertices that are adjacent to v in $G = (V, E)$. Furthermore, let $N_G[v] := N_G(v) \cup \{v\}$ denote the *closed neighborhood* of v. For a set of vertices $V' \subseteq V$, we define $N_G(V') := \bigcup_{v \in V'} N_G(v) \setminus V'$ and $N_G[V'] := \bigcup_{v \in V'} N_G[v]$. The *degree* of a vertex v, denoted by $\deg_G(v)$, is the cardinality of $N_G(v)$. For a set of vertices $V' \subseteq V$, the *induced subgraph* $G[V']$ is the graph over the vertex set V' with edge set $\{\{v, w\} \in E \mid v, w \in V'\}$. For $V' \subseteq V$ we use $G - V'$ as abbreviation for $G[V \setminus V']$, and for a vertex $v \in V$ let $G - v$ denote $G - \{v\}$. For two sets X and Y with $X \cap Y = \emptyset$, let $E_{X,Y}$ denote the set $\{\{x, y\} \mid x \in X \wedge y \in Y\}$. As an abbreviation for $E_{\{x\},Y}$ we write $E_{x,Y}$.

For two sets E and F, define $E \Delta F := (E \setminus F) \cup (F \setminus E)$ (the *symmetric difference*). Further, for a bipartite graph $G = (V_c, V_t, E)$ and a set $F \subseteq E_{V_c,V_t}$ define $G \Delta F := (V_c, V_t, E \Delta F)$. With this notation, the definition of Minimum-Flip Consensus Tree (MFCT) reads as follows.

> Given a bipartite graph $G = (V_c, V_t, E)$ and an integer $k \geq 0$, does there exist $S \subseteq E_{V_c,V_t}$ with $|S| \leq k$ such that $G \Delta S$ is M-free?

Herein, S is called a solution. We call a solution *optimal* if it has minimum cardinality among all solutions. The elements of S are called *edge modifications*. We say that an edge modification e *involves* a vertex v if $v \in e$. Furthermore, a vertex v is called *affected* if S contains an edge modification involving v. Sometimes we refer to a vertex $v \in V_c$ as *c-vertex*, and to a vertex $v \in V_t$ as *t-vertex*.

A graph property Π is called *hereditary* if it is closed under vertex deletion. That is, if a graph G fulfills a hereditary graph property Π, then all induced subgraphs of G also fulfill Π. All graph properties that can be described by a (not necessarily finite) set of forbidden induced subgraphs (such as M-freeness for example) are hereditary.

For our data reduction we use a structure called *critical independent set*.

Definition 1 Given an undirected graph $G = (V, E)$, a set $I \subseteq V$ is called a *critical independent set* if $N_G(v) = N_G(w)$ for any two vertices $v, w \in I$ and I is maximal with respect to this property.

Note that $N_G(v) = N_G(w)$ directly implies that I is an independent set. All critical independent sets of a graph can be found in linear time [26]. Given a graph $G = (V, E)$ and the collection $\mathcal{I} = \{I_1, I_2, \ldots, I_q\}$, $q \leq n$, of its critical independent sets, the *critical independent set graph* of G is the undirected graph $(\mathcal{I}, \mathcal{E})$ with $\{I_i, I_j\} \in \mathcal{E}$ if and only if $\forall u \in I_i, v \in I_j : \{u, v\} \in E$. That is, the vertices of the critical independent set graph represent the critical independent sets and two vertices are adjacent if and only if the corresponding critical independent sets together form

a biclique. The critical independent set graph is defined in analogy to the *critical clique graph* which plays a decisive role in a kernelization of Cluster Editing [22].

A bipartite graph $G = (X, Y, E)$ is called a *chain graph* if the neighborhoods of the vertices in X form a chain [36]. That is, there is an ordering of the vertices in X, say $x_1, x_2, \ldots, x_{|X|}$, such that $N_G(x_1) \subseteq N_G(x_2) \subseteq \ldots \subseteq N_G(x_{|X|})$. It is easy to see that the neighborhoods of Y also form a chain if G is a chain graph. Moreover, a bipartite graph is a chain graph if and only if it is $2K_2$-free [36] (a $2K_2$ is the graph that consists of two independent edges). Since every M-graph contains an induced $2K_2$, the set of chain graphs is contained in the class of M-free graphs. One of our data reduction rules is based on identifying and reducing the size of subgraphs of the input graphs that are chain graphs and additionally have a special neighborhood structure.

We use the following notation concerning rooted trees. A vertex of a tree is called *node*. For a rooted tree T let $L(T)$ denote the *leaves* of T (that is, the nodes of degree one). The nodes in $V(T) \setminus L(T)$ are denoted as *inner nodes*. The root of T is denoted by $r(T)$. Moreover, for a node $v \in V(T)$, the maximal subtree rooted at v is denoted by T_v. We refer to a child of a node v as *leaf child* of v if it is a leaf; otherwise, it is called *non-leaf child* of v. We speak of the leaves (inner nodes) of a forest to refer to the union of the leaves (inner nodes) of the trees of the forest.

Induced M-graphs and M-freeness. Recall that a bipartite graph $G = (V_c, V_t, E)$ is called M-free if it does not contain an induced M-graph. An induced M-graph is denoted by a 5-tuple $(t_l, c_l, t_m, c_r, t_r)$, where $c_l, c_r \in V_c$ and $t_l, t_m, t_r \in V_t$. Two c-vertices c_l and c_r are *in conflict* if there exists an induced M-graph containing both c_l and c_r. Clearly, for an induced M-graph $(t_l, c_l, t_m, c_r, t_r)$ we have $t_l \in N_G(c_l) \setminus N_G(c_r)$, $t_m \in N_G(c_l) \cap N_G(c_r)$ and $t_r \in N_G(c_r) \setminus N_G(c_l)$. Thus, two vertices $c_l, c_r \in V_c$ are in conflict if and only if

$$(N_G(c_l) \setminus N_G(c_r) \neq \emptyset) \wedge (N_G(c_l) \cap N_G(c_r) \neq \emptyset) \wedge (N_G(c_r) \setminus N_G(c_l) \neq \emptyset).$$

If $c_l, c_r \in V_c$ are in conflict, we write $c_l \perp c_r$. In summary, for a bipartite graph $G = (V_c, V_t, E)$, the following statements are equivalent:

- G is M-free,
- no two c-vertices are in conflict, and
- for each pair of c-vertices c_l and c_r, it holds that $N(c_l) \cap N(c_r) = \emptyset$ or $N(c_l) \subseteq N(c_r)$ or $N(c_r) \subseteq N(c_l)$.

M-free graphs and rooted phylogenetic trees are closely related. Given a connected and M-free graph $G = (V_c, V_t, E)$, one can construct a rooted tree T with node set $V_t \cup V_c$ and with $L(T) = V_t$ such that $t_i \in V_t$ is a descendant of $c_j \in V_c$ if and only if $t_i \in N_G(c_j)$, see [9, 25, 33] for details. An example is given in Figure 2.

Parameterized algorithmics. Parameterized algorithmics [13, 17, 31] aims at a multivariate complexity analysis of problems. This is done by studying relevant problem parameters and their influence on the computational complexity. The decisive question is whether a given parameterized problem is *fixed-parameter tractable (FPT)* with respect to the parameter k. In other words, we ask for the existence of a solving algorithm with running time $f(k) \cdot \text{poly}(n)$ for some computable function f. A core tool in the development of parameterized algorithms that has been

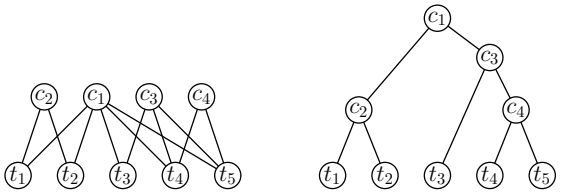


Fig. 2 An M-free graph G and the corresponding tree. In a connected M-free graph there exists a "universal" c-vertex, that is, a vertex adjacent to all t-vertices. In this example, c_1 is a universal vertex. This universal vertex is the root of the corresponding tree. Thus, one can build a tree by applying the above argument recursively for the connected components of $G - c_1$.

recognized as one of the most important contribution of parameterized algorithmics to practical computing [6, 14, 24, 27, 31] is polynomial-time preprocessing by *data reduction rules*, often yielding a *problem kernel*. Herein, the goal is, given any problem instance G with parameter k, to transform it in polynomial time into a new instance G' with parameter k' such that the size of G' is bounded from above by some function only depending on k, $k' \leq k$, and (G, k) is a yes-instance if and only if (G', k') is a yes-instance. We call a data reduction rule *correct* if the new instance after an application of this rule is a yes-instance if and only if the original instance is a yes-instance. An instance is called *reduced* with respect to a set of data reduction rules if each of the data reduction rules has been exhaustively applied.

3 A Universal Rule for Critical Independent Sets

In this section, we describe a polynomial-time data reduction rule that applies to parameterized graph modification problems for a family of hereditary graph properties. It is based on the modular decomposition of a graph and has been applied to Bicluster Editing [34]. We use this reduction rule for obtaining a kernel for MFCT, but we believe that it can be useful for other graph modification problems as well. To this end, we show that this reduction rule can be applied to a wide range of Π-Edge Modification problems, including MFCT.

The basic idea of the data reduction is to show that, for some graph properties, vertices that belong to the same critical independent set are subject to the "same" edge modifications. Therefore, large critical independent sets can be reduced. First, we give a description of these graph properties.

Let Π be a hereditary graph property. We call Π *critical independent set preserving* (*cisp*) whenever for all forbidden induced subgraphs F of Π, there are no two vertices $u, v \in V(F)$ that belong to the same critical independent set in F (that is, all critical independent sets of F have size one). These forbidden subgraphs include for example all paths of length at least four and cycles of length $\neq 4$. Note that formally M-freeness is not a graph property because it is not invariant under graph isomorphism: An M-graph has its first vertex in V_t. Therefore, exchanging V_t and V_c may transform an M-free graph into a graph that contains an M-graph as induced subgraph and vice versa. Nevertheless, all vertices in an induced M-graph have different neighborhoods. Therefore, the following lemmas and reduction rule also apply directly to MFCT.

First, we show that cisp graph properties are closed under a certain vertex-addition operation.

Lemma 1 *Let $G = (V, E)$ be a graph fulfilling a cisp graph property Π. Let G' be the graph that results by adding to G a new vertex $x \notin V$ and making it adjacent to $N_G(v)$ for an arbitrary vertex $v \in V$. Then, G' also fulfills Π.*

Proof Let G' be the graph that is obtained by adding x to G and inserting all edges between x and $N_G(v)$ for some $v \in V$. We prove the lemma by showing that G' does not contain a forbidden induced subgraph.

First, since G has the graph property Π, there is no forbidden induced subgraph $G'[S]$ such that $x \in S$ and $v \notin S$. Otherwise, the vertex set $\{v\} \cup (S \setminus \{x\})$ would induce a forbidden subgraph in G, contradicting the assumption that G has property Π. Second, since v and x are part of a critical independent set and Π is cisp, there is no forbidden induced subgraph that contains both v and x. Therefore, G' has property Π. □

Using Lemma 1, we can show that for graph modification problems for cisp graph properties there is an optimal solution that "treats" the vertices of any critical independent set equally. This type of solution is formally defined as follows.

Definition 2 A solution S for an instance (G, k) of Π-Edge Modification is called *regular* if every critical independent set of G is contained in a critical independent set of $G \Delta S$.

In other words, if S is regular then two vertices that have an identical open neighborhood in the input graph G also have an identical open neighborhood in $G \Delta S$.

Lemma 2 *Let Π be a cisp graph property and let $G = (V, E)$ denote an undirected graph. Moreover, let S denote a solution for Π-Edge Modification on G and let X denote the set of vertices affected by S. Then, there exists a solution S^* such that*

- $|S^*| \leq |S|$,
- *S^* is regular,*
- *$X^* \subseteq X$, where X^* denotes the set of vertices affected by S^*.*

Proof Assume that there exists a critical independent set I of G that is not contained in a critical independent set in $G \Delta S$. We show that, by a local modification, one can find a graph $G' = (V, E')$ fulfilling Π such that

1. the number of edge modifications to transform G into G' is at most the number of edge modifications to transform G into $G \Delta S$, that is, $|E \Delta E'| \leq |S|$,
2. I is contained in a critical independent set in G',
3. for each critical independent set I^* of G the number of critical independent sets of G' intersecting with I^* is at most the number of critical independent sets of $G \Delta S$ intersecting with I^*, and
4. the set of affected vertices in G' is a subset of X.

By Condition 3, this local modification step can be applied iteratively until the solution is regular. Moreover, by Condition 4 the set of affected vertices in the resulting graph is a subset of X.

First, we formally specify the modification. Then, we show that the above conditions are fulfilled. Let $I_1, I_2, \ldots, I_\ell$ denote the critical independent sets in $G \Delta S$ with $I_i \cap I \neq \emptyset$, $1 \leq i \leq \ell$. Observe that $\ell > 1$. For a vertex $w \in I$ let S_w denote the set of edge modifications from S involving w and vertices in $V \setminus I$, that is, $S_w := \{\{w, x\} \in S \mid x \in V \setminus I\}$. Let $v \in I$ such that $|S_v|$ is minimal and assume without loss of generality that $v \in I_1$. Build a graph G' as follows from $G \Delta S$. First, remove all vertices in $I \setminus \{v\}$ from $G \Delta S$. Then, add the vertices in $I \setminus \{v\}$ step-by-step making each vertex adjacent to the vertices in the current open neighborhood of v. Let $S' := E \Delta E'$ and note that $G' = G \Delta S'$.

By Lemma 1, G' fulfills Π. To prove Condition 1, we show that $|S'| \leq |S|$. Note that in the construction above we "change" only edges incident with the vertices in I and, hence, $G'[V \setminus I]$ is identical to $G \Delta S[V \setminus I]$. Moreover, since each vertex $w \in I \setminus I_1$ gets the same closed neighborhood as v (more specifically, the vertices in $N_{G \Delta S}(v) \setminus I$), we have to spend at most $|S_v|$ edge modifications for every $w \in I \setminus \{v\}$ (instead of at least $|S_w|$). Since $|S_v| \leq |S_w|$ by the choice of v, it follows that $|S'| \leq |S|$.

Condition 2 follows directly from the fact that by construction all vertices in I have the same open neighborhood in G' (namely, $N_{G \Delta S}(v) \setminus I$).

Next, we show that Condition 3 is fulfilled. To this end, note that deleting a vertex does not increase the number of critical independent sets of a graph. Moreover, observe that two nonadjacent vertices with an identical neighborhood have an identical neighborhood after adding a vertex and making it adjacent to the neighbors of an existing vertex. Hence, for each critical independent set I^* of G the number of critical independent sets of G intersecting with I^* is at most the number of critical independent sets of G' intersecting with I^*.

Clearly, by construction a vertex w not affected by S is not affected by S' implying Condition 4. □

Note that since the modification operations in the proof of Lemma 2 can be performed in polynomial time, we can compute a regular solution from a given (arbitrary) solution in polynomial time.

Protti et al. [34] devised the following data reduction rule for Bicluster Editing. With Lemma 2 at hand, we can show that this rule is correct for all cisp graph properties.

Rule 1 *Let $I \subseteq V$ be a critical independent set. If $|I| > k+1$, then delete $|I| - (k+1)$ arbitrary vertices from I.*

Lemma 3 *Rule 1 is correct and can be exhaustively applied in $O(|V| + |E|)$ time.*

Proof By Lemma 2, we know that if we modify an edge incident with a vertex in a critical independent set I, then we must perform the same edge modification to any other vertex in I. That is, as long as $|I| > k$, a solution of size at most k does not modify any edge incident with a vertex in I. Protti et al. [34] have shown that Rule 1 can be exhaustively applied in $O(|V| + |E|)$ time. □

We can apply Rule 1 for MFCT since the graph property of being M-free is critical independent set preserving: all vertices in an induced M-graph have different neighborhoods. This general data reduction rule also applies to the Completion and Deletion version of a Π-Edge Modification problem for a cisp

graph property Π. Examples for graph modification problems to which this rule can be applied are Chain Deletion and Co-trivially Perfect Deletion.[1]

Bessy et al. [2] independently obtained a similar result. They considered graph properties that are closed under "true twin addition". More specifically, a graph property Π is closed under true twin addition if for any graph G fulfilling Π adding a vertex and making it adjacent to each vertex in $N[v]$ for some vertex $v \in V(G)$, yields a graph that also fulfills Π. This true twin addition operation is similar to the vertex addition operation applied in Lemma 1. The difference is that in the case of "true twin addition" the new vertex is made adjacent to all vertices in the *closed* neighborhood of an existing vertex, whereas in Lemma 1 the new vertex is made adjacent to all vertices in the *open* neighborhood of an existing vertex. Bessy et al. [2] showed that for graph properties closed under true twin addition there is a solution for the corresponding edge modification problem such that two adjacent vertices with an identical closed neighborhood in the input graph also have an identical closed neighborhood in the modified graph[2]. Hence, a reduction rule similar to Rule 1 applies for graph properties closed under true twin addition. For a discussion on graph properties generalizing cisp graph properties and graph properties closed under true twin addition and further corresponding generalized data reductions rules we refer to [35, Chapter 3.4].

Note that for Bicluster Editing this data reduction rule (together with the trivial rule to remove all connected components that are already biclusters) yields a problem kernel with $O(k^2)$ vertices [34]. For most cisp graph properties, however, these two rules are not sufficient to obtain a kernel. This is also the case for MFCT. Hence, in Section 5, we present further more intricate rules specific to MFCT.

4 A Decomposition Property

In this and the following sections, we focus on the MFCT problem. Here, we show that, given a set C of c-vertices such that no vertex in C is in conflict with any c-vertex outside of C, the conflicts involving the vertices in C can be resolved independently from the conflicts not involving vertices in C. This fact is used to prove the correctness of one of our data reduction rules in Section 5 and to identify a polynomial-time solvable special case of MFCT in Section 7 that is the basis for an improved search tree algorithm.

For an input graph $G = (V_c, V_t, E)$, we consider the *conflict graph* $G_\perp = (V_c, F)$ with vertex set V_c and edge set $F := \{\{c', c''\} \mid c', c'' \in V_c \wedge c' \perp c''\}$. Roughly speaking, we show that for each connected component of the conflict graph, the conflicts can be resolved independently. To show this property, we need the following lemma. Throughout this section, we identify a connected component by its vertex set.

Lemma 4 *Let C_1 and C_2 be two connected components of $G_\perp$ with $N_G(C_1) \cap N_G(C_2) \neq \emptyset$. Then, either*

[1] Kernelization results for these problems have been obtained by Guo [21].

[2] Note that two vertices with the same open neighborhood have the same closed neighborhood in the complement graph and vice versa. In this sense, our results can be seen as equivalent to the result of Bessy et al. [2] when considering the graph property defined by the complements of the forbidden induced subgraphs in the complement graph.

– *for each* $a \in C_1$ *it holds that* $N_G(C_2) \subseteq N_G(a)$ *or* $N_G(a) \cap N_G(C_2) = \emptyset$*, or*
– *for each* $b \in C_2$ *it holds that* $N_G(C_1) \subseteq N_G(b)$ *or* $N_G(b) \cap N_G(C_1) = \emptyset$*.*

Proof Since $N_G(C_1) \cap N_G(C_2) \neq \emptyset$, there is a vertex $x \in C_1$ and a vertex $y \in C_2$ with $N_G(x) \cap N_G(y) \neq \emptyset$. Furthermore, because x and y are in two different connected components of $G_\perp$, they are not in conflict, and, thus $N_G(x) \subseteq N_G(y)$ or $N_G(y) \subseteq N_G(x)$. Assume without loss of generality that $N_G(y) \subseteq N_G(x)$. To simplify the notation in the remainder of the proof, we use $N(v) := N_G(v)$ to denote the open neighborhood of a vertex v in G.

First, we show $N(C_2) \subseteq N(x)$. Assume towards a contradiction that $N(C_2) \setminus N(x) \neq \emptyset$. Let $A := \{y' \in C_2 \mid N(y') \subseteq N(x)\}$. Note that $y \in A$ and, hence, $A \neq \emptyset$. Furthermore, let $B := C_2 \setminus A$ and observe that $B \neq \emptyset$ by the assumption that $N(C_2) \setminus N(x) \neq \emptyset$. Since C_2 is a connected component of $G_\perp$ there is a vertex $y' \in A$ and a vertex $y'' \in B$ being in conflict with each other. That is,

$$N(y') \setminus N(y'') \neq \emptyset \text{ and } N(y') \cap N(y'') \neq \emptyset \text{ and } N(y'') \setminus N(y') \neq \emptyset.$$

Since $N(y') \subseteq N(x)$ the above directly implies $N(x) \cap N(y'') \neq \emptyset$ and $N(x) \setminus N(y'') \neq \emptyset$. Furthermore, $y'' \in B$ implies $N(y'') \setminus N(x) \neq \emptyset$. As a consequence, x is in conflict with y'', contradicting the fact that x and y'' are contained in distinct connected components of $G_\perp$.

Next, we prove that for each $a \in C_1$ it holds that $N(C_2) \subseteq N(a)$ or $N(a) \cap N(C_2) = \emptyset$. To this end, consider the following partition of C_1:

$$C_1^{\supseteq,\emptyset} := \{a \in C_1 \mid N(a) \supseteq N(C_2) \text{ or } N(a) \cap N(C_2) = \emptyset\},$$

$$C_1^{\subset} := \{a \in C_1 \mid N(a) \subset N(C_2)\},$$

$$C_1^{r} := C_1 \setminus (C_1^{\supseteq,\emptyset} \cup C_1^{\subset}).$$

Note that $x \in C_1^{\supseteq,\emptyset}$, and, thus, $C_1^{\supseteq,\emptyset} \neq \emptyset$. To prove the above claim, we show that $C_1^{\subset} \cup C_1^{r} = \emptyset$. Assume towards a contradiction that $C_1^{\subset} \cup C_1^{r} \neq \emptyset$. We distinguish two cases based on whether $C_1^{r} = \emptyset$ or not.

Case 1: $C_1^{r} = \emptyset$**.** From assumption $C_1^{\subset} \cup C_1^{r} \neq \emptyset$ it follows that $C_1^{\subset} \neq \emptyset$. Furthermore, note that no vertex in $C_1^{\subset}$ is in conflict with a vertex in $C_1^{\supseteq,\emptyset}$. Since $C_1^{\supseteq,\emptyset} \neq \emptyset$ and $C_1^{\subset} \neq \emptyset$, this implies that $G_\perp[C_1]$ consists of at least two connected components; a contradiction.

Case 2: $C_1^{r} \neq \emptyset$**.** Let $q \in C_1^{r}$ be arbitrarily chosen. Since $N(q) \setminus N(C_2) \neq \emptyset$ by the definition of C_1^{r} and q is not in conflict with any vertex in C_2, for every vertex $y' \in C_2$ we have either $N(y') \subseteq N(q)$ or $N(y') \cap N(q) = \emptyset$. Let $Y := \{y' \in C_2 \mid N(y') \subseteq N(q)\}$ and $Z := \{y' \in C_2 \mid N(y') \cap N(q) = \emptyset\}$. Note that $C_2 = Y \cup Z$, $Y \neq \emptyset$ (since $N(q) \cap N(C_2) \neq \emptyset$), and $Z \neq \emptyset$ (since $N(C_2) \setminus N(q) \neq \emptyset$). Moreover, $N(Y) \cap N(Z) = \emptyset$ and, hence, no vertex in Y is in conflict with a vertex in Z. Thus, graph $G_\perp[C_2]$ consists of at least two connected components; a contradiction. □

The following proposition now states the decomposition property. Recall that a solution S for an instance (G, k) is *regular* if every critical independent set of G is contained in a critical independent set of $G \Delta S$.

Proposition 1 *Let $C_1, C_2, \ldots, C_\ell$ denote the connected components of $G_\perp$ and define $G_i := G[C_i \cup N_G(C_i)]$. Let S_i denote a regular optimal solution for G_i, $1 \leq i \leq \ell$. Then, $S := \bigcup_{i=1}^{\ell} S_i$ is an optimal solution for G.*

Proof The proof is organized as follows. First, we show that $|S|$ is a lower bound for an optimal solution, then we show that S is a solution, that is, $G \Delta S$ is M-free.

First, we show that $\sum_{i=1}^{\ell} |S_i|$ is a lower bound for the size of an optimal solution for G (note that $|S| = \sum_{i=1}^{\ell} |S_i|$ since all S_i's are pairwise disjoint). To this end, observe that for every solution S' clearly $S'_i := S' \cap \{\{v_c, v_t\} \mid v_c \in C_i, v_t \in N_G(C_i)\}$ is a solution for G_i. Moreover, $|S_i| \leq |S'_i|$ since S_i is an optimal solution for G_i. Since all S'_i's are pairwise disjoint, we thus have $|S| \leq |S'|$.

Second, we show that $G \Delta S$ is M-free. Assume towards a contradiction that there are two c-vertices c_1 and c_2 that are in conflict in $G \Delta S$. Without loss of generality, let C_1 be the connected component of $G_\perp$ containing c_1 and C_2 be the connected component of $G_\perp$ containing c_2. Since $N_{G\Delta S}(c_1) \cap N_{G\Delta S}(c_2) \neq \emptyset$ and all edge insertions of S incident with c_1 and c_2 are between c_1 and $N_G(C_1)$ and c_2 and $N_G(C_2)$, respectively, it follows that $N_G(C_1) \cap N_G(C_2) \neq \emptyset$. Thus, according to Lemma 4, we can assume without loss of generality that for every $x \in C_2$ it holds that either $N_G(C_1) \subseteq N_G(x)$ or $N_G(x) \cap N_G(C_1) = \emptyset$. As a consequence, $N_G(C_1) \subseteq N_G(C_2)$. Let $C_{2,a} := \{x \in C_2 \mid N_G(C_1) \subseteq N_G(x)\}$ and $C_{2,b} := \{x \in C_2 \mid N_G(C_1) \cap N_G(x) = \emptyset\}$. By Lemma 4, $C_2 = C_{2,a} \cup C_{2,b}$. Hence, $N_G(C_1)$ belongs to a critical independent set in G_2: for every vertex $t \in N_G(C_1)$ it holds that $N_{G_2}(t) = C_{2,a}$. Let $t_1, t_{1,2}, t_2$ denote three t-vertices that, together with c_1 and c_2, induce an M-graph in $G \Delta S$. Without loss of generality, assume that $t_1 \in N_{G\Delta S}(c_1) \setminus N_{G\Delta S}(c_2)$, $t_{1,2} \in N_{G\Delta S}(c_1) \cap N_{G\Delta S}(c_2)$, and $t_2 \in N_{G\Delta S}(c_2) \setminus N_{G\Delta S}(c_1)$. Since both t_1 and $t_{1,2}$ are neighbors of c_1 in $G \Delta S$ and all edge modifications involving vertices in C_1 are between C_1 and $N_G(C_1)$ we have $\{t_1, t_{1,2}\} \subseteq N_G(C_1)$. Finally, observe that $N_G(C_1)$ is not a critical independent set in $G_2 \Delta S_2$ since $t_1 \notin N_{G_2 \Delta S_2}(c_2)$ but $t_{1,2} \in N_{G_2 \Delta S_2}(c_2)$: this is a contradiction to the assumption that S_2 is regular since $N_G(C_1)$ is a critical independent set in G_2. □

5 Specific Data Reduction Rules for Minimum-Flip Consensus Tree

In this section, we present three further polynomial-time data reduction rules that together with Rule 1 produce an $O(k^3)$-vertex kernel.

The first data reduction rule is obvious.

Rule 2 *Remove M-free connected components from the input graph.*

Applying Rule 1 and a rule that removes all isolated bicliques already yields an $O(k^2)$-vertex kernel for Bicluster Editing [34]. However, for Minimum-Flip Consensus Tree we need two further, more involved data reduction rules. The main difference here is that for M-free graphs, we have a much more complicated neighborhood structure than for P_4-free bipartite graphs (so-called bicluster graphs), where each connected component is a complete bipartite graph. That is, in contrast to bicluster graphs, where each connected component contains at most two critical independent sets, in case of M-free-graphs each connected component might contain an unbounded number of critical independent sets.

The next data reduction rule removes c-vertices from G that do not appear in any M-graph. The correctness of this rule follows from the decomposition property introduced in Section 4: a c-vertex c' that is not in any conflict forms a connected component $\{c'\}$ in the conflict graph $G_\perp$ whose associated MFCT–subinstance $G[\{c'\} \cup N_G(c')]$ is M-free. Thus, according to Proposition 1, the sizes of optimal solutions for G and $G - c'$ are equal.

Rule 3 *Let $G = (V_c, V_t, E)$ be a bipartite graph. If there exists a vertex $c \in V_c$ that is not in conflict with any other vertex in V_c, then remove c.*

Lemma 5 *Rule 3 is correct and can be exhaustively applied in $O(|V_c|^2 \cdot |V_t|)$ time.*

Proof As discussed above, the correctness of Rule 3 follows directly from Proposition 1.

For the running time consider the following. To test the adjacency of two vertices in constant time, an adjacency matrix is built prior to the application of the rule. Then, for each pair of vertices $c_1, c_2 \in V_c$, we can determine in $O(|V_t|)$ time whether they are in conflict by checking for each vertex $t \in V_t$, whether it is adjacent to c_1, c_2, or both. Each c-vertex that is in conflict with some other vertex is marked. Finally, unmarked vertices are removed from the graph. This can be done in $O(|E|)$ time. The overall running time is thus $O(|V_c|^2 \cdot |V_t|)$. Note that every vertex that is in conflict prior to the application of the rule is also in conflict after the application of the rule. Thus, above procedure results in an instance that is reduced with respect to Rule 3. □

The structurally "deepest" reduction rule shrinks induced subgraphs of the input graph that resemble "local" chain graphs. We call such a subgraph *P-structure*:

Definition 3 Let $G = (V_c, V_t, E)$ be a bipartite graph. A pair (P_c, P_t) of vertex sets $P_c \subseteq V_c$ and $P_t \subseteq V_t$ forms a *P-structure* if the following three properties are fulfilled:

1. $G[P_c \cup P_t]$ is a chain graph,
2. for all $c', c'' \in P_c$ it holds that $N_G(c') \setminus P_t = N_G(c'') \setminus P_t$, and
3. for all $t', t'' \in P_t$ it holds that $N_G(t') \setminus P_c = N_G(t'') \setminus P_c$.

For a P-structure (P_c, P_t) of a bipartite graph G the neighborhoods in G of the vertices in P_c (and P_t) also form a chain (since "outside" of the P-structure they have the same neighbors). Moreover, note that $G[P_c \cup P_t]$ is M-free.

Our last reduction rule shrinks large P-structures. See Figure 3 for an example.

Rule 4 *Let $(G = (V_c, V_t, E), k)$ denote an MFCT-instance and let (P_c, P_t) be a P-structure in G, where $P_t = \{t_1, t_2, \ldots, t_\ell\}$ with $N_G(t_1) \subseteq N_G(t_2) \subseteq \ldots \subseteq N_G(t_\ell)$. If $\ell > 2(k+1)$, then remove $R_t := \{t_{k+2}, t_{k+3}, \ldots, t_{\ell-(k+1)}\}$ from G.*

Next, we prove the correctness of Rule 4. Subsequently, we show that it can be applied in polynomial time.

Lemma 6 *Rule 4 is correct.*

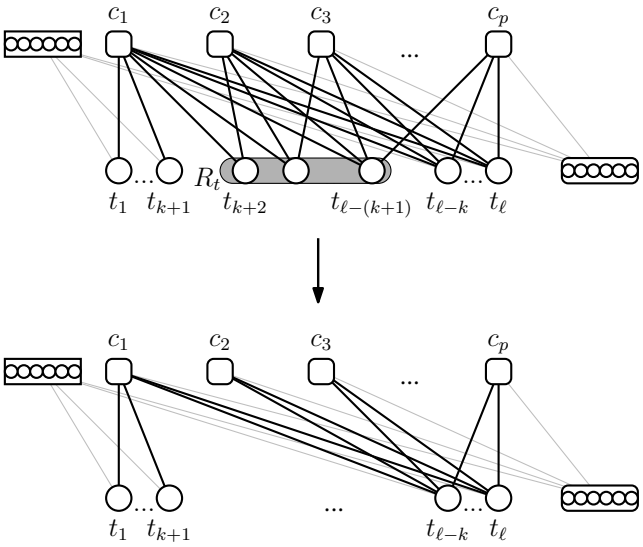


Fig. 3 Illustration of the application of Rule 4. The edges within the P-structure are colored black and the edges between a vertex of the P-structure and a vertex outside the P-structure are colored grey. Rule 4 removes all but the first and last $k+1$ t-vertices of the P-structure.

Proof Let $G = (V_c, V_t, E)$, P_c, P_t, and R_t be specified as in Rule 4. Let $P_c = \{c_1, c_2, \ldots, c_p\}$. Since (P_c, P_t) forms a P-structure, we can assume that $N_G(t_1) \subseteq N_G(t_2) \subseteq \ldots \subseteq N_G(t_\ell)$ and $N_G(c_1) \supseteq N_G(c_2) \supseteq \ldots \supseteq N_G(c_p)$. Furthermore, let $G' := G - R_t$ denote the reduced graph.

For the correctness of Rule 4, we prove the following.

Claim: (G, k) is a yes-instance $\iff$ (G', k) is a yes-instance.

"$\Rightarrow$": This direction follows directly from the fact that M-freeness is a hereditary graph property.

"$\Leftarrow$": Given a solution S' of size at most k for G', we show that one can construct a solution of size at most k for G. First, note that since $|S'| \leq k$ there exists an i with $1 \leq i \leq k+1$ such that t_i is not involved in any edge modification in S', that is, $N_{G'}(t_i) = N_{G' \Delta S'}(t_i)$. Analogously, there is a j with $\ell - k \leq j \leq \ell$ such that t_j is not involved in any edge modification in S', that is, $N_{G'}(t_j) = N_{G' \Delta S'}(t_j)$. Note that $N_{G'}(t_i) \subseteq N_{G'}(t_j)$ and, hence, $N_{G' \Delta S'}(t_i) \subseteq N_{G' \Delta S'}(t_j)$. Furthermore, let $T := \{t_{i+1}, t_{i+2}, \ldots, t_{j-1}\}$ and $T' := T \setminus R_t$. Consider the edge modification set $S'' := S' \setminus E_{T', V_c}$. In other words, S'' contains all edge modifications from S' except those involving a vertex from T'. Clearly, S'' is a solution of size at most k for $G - T = G' - T'$. We show that from S'' one can build a solution of size at most k for G. To this end, we distinguish the cases that $N_G(t_i) = N_G(t_j)$ and $N_G(t_i) \subset N_G(t_j)$.

Case 1: $N_G(t_i) = N_G(t_j)$**.** This condition implies that $N_G(t') = N_G(t_i) = N_G(t_j)$ for every $t' \in T' \cup R_t$ since $N_G(t_i) \subseteq N_G(t') \subseteq N_G(t_j)$ and $N_G(t_i) = N_G(t_j)$. That is, in $G \Delta S''$, every vertex $t' \in T' \cup R_t$ has the same neighborhood as t_i and t_j (since t_i, t_j, and t' are not involved in any edge modification in S'') and, as a consequence of Lemma 1, $G \Delta S''$ is M-free (observe that $G \Delta S''$ results

from $(G' - T')\Delta S''$ by adding the vertices in T step-by-step, making each vertex adjacent to every vertex in $N_G(t_i)$). Hence, S'' is a solution of size at most k for G.

Case 2: $N_G(t_i) \subset N_G(t_j)$**.** We need some observations on the structure of $G' - T'$ and $(G' - T')\Delta S''$.

First, we show that $N_G(t_j) \setminus N_G(t_i)$ is part of a critical independent set in $G' - T'$. By $N_G(t_i) \setminus P_c = N_G(t_j) \setminus P_c$ (Condition 3 of Definition 3), it follows that $N_G(t_j) \setminus N_G(t_i) \subseteq P_c$. Hence, all vertices from $N_G(t_j) \setminus N_G(t_i)$ have the same neighbors in $V_t \setminus P_t$ in $G' - T'$ according to Condition 2 of Definition 3. Moreover, every vertex in $N_G(t_j) \setminus N_G(t_i)$ is nonadjacent to all vertices in $\{t_1, \ldots, t_i\}$ since $N_G(t_1) \subseteq \ldots \subseteq N_G(t_i) \subset N_G(t_j)$ and is adjacent to all vertices in $\{t_j, \ldots, t_\ell\}$ since $N_G(t_i) \subset N_G(t_j) \subseteq \ldots \subseteq N_G(t_\ell)$. Thus, all vertices in $N_G(t_j) \setminus N_G(t_i)$ have an identical neighborhood in $G' - T'$.

Thus, by Lemma 2, there is a solution S of size at most k for $G' - T' = G - T$ such that $N_G(t_j) \setminus N_G(t_i)$ is contained in a critical independent set in $(G - T)\Delta S$ and neither t_i nor t_j are involved in any edge modification from S.

Next, we argue that $G_S := G\Delta S$ is M-free. Assume towards a contradiction that G_S contains an M-graph. We show that then G_S contains an M-graph $(t_l, c_l, t_m, c_r, t_r)$ such that

$$\text{exactly one of } t_l \text{ and } t_r \text{ is contained in } T \text{ and } t_m \notin T.$$

From the existence of such an M-graph we then derive a contradiction to the fact that $(G - T)\Delta S$ is M-free. To prove the existence of an M-graph as described above, consider an arbitrary M-graph $(t_l^*, c_l^*, t_m^*, c_r^*, t_r^*)$ in G_S. First, we show that not both of t_l^* and t_r^* are contained in T. Observe that t_l^* has a neighbor not contained in $N_{G_S}(t_r^*)$ (namely c_l^*), and, vice versa, t_r^* has a neighbor not contained in $N_{G_S}(t_l^*)$ (namely c_r^*). Since, however, the neighborhoods of the vertices in T form a chain in G_S for any two vertices $t_x, t_y \in T$ either $N_{G_S}(t_x) \subseteq N_{G_S}(t_y)$ or $N_{G_S}(t_y) \subseteq N_{G_S}(t_x)$. Thus not both of t_l^* and t_r^* can be contained in T.

Next, assume that $t_m^* \in T$. We show that then $c_l^*, c_r^*, t_l^*, t_r^*$ and t_j (instead of t_m^*) induce an M-graph. Since $t_m^* \in T$ it holds that $N_{G_S}(t_m^*) \subseteq N_{G_S}(t_j)$, implying $t_r^* \neq t_j$ and $t_l^* \neq t_j$. Thus, since t_j is adjacent to c_l^* and c_r^* in G_S (this follows from $N_{G_S}(t_m^*) \subseteq N_{G_S}(t_j)$) the vertices c_l^*, c_r^*, t_l^*, t_r^*, and t_j (instead of t_m^*) induce an M-graph in G_S.

In summary, there is an M-graph $(t_l, c_l, t_m, c_r, t_r)$ in G_S such that not both t_l and t_r are contained in T and $t_m \notin T$. Since $(G - T)\Delta S$ is M-free, every M-graph in G_S contains at least one vertex from T. Thus, either $t_l \in T$ or $t_r \in T$. Without loss of generality assume that $t_l \in T$.

Finally, note that $c_r \notin N_{G_S}(t_i)$ because $c_r \notin N_{G_S}(t_l) \supseteq N_{G_S}(t_i)$. Moreover, note that $c_l \in N_{G_S}(t_j)$, since $N_{G_S}(t_l) \subseteq N_{G_S}(t_j)$. We distinguish the two cases $c_l \in N_G(t_i)$ and $c_l \in N_G(t_j) \setminus N_G(t_i)$ and in each case derive a contradiction.

Case 2.1: $c_l \in N_G(t_i)$**.** Then, $\{t_i, c_l, t_m, c_r, t_r\}$ induces an M-graph not containing a vertex from T because $c_l \in N_{G_S}(t_i)$ (since $c_l \in N_G(t_i)$ by the case condition) but $\{t_i, c_r\} \notin E(G_S)$ (since $c_r \notin N_{G_S}(t_i)$). This contradicts the assumption that $(G - T)\Delta S$ is M-free.

Case 2.2: $c_l \in N_G(t_j) \setminus N_G(t_i)$**.** Recall that by the discussion above $t_r, t_m \notin T$ and $c_r \notin N_G(t_i)$. Further, we can assume that c_r is not contained in $N_G(t_j) \setminus N_G(t_i)$; otherwise, since $N_G(t_j) \setminus N_G(t_i)$ forms a critical independent set in $(G - T)\Delta S$ and $t_r, t_m \notin T$, it would follow that c_l and t_r are adjacent (however, $t_r \in N_{G_S}(c_r) \setminus$

$N_{G_S}(c_l)$). Hence, $c_r \notin N_G(t_j)$ and, since c_r is adjacent to both t_m and t_r, it follows that $t_m \neq t_j$ and $t_r \neq t_j$. Thus, since $c_l \in N(t_l) \subseteq N(t_j)$ (by the case condition) but $c_r \notin N_G(t_j)$ the vertex set $\{t_j, c_l, t_m, c_r, t_r\}$ induces an M-graph in G_S not containing any vertex from T. This contradicts the M-freeness of $(G-T)\Delta S$. □

Finally, we show that Rule 4 can be applied in polynomial time.

Lemma 7 *A P-structure (P_c, P_t) such that $|P_t|$ is maximal can be found in $O(|V_c|^2 \cdot |E|)$ time.*

Proof Assume that G contains a P-structure (P_c, P_t) where $P_c = \{c_1, c_2, \ldots, c_p\}$ and $N(c_1) \supseteq N(c_2) \supseteq \ldots \supseteq N(c_p)$. We show that given the two vertices $c_1, c_p \in P_c$ we can find a P-structure (C, T) with $C = P_c$ such that $|T|$ is maximal in $O(|E|)$ time. Hence, by trying all pairs $c', c'' \in V_c$ we find a P-structure (P_c, P_t) for which $|P_t|$ is maximal in $O(|V_c|^2 \cdot |E|)$ time. We distinguish the cases $N(c_1) = N(c_p)$ and $N(c_1) \supset N(c_p)$.

Case 1: $N(c_1) = N(c_p)$**.** Any two vertices in P_c have an identical neighborhood. Hence, we can assume that $P_c = I$, where I denotes the critical independent set containing c_1 and c_p. Furthermore, since P_c is a critical independent set, Definition 3 implies that P_t is a critical independent set, too. Moreover, it is not hard to verify that I together with any critical independent set in $N(c_1)$ forms a P-structure. Note that the set of all critical independent sets of a graph can be computed in $O(|E|)$ time [29]. Hence, one can find a P-structure (I, P_t) such that $|P_t|$ is maximal in $O(|E|)$ time.

Case 2: $N(c_1) \supset N(c_p)$**.** We use the following notation. For two c-vertices c', c'' with $N(c'') \subseteq N(c')$ let

$$\begin{aligned} S(c', c'') &:= \{x \in V_c \mid N(c'') \subseteq N(x) \subseteq N(c')\}, \\ T'(c', c'') &:= N(c') \setminus N(c''), \\ C_{out}(c', c'') &:= N(T'(c', c'')) \setminus S(c', c''), \\ T''(c', c'') &:= \{t \in N(c'') \mid N(t) = (S(c', c'') \cup C_{out}(c', c''))\}. \end{aligned}$$

Let (P_c, P_t) denote a P-structure of G where $P_c = \{c_1, c_2, \ldots, c_p\}$ with $N(c_1) \supseteq N(c_2) \supseteq \ldots \supseteq N(c_p)$ and $N(c_1) \supset N(c_p)$. We show that in this case, (P_c, P_t) is uniquely determined by c_1 and c_p. More precisely, we show the following.

Claim: If (P_c, P_t) is a P-structure where $|P_t|$ is maximal, then $P_c = S(c_1, c_p)$ and $P_t = T'(c_1, c_p) \cup T''(c_1, c_p)$.

First, we show that $P_c = S(c_1, c_p)$. From the definition of a P-structure, it follows directly that $P_c \subseteq S(c_1, c_p)$. Hence, it remains to show that $S(c_1, c_p) \subseteq P_c$. Assume towards a contradiction that there exists a vertex $x \in S(c_1, c_p) \setminus P_c$. Note that since $N(c_1) \setminus P_t = N(c_p) \setminus P_t$ (Condition 2 of Definition 3) it holds that $T'(c_1, c_p) \subseteq P_t$. Furthermore, we can assume that x has not the same neighborhood as c_1 or c_p: since (P_c, P_t) is maximal, every vertex with the same neighborhood as c_1 or c_p belongs to P_c. Thus, $N(c_1) \supset N(x) \supset N(c_p)$, and, consequently, there is a vertex $t' \in N(x) \setminus N(c_p)$ and a vertex $t'' \in N(c_1) \setminus N(x)$. Clearly, $t', t'' \in T'(c_1, c_p)$ and, thus, $t', t'' \in P_t$. Since by assumption $x \notin P_c$, it holds that $x \in N(t') \setminus P_c$ but $x \notin N(t'') \setminus P_c$; a contradiction to Condition 3 of Definition 3.

Second, we show that $P_t = T'(c_1, c_p) \cup T''(c_1, c_p)$. As argued above, $T'(c_1, c_p) \subseteq P_t$ and, thus, it remains to show that $P_t \setminus T'(c_1, c_p) = T''(c_1, c_p)$. Observe that

$C_{out}(c_1, c_p) = N(t) \setminus P_c$ for each vertex $t \in P_t$ since $T'(c_1, c_p) \subseteq P_t$, $P_c = S(c_1, c_p)$, and due to Condition 3 of Definition 3. Hence, $P_t \setminus T'(c_1, c_p) \subseteq T''(c_1, c_p)$, since each vertex in $P_t \setminus T'(c_1, c_p)$ is adjacent to all vertices in $C_{out}(c_1, c_p) \cup S(c_1, c_p)$. Finally, note that $(P_c, T'(c_1, c_p) \cup T''(c_1, c_p))$ is a P-structure. Thus, $P_t = T'(c_1, c_p) \cup T''(c_1, c_p)$ by the maximality of (P_c, P_t). This concludes the proof of the above claim.

By the above claim, a P-structure (P_c, P_t) where $P_c = \{c_1, c_2, \ldots, c_p\}$ with $N(c_1) \supseteq N(c_2) \supseteq \ldots \supseteq N(c_p)$ and $N(c_1) \supset N(c_p)$ is uniquely determined by c_1 and c_p. Thus, it remains to show that computing the sets $C := S(c', c'')$ and $T := T'(c', c'') \cup T''(c', c'')$ and testing whether the found subgraph is a P-structure are doable in $O(|E|)$ time.

To compute the set $S(c', c'')$ proceed as follows. First, compute the sets $N(c')$ and $N(c'')$ and check whether $N(c'') \subseteq N(c')$. If so, in one iteration over E compute for every c-vertex x the values $a(x) := |N(c') \cap N(x)|$, $b(x) := |N(c'') \cap N(x)|$, and $c(x) := |N(x) \setminus N(c')|$. Clearly, exactly if $a(x) \leq |N(c')|$, $b(x) = |N(c'')|$, and $c(x) = 0$, then $x \in S(c', c'')$. Using similar ideas, it is straightforward to verify that the sets $T'(c', c'')$ and $T''(c', c'')$ can be computed in $O(|E|)$ time.

Finally, we show that, given a tuple (P_c, P_t), it can be decided in $O(|E|)$ time whether it forms a P-structure. To this end, proceed as follows. First, compute the sets $T' := N(P_c) \setminus P_t$ and $C' := N(P_t) \setminus P_c$. Note that then in one iteration over E it is possible to check whether $N(c) \setminus P_t = T'$ for each $c \in P_c$. Analogously, check whether $N(t) \setminus P_c = C'$ for each $t \in P_c$. If this test is passed without conflicts it remains to verify that the neighborhoods of the vertices in P_c form a chain. To this end, sort the vertices in P_c in decreasing order of their degrees using bucket sort (in linear time). Let $P_c = \{c_1, \ldots, c_p\}$ where $\deg(c_1) \geq \ldots \geq \deg(c_p)$. Clearly, by first marking the neighbors of c_i and then iterating over c_{i-1}'s neighbors, it is possible to test whether $N(c_i) \subseteq N(c_{i-1})$ in time $O(\deg(c_{i-1}))$. Hence, checking whether the neighborhoods of the vertices in P_c form a chain is doable in $O(\sum \deg(c_i)) = O(|E|)$ time. □

Lemma 8 *In $O(|V_c|^2|E|)$ time, Rule 4 can be applied once or it can be decided that the instance is reduced with respect to Rule 4.*

Proof By Lemma 7, a P-structure (P_c, P_t) such that $|P_t|$ is maximal can be computed in $O(|V_c|^2 \cdot |E|)$ time. Note that only the size of $|P_t|$ is important for the decision whether Rule 4 applies. Thus, if $|P_t| \leq 2(k+1)$ the instance is reduced with respect to Rule 4. Otherwise, Rule 4 can be applied in $O(|E|)$ time. Thus, the overall running time is $O(|V_c|^2|E|)$. □

6 Analysis of the Problem Kernel Size

In this section, we bound the maximum number of vertices in an instance that is reduced with respect to Rules 1–4.

For the analysis of the problem kernel size, we make use of the representation of M-free graphs as rooted trees. Recall that given a connected and M-free graph $G = (V_c, V_t, E)$, one can construct a rooted tree T with node set $V_t \cup V_c$ and with $L(T) = V_t$ such that $t_i \in V_t$ is a descendant of $c_j \in V_c$ if and only if $t_i \in N_G(c_j)$, see [9, 25, 33] for details.

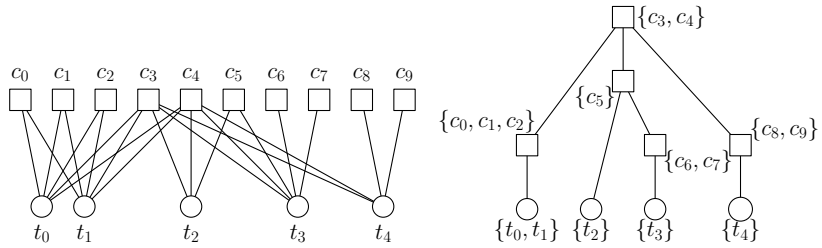


Fig. 4 An M-free graph G and the corresponding tree $T_{cis}(G)$.

Note that the critical independent set graph of an M-free graph is M-free. Hence, we can find a tree with the property that every leaf one-to-one corresponds to a critical independent set of the t-vertices and every inner vertex one-to-one corresponds to a critical independent set of the c-vertices. For a connected and M-free graph G, we denote this tree by $T_{cis}(G)$. Figure 4 shows an M-free graph G together with $T_{cis}(G)$.

The following easy observations are helpful in the analysis of the kernel size.

Observation 1 *For an M-free graph G the following holds.*

1. *Every inner node of $T_{cis}(G)$ has at most one leaf child.*
2. *Every inner node of $T_{cis}(G)$ with at most one non-leaf child has exactly one leaf child.*

Now, we arrive at our main result.

Theorem 1 *MFCT admits an $O(k^3)$-vertex problem kernel. The kernelization runs in $O(|V_c|^2 \cdot |V_t| \cdot |E|)$ time.*

Proof Consider an instance $(G = (V_c, V_t, E), k)$ that is reduced with respect to Rules 1–4. We show that if (G, k) is a yes-instance, then the number of vertices in $V_c \cup V_t$ is $O(k^3)$.

Assume that (G, k) is a yes-instance and let S denote an optimal solution of size at most k for G. Moreover, let $G_S := G \Delta S$. Recall that the vertices that are involved in any edge modification of S are called *affected*. All other vertices are called *nonaffected*. Let X_c denote the set of affected c-vertices and let Y_c denote the set of nonaffected c-vertices. Analogously, define X_t and Y_t as the sets of affected and nonaffected t-vertices, respectively. Since every edge modification involves a c-vertex and a t-vertex, we have $|X_c| \leq k$ and $|X_t| \leq k$. Hence, it remains to bound $|Y_c \cup Y_t|$.

Let $G_{S,1}, G_{S,2}, \ldots, G_{S,p}$ denote the connected components of G_S. For each i, $1 \leq i \leq p$, let $T_{cis}(G_{S,i})$ be the rooted tree corresponding to the critical independent set graph of $G_{S,i}$. Moreover, let T denote the forest comprising all $T_{cis}(G_{S,i})$'s. Recall that the leaves of T one-to-one correspond to the critical independent sets of V_t in G_S and that the inner nodes of T one-to-one correspond to the critical independent sets of V_c in G_S. For a node $z \in V(T)$, let $\mathcal{C}(z)$ denote the set of vertices contained in the critical independent set corresponding to z. Moreover, for $Z \subseteq V(T)$, define $\mathcal{C}(Z) := \bigcup_{z \in Z} \mathcal{C}(z)$. Finally, let $T' := T - L(T)$ denote the subforest of T induced by the critical independent sets corresponding to the c-vertices.

For the analysis of the kernel size, we partition the set of inner nodes into three sets A, B, and Q. The set A contains all inner nodes z of T for which at least one of the following two holds: $\mathcal{C}(z) \cap X_c \neq \emptyset$ or z has a leaf child w with $\mathcal{C}(w) \cap X_t \neq \emptyset$. Note that $|A| \leq 2k$ since there are at most $2k$ affected vertices. Moreover, let B be the set of the inner nodes that are not contained in A and that have at least two non-leaf children. Further, Q contains all inner nodes not contained in $A \cup B$. For a set of inner nodes Z, let L_Z denote the set of leaves incident with a vertex in Z.

First, we bound the number of the vertices contained in the critical independent sets corresponding to the nodes in $A \cup B$ and $L_A \cup L_B$. To this end, we show the following.

1. For every inner node $x \in B \cup Q$, there exists at least one node $y \in V(T_x)$ with $y \in A$ (recall that T_x denotes the maximal subtree of T rooted at v).
2. The cardinality of B is at most $2k$.
3. The number of vertices contained in the critical independent sets corresponding to the nodes in $A \cup B \cup L_A \cup L_B$ is $O(k^2)$.

1.) Assume towards a contradiction that there is an inner node $x \in B \cup Q$ such that $V(T_x) \cap A = \emptyset$. That is, no vertex in $\mathcal{C}(V(T_x))$ is affected. Consider a vertex $c' \in \mathcal{C}(x)$. We show that c' is not contained in any conflict in G, contradicting the fact that G is reduced with respect to Rule 3. First, for every c-vertex y in $\mathcal{C}(V(T_x))$ it holds that $N_G(y) \subseteq N_G(c')$ since $N_{G_S}(y) \subseteq N_{G_S}(c')$ and S does not affect c or y. Second, for every c-vertex y in $\mathcal{C}(V(T) \setminus V(T_x))$ it holds that $N_{G_S}(c') \cap N_{G_S}(y) = \emptyset$ or $N_{G_S}(c') \subseteq N_{G_S}(y)$. As a consequence, since neither c' nor any t-vertex in $N_{G_S}(c')$ is affected, it follows that $N_G(c') \cap N_G(y) = \emptyset$ or $N_G(c') \subseteq N_G(y)$. This means that c' is not contained in any conflict in G.

2.) Recall that the forest T' results from deleting all leaves of T. Since each node of B has at least two non-leaf children in T, it has at least two children in T'. From 1.) it follows directly that the leaves of T' are contained in A and, hence, their number is at most $2k$. Since the number of inner nodes with at least two children is bounded by the number of leaves, we arrive at the bound $|B| \leq 2k$.

3.) First, note that $|A \cup B| \leq 4k$ since A and B each have cardinality at most $2k$. Moreover, $|L_A \cup L_B| \leq 4k$ since every inner node has at most one leaf child (see Observation 1). For every node $y \in A \cup B \cup L_A \cup L_B$, define $\mathcal{C}'(y) := \mathcal{C}(y) \setminus (X_c \cup X_t)$ and observe that $\mathcal{C}'(y)$ is part of a critical independent set in G since no vertex in $\mathcal{C}'(y)$ is affected. Thus, since G is reduced with respect to Rule 1, it follows that $|\mathcal{C}'(y)| \leq k + 1$. Putting all together, we obtain

$$|\mathcal{C}(A \cup B \cup L_A \cup L_B)| \leq |X_c| + |X_t| + \sum_{y \in A \cup B \cup L_A \cup L_B} |\mathcal{C}'(y)| \leq 2k + 8k(k+1).$$

So far, we have bounded the number of vertices in $X_c \cup X_t$ and $\mathcal{C}(A \cup B \cup L_A \cup L_B)$ by $O(k^2)$. It remains to bound the number of vertices contained in $\mathcal{C}(Q \cup L_Q)$.

Observe that each inner node contained in Q (and hence not contained in $A \cup B$) has exactly one non-leaf child since $L(T') \subseteq A$ (see 2.) above). That is, in the forest $T' = T - L(T)$ these vertices have degree two. Moreover, by Observation 1, each node contained in Q has exactly one leaf child. Recall that all leaves of T' are contained in A and hence $|L(T')| \leq 2k$. Consider a path $P = (\{x, y_1\}, \{y_1, y_2\}, \ldots, \{y_{l-1}, y_l\}, \{y_l, z\})$ in T' with $y_i \in Q$ for all $1 \leq i \leq l$ and $x, z \in A \cup B$. Such a path is called a *degree-two-path* in the following since by the above

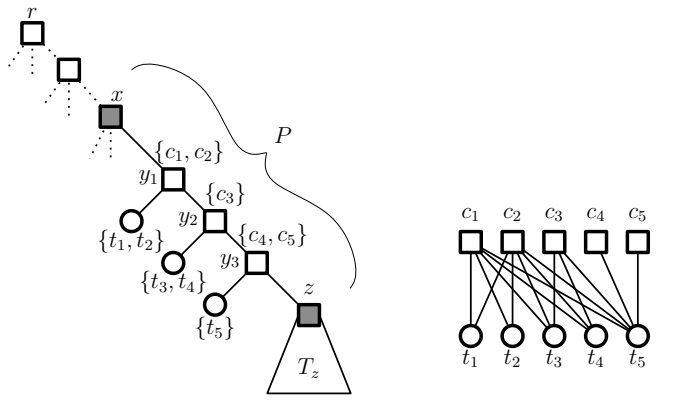


Fig. 5 A degree-two-path P and the corresponding chain graph. Herein, $\mathcal{C}(y_1) = \{c_1, c_2\}$, $\mathcal{C}(y_2) = \{c_3\}$, $\mathcal{C}(y_3) = \{c_4, c_5\}$, and $\mathcal{C}(y_4) = \{c_6\}$.

discussion $\deg_{T'}(y_i) = 2$ for all $1 \leq i \leq l$. Further, for every y_i, let w_i denote the leaf child of y_i in T. Note that in the forest T', there are at most $8k$ degree-two-paths since $L(T') \subseteq A$ and $|A \cup B| \leq 4k$. In the following, we bound the length of each degree-two-path by $2(k+1)$. Hence, for each such path we have

$$\sum_{i=1}^{l} (|\mathcal{C}(y_i)| + |\mathcal{C}(w_i)|) \leq l \cdot (2(k+1)) \leq (2(k+2)) \cdot 2(k+1)$$

vertices in G. Adding up over the at most $8k$ degree-two-paths, this amounts to $8k \cdot 2(k+1)(2(k+2)) \leq 32k(k+1)(k+2)$ vertices, yielding the bound of $O(k^3)$ vertices in total.

Next, we bound the length of each degree-two-path. To this end, consider such a degree-two-path $P = (\{x, y_1\}, \{y_1, y_2\}, \ldots, \{y_{l-1}, y_l\}, \{y_l, z\})$ in T', that is, $x, z \in A \cup B$ and $y_i \in Q$ for all $1 \leq i \leq l$. Without loss of generality, we assume that y_l is a descendant of y_1. For each y_i let w_i denote the one and only leaf child adjacent to y_i. See Figure 5 for an example. Let $P_c := \bigcup_{i=1}^{l} \mathcal{C}(y_i)$ and $P_t := \bigcup_{i=1}^{l} \mathcal{C}(w_i)$. We show that (P_c, P_t) forms a P-structure in G. First, note that $P_c \subseteq V_c$ and $P_t \subseteq V_t$. Recall that by definition all vertices in $P_c \cup P_t$ are unaffected. Next, observe that $G[P_c \cup P_t]$ forms a chain graph. This can be seen as follows. In G_S a vertex in $\mathcal{C}(y_1)$ is clearly adjacent to all vertices in P_t, a vertex in $\mathcal{C}(y_2)$ is adjacent to all vertices in $P_t \setminus \mathcal{C}(w_1)$, a vertex in $\mathcal{C}(y_3)$ is adjacent to all vertices in $P_t \setminus \mathcal{C}(\{w_1, w_2\})$, and so on. Hence, $G_S[P_c \cup P_t]$ is a chain graph and, since no vertex in P_c is involved in an edge modification, we have that $G[P_c \cup P_t]$ forms a chain graph, too (see Figure 5). Next, we show that P_c and P_t fulfill the second and third property of a P-structure. First, every vertex in P_c is adjacent in G_S to all vertices contained in the critical independent sets corresponding to the leaves in T_z and, hence, for all $c, c' \in P_c$, we have $N_{G_S}(c) \setminus P_t = N_{G_S}(c') \setminus P_t$. Since no vertex in P_c is affected, this implies that $N_G(c) \setminus P_t = N_G(c') \setminus P_t$ for all $c, c' \in P_c$. Second, every vertex $t \in P_t$ is adjacent in G_S (and hence in G) to all c-vertices contained in a critical independent set on the path from the root r to z. Hence, for any two vertices $t, t' \in P_t$ it holds that $N_G(t) \setminus P_t = N_G(t') \setminus P_t$. In summary, (P_c, P_t) forms a P-structure.

Finally, we show that $l \leq 2(k+1)$. Assume towards a contradiction that $l > 2(k+1)$. This implies that $|P_t| > 2(k+1)$, too, since every y_i has exactly one leaf

child that corresponds to a critical independent set of V_t. Hence, $|P_t| > 2(k+1)$ and thus all conditions to apply Rule 4 are fulfilled: a contradiction to the fact that G is reduced.

To prove the running time bound, we assume that the data reduction rules are applied as follows. First, we show that an instance reduced with respect to the Rules 3 and 4 can be computed in $O(|V_c|^2 \cdot |V_t| \cdot |E|)$ time. To this end, proceed as follows. Apply Rule 3 exhaustively. Then, check whether Rule 4 can be applied. If not, the instance is reduced with respect to both rules. Otherwise, apply Rule 3 exhaustively again and subsequently check whether Rule 4 can be applied, and so on. Note that Rule 4 is applied $O(|V_t|)$ times since each time (except for the last time) at least one t-vertex is removed. Hence, by Lemmas 5 and 8 one obtains a graph that is reduced with respect to Rules 3 and 4 in $O(|V_t| \cdot (|V_c|^2|V_t| + |V_c|^2|E|)) = O(|V_c|^2|V_t||E|)$ time.

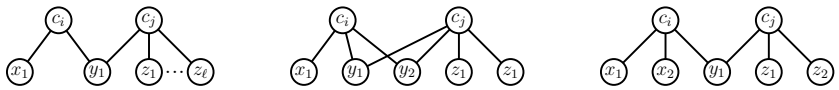
Finally, observe that applying Rule 2 and subsequently Rule 1 to a graph reduced with respect to Rules 3 and 4 leaves a graph reduced with respect to Rules 3 and 4. Thus, the running time is dominated by the application of Rules 3 and 4. Hence, the kernelization runs in $O(|V_c|^2 \cdot |V_t| \cdot |E|)$ time. □

7 An $O(3.68^k)$-size search tree

In this section, we present an improved search tree algorithm for Minimum-Flip Consensus Tree. Basically, we use the same branching strategy as Böcker et al. [5]. Our main achievement is that exploiting Proposition 1 allows us to stop the branching process at an earlier stage.

For the presentation of our results, we use the following notation. Following Böcker et al. [5], for two c-vertices $c_i, c_j \in V_c$ we define $X(c_i, c_j) := N(c_i) \setminus N(c_j)$, $Y(c_i, c_j) := N(c_i) \cap N(c_j)$, and $Z(c_i, c_j) := N(c_j) \setminus N(c_i)$. Note that $c_i \perp c_j$ if and only if all three sets are nonempty. We use the concept of branching rules for the presentation of our search tree algorithm. Given an instance (G, k), a branching rule creates $\ell \geq 1$ subinstances $(G_1, k_1), \ldots, (G_\ell, k_\ell)$. A branching rule is correct if (G, k) is a yes-instance if and only if (G_i, k_i) is a yes-instance for some $1 \leq i \leq \ell$. Branching rules lead to a search algorithm by solving each of the created subinstances recursively, terminating the recursion when $k \leq 0$ or none of the branching rules applies. For a branching rule creating $\ell \geq 2$ subinstances, a branching vector is an ℓ-tuple describing how the parameter is decreased in each subinstance. Using standard branching analysis tools, a branching number can be computed from the branching vector [31]. The branching number describes the base of the (exponential) search tree size.

Böcker et al. [5] introduced a branching rule that, for two conflicting c-vertices c_i and c_j, branches into all possibilities to make one of the sets $X(c_i, c_j)$, $Y(c_i, c_j)$, and $Z(c_i, c_j)$ empty. To this end, they branch into $2^{|X(c_i,c_j)|} + 2^{|Y(c_i,c_j)|} + 2^{|Z(c_i,c_j)|}$ cases. In each of the first $2^{|X(c_i,c_j)|}$ branches, parameter k is decreased by $|X(c_i, c_j)|$, in the each of the subsequent $2^{|Y(c_i,c_j)|}$ branches parameter k is decreased by $|Y(c_i, c_j)|$, and in each of the last $2^{|Z(c_i,c_j)|}$ cases parameter k is decreased by $|Z(c_i, c_j)|$. See [5] for any details and the correctness of this branching strategy. For example, if $|X(c_i, c_j)| = 3$, $|Y(c_i, c_j)| = 2$ and $|Z(c_i, c_j)| = 1$ the branching vector is $(3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 2, 1, 1)$ leading to a branching number of 3.68. Indeed, using standard branching analysis tools it is easy to verify that the branching



a) *b)* *c)*

Fig. 6 The three cases for which new branching rules are designed.

number is at most 3.68 if the size of one of these sets is at least 3 and the size of two of these sets is at least 2. Moreover, if $|X(c_i, c_j)| = |Y(c_i, c_j)| = |Z(c_i, c_j)| = 2$, then the branching vector is $(2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2)$ giving a branching number of 3.47. Further, one can verify that the branching number is at most 3.47 if all of these sets have size at least two.

The basic idea behind the improved search tree algorithm is as follows. We will show that if for all conflicting c-vertices c_i and c_j it holds that $|X(c_i, c_j)| = |Z(c_i, c_j)| = 1$, then MFCT can be solved in polynomial time. Moreover, for all other cases we present branching rules with corresponding branching numbers better than 3.68. To this end, we use the branching strategy introduced by Böcker as long as it yields a branching number better than 3.68. For the remaining cases we devise new, refined branching strategies.

Altogether, we use the following branching rules.

1. If there are two conflicting c-vertices c_i, c_j such at least one of the three sets $X(c_i, c_j)$, $Y(c_i, c_j)$, and $Z(c_i, c_j)$ has cardinality three and at least two of the three sets $X(c_i, c_j)$, $Y(c_i, c_j)$, and $Z(c_i, c_j)$ have cardinality at least two, or if all three sets $X(c_i, c_j)$, $Y(c_i, c_j)$, and $Z(c_i, c_j)$ have cardinality at least two, then apply the branching strategy by Böcker et al. [5].
2. If there are two conflicting c-vertices c_i, c_j with $|X(c_i, c_j)| = |Y(c_i, c_j)| = 1$ and $|Z(c_i, c_j)| > 1$ (that is, c_i has degree two), then proceed as follows. See Figure 6 *a*) for notation. First, branch into the case to delete the edge $\{x_1, c_i\}$. Second, branch into the case to add the edge $\{x_1, c_j\}$. Third, branch into the case to delete the edge $\{y_1, c_j\}$. Fourth, branch into the case to delete all edges $\{\{z_p, c_i\} \mid 1 \leq p \leq \ell\}$.
3. If there are two conflicting c-vertices c_i, c_j with $|X(c_i, c_j)| = 1$ and $|Y(c_i, c_j)| = |Z(c_i, c_j)| = 2$, then proceed as follows. See Figure 6 *b*) for notation. First, branch into the case to delete $\{x_1, c_i\}$. Second, branch into the case to add $\{x_1, c_j\}$. Third, branch into the three cases to delete each time the two edges $S_{r,s} := \{\{y_1, c_r\}, \{y_2, c_s\}\}$ for $r \in \{i, j\}$ and $s \in \{i, j\}$ with $r = j$ or $s = j$. Fourth, branch into the three cases to modify each time the two edges $S_{r,s} := \{\{z_1, c_r\}, \{z_2, c_s\}\}$ for $r \in \{i, j\}$ and $s \in \{i, j\}$ with $r = j$ or $s = j$.
4. If there are two conflicting c-vertices c_i, c_j with $|X(c_i, c_j)| = 2$, $|Y(c_i, c_j)| = 1$, and $|Z(c_i, c_j)| = 2$, then proceed as follows. See Figure 6 *c*) for notation. First, branch into the case to delete $\{y_1, c_i\}$. Second, branch into the case to delete $\{y_1, c_j\}$. Third, branch into the two cases to modify each time the two edges $S_{r,s} := \{\{x_1, c_s\}, \{x_2, c_r\}\}$ for $r \in \{i, j\}$ and $s \in \{i, j\}$ with $r \neq s$. Fourth, branch into the two cases to modify each time the two edges $S_{r,s} := \{\{z_1, c_s\}, \{z_2, c_r\}\}$ for $r \in \{i, j\}$ and $s \in \{i, j\}$ with $r \neq s$.

In all branching rules, the parameter is decreased by the number of modified edges in each branch.

The correctness of the new branching rules is based on the simple observation that a degree-one c-vertex is not part of any induced M-graph, and, hence, can be deleted according to Rule 3. This observation implies two important properties of optimal solutions. First, every c-vertex c' is involved in at most $\deg(c') - 1$ edge modifications. Second, if a c-vertex c' is involved in $\deg(c') - 1$ edge modifications, then we can assume that all these edge modifications are edge deletions and, moreover, one can delete $\deg(c') - 1$ arbitrary edges incident with c'. Summarizing, we obtain the following.

Observation 2 *Let $c' \in V_c$ and let $t \in N(c')$. Every optimal solution contains at most $\deg_G(c') - 1$ edge modifications that are incident with c'. If there is a solution S containing at least $\deg_G(c') - 1$ edge modifications incident with c', then there is a solution S', $|S'| \leq |S|$ with $\{c', t\} \in S$.*

Lemma 9 *The above branching rules are correct. The branching number of all branching rules is bounded from above by* 3.68.

Proof For the correctness of Branching Rule 1 see Böcker et al [5]. The correctness of the other three branching rules is based on Observation 2. If $|X(c_i, c_j)| = |Y(c_i, c_j)| = |Z(c_i, c_j)|$, the branching number is at most 3.47 and if one of these sets has cardinality three, one has cardinality two, and the one has cardinality one, then the branching number is at most 3.68. Hence, the branching number of Branching Rule 1 is at most 3.68.

Correctness of Branching Rule 2. By Observation 2, if there is a solution that contains an edge modification involving c_i, then we can assume that $\{c_i, x_1\}$ is deleted. The correctness follows by the fact that the branching rule branches into all cases to make one of the sets $X(c_i, c_j)$, $Y(c_i, c_j)$, and $Z(c_i, c_j)$ empty, omitting the cases that c_i is involved into a edge modification other than the deletion of $\{c_i, x_1\}$. The branching vector of Branching Rule 2 is $(1, 1, 1, x)$ with $x > 1$. Hence, the branching number is bounded by 3.31.

Correctness of Branching Rule 3. By Observation 2, if there is a solution containing $\{\{y_1, c_i\}, \{y_2, c_i\}\}$ or $\{\{z_1, c_i\}, \{z_2, c_i\}\}$, then there exists a solution containing $\{x_1, c_i\}$. The case to that $\{x_1, c_i\}$ is contained in the solution is considered by the rule. Hence, the correctness of the rule follows from the fact that the branching rule branches into all cases to make one of the sets $X(c_i, c_j)$, $Y(c_i, c_j)$, and $Z(c_i, c_j)$ empty, omitting the cases that c_i is involved into two edge modification (not containing $\{c_i, x_1\}$). The branching vector of Branching Rule 3 is $(1, 1, 2, 2, 2, 2, 2, 2)$, giving a branching number of 3.65.

Correctness of Branching Rule 4. By Observation 2, if there is a solution containing $\{\{x_1, c_i\}, \{x_2, c_i\}\}$, $\{\{z_1, c_i\}, \{z_2, c_i\}\}$, $\{\{x_1, c_j\}, \{x_2, c_j\}\}$, or $\{\{z_1, c_j\}, \{z_2, c_j\}\}$, then there is a solution containing either $\{y_1, c_i\}$ or $\{y_1, c_j\}$. Hence, these four cases must not considered by branching rule 4. The branching vector of Branching Rule 4 is $(1, 1, 2, 2, 2, 2)$, leading to a branching number of 3.24. □

Next, we show that, by exploiting Proposition 1, an instance to which none of these branching rules applies can be solved in polynomial time. A graph fulfilling the property that for any two conflicting c-vertices c_i and c_j it holds that $|X(c_i, c_j)| = 1$ and $|Z(c_i, c_j)| = 1$, is called *conflict regular* in the following. It is easy to observe that if none of the four branching rules applies, then the instance is conflict regular. We show that for conflict regular graphs MFCT can be solved in polynomial time.

An easy observation is that in a conflict regular graph any two conflicting vertices have the same degree. The next lemmas describe the structure of conflict regular graphs in more detail.

Lemma 10 *Let $G = (V_c, V_t, E)$ denote a conflict regular graph. Let c_1 and c_2 denote two conflicting c-vertices of degree at least three. Let $c' \in V_c$ with $N(c') \neq N(c_1)$. If c' is in conflict with c_2, then c' is in conflict with c_1.*

Proof Consider a vertex c' with $c' \perp c_2$. Assume towards a contradiction that c' is not in conflict with c_1. Since G is conflict regular, all three vertices have the same degree and $N(c') \cap Y(c_1, c_2) \neq \emptyset$. Thus, since c' is not in conflict with c_1, it holds that $N(c') \subseteq N(c_1)$ or $N(c_1) \subseteq N(c')$. Since $N(c') \neq N(c_1)$, this is a contradiction to the fact that all three vertices have the same degree. □

Lemma 11 *Let $G = (V_c, V_t, E)$ denote a conflict regular graph and let C denote a connected component of $G_\perp$ with $|C| \geq 2$ and $\deg_G(c) \geq 3$ for all $c \in C$. Then, one of the following statements holds.*

1. *There is a set $Y \subseteq N_G(C)$ such that for any two vertices $c, c' \in C$ with $N_G(c) \neq N_G(c')$ it holds that $Y = Y(c, c')$.*
2. *For all $c \in C$, it holds that $|N_G(C) \setminus N_G(c)| = 1$.*

Proof Let $X_1, \ldots, X_\ell$ denote the critical independent sets of G formed by the vertices in C. Note that Lemma 10 implies that any two vertices from different X_i's are in conflict. Moreover, observe that if $\ell = 2$, then the first statement of the lemma holds. Hence, in the following, we focus on the case $\ell > 2$.

Let $c_1 \in X_1$, $c_2 \in X_2$. Moreover, let $C' := \{c \in C \setminus (X_1 \cup X_2) \mid Y(c, c_1) = Y(c, c_2) = Y(c_1, c_2)\}$.

First, consider the case that $C' \neq \emptyset$. Assume without loss of generality that $C' = \bigcup_{i=3}^{s} X_i$ for some $s \geq 3$. We show that $s = \ell$ and, hence, the first statement of the lemma holds for $Y = Y(c_1, c_2)$. To this end, let $c_3 \in X_3$ and observe that $|N_G(c_i) \setminus (N_G(c_j) \cup N_G(c_k))| = 1$ for any three distinct $i, j, k \in \{1, 2, 3\}$. Hence, $|N(c_1) \cup N(c_2) \cup N(c_3)| \geq d + 2$, where d is the degree of the vertices in C (recall that all vertices in C have the same degree). Assume towards a contradiction that $s < \ell$ and let $c_\ell \in X_\ell$. By inductively applying Lemma 10 it follows that c_ℓ is in conflict with each of c_1, c_2, and c_3. Note that since $c_\ell \notin C'$ and G is conflict regular it holds that $|Y(c_1, c_2) \setminus N(c_\ell)| = 1$ and, except for this vertex, c_ℓ is adjacent to all vertices in $N(c_1) \cup N(c_2) \cup N(c_3)$; a contradiction to the fact that $\deg_G(c_\ell) = d$.

Second, consider the case that $C' = \emptyset$. That is, for every vertex $c' \in \bigcup_{i=3}^{\ell} X_i$ it holds that $Y(c_1, c_2) \setminus N_G(c') \neq \emptyset$. By Lemma 10, c' is in conflict with both c_1 and c_2 and, hence, $\deg(c_1) = \deg(c_2) = \deg(c')$. Moreover, since G is conflict regular, c' is adjacent to all but one vertex in $N_G(c_1)$ and $N_G(c_2)$. Altogether, this implies that $N(c') \subseteq N(c_1) \cup N(c_2)$ since c' is nonadjacent to a common neighbor of c_1 and c_2. Hence, $N_G(C) \subseteq N_G(c_1) \cup N_G(c_2)$ and since all vertices in C have the same degree, the second statement of the lemma holds. □

Proposition 2 *MFCT can be solved in $O(|V_c|^2 \cdot |V_t|)$ time for conflict regular graphs.*

Proof Let $(G = (V_t, V_c, E), k)$ denote an MFCT-instance where G is a conflict regular graph. Moreover, let $G_\perp$ denote the conflict graph of G. By Proposition 1,

we can resolve the conflicts for every connected component of $G_\perp$ independently. Let C denote a connected component of $G_\perp$. Clearly, all vertices of C have the same degree in G. If all vertices have degree two, one can resolve all conflicts between vertices in C by the computation of a maximum weight matching [5].

Hence, in the following we focus on the case that $\deg_G(c) = d \geq 3$ for all $c \in C$. We use the following notation. Let $G' := G[C \cup N_G(C)]$. Moreover, let $X_1, \ldots, X_\ell$ denote the critical independent sets of G' formed by the vertices in C and assume without loss of generality that $|X_1| \leq \ldots \leq |X_\ell|$. We distinguish the cases that either the first or the second statement of Lemma 11 is true. In both cases we show how to construct a regular optimal solution for G'.

Case 1: The first statement of Lemma 11 holds**.** That is, there is a set Y of t-vertices such that for any two vertices $c, c' \in C$ with $N_G(c) \neq N_G(c')$, we have $Y(c, c') = Y$. Note that $|Y| = d - 1 > 1$ and $|N(c) \setminus Y| = 1$ for all $c \in C$. The structure of G' is depicted in Figure 7. To resolve the conflicts between the vertices in C delete the edges $S' := \bigcup_{i=1}^{\ell-1} E_{X_i, N(X_i)\setminus Y}$. Note that $|S'| = \sum_{i=1}^{\ell-1} |X_i|$ since $|N(X_i) \setminus Y| = 1$. Since $N_{G'\Delta S'}(X_i) = Y$ for all $1 \leq i \leq \ell - 1$ and $Y \subset N_{G'\Delta S'}(X_\ell)$, all M-graphs in $G' := G[C \cup N(C)]$ are destroyed by S'. Finally, we argue that S' is a regular optimal solution for G'. Observe that by construction for each X_i any two vertices of X_i have an identical neighborhood in $G'\Delta S'$. In this sense, S' applies the "same" edge modifications to all vertices in X_i and, hence, is regular. Moreover, since two vertices from different X_i's are in conflict, we have to modify the neighborhood structure of all but one X_i. Hence, for any X_i that is affected, one has to spend at least $|X_i|$ modifications (recall that each X_i is a critical independent set and, hence, there is an optimal solution that applies the "same" edge modifications to all vertices in X_i, see Lemma 2). Thus, $\sum_{i=1}^{\ell-1} |X_i| = |S'|$ is a lower bound for the solution size since $|X_\ell| \geq |X_i|$ for all $1 \leq i \leq \ell - 1$.

Case 2: The second statement of Lemma 11 holds**.** That is, $|N(C) \setminus N(c)| = 1$ for all $c \in C$. To resolve the conflicts between the vertices in C, add the edges $S' := \bigcup_{i=1}^{\ell-1} E_{X_i, N(C)\setminus N(X_i)}$. Note that $|S'| = \sum_{i=1}^{\ell-1} |X_i|$ since $|N(C) \setminus N(X_i)| = 1$. Since $N_{G'\Delta S'}(X_i) = N_G(C)$ for all $1 \leq i \leq \ell - 1$ and $N_{G'\Delta S'}(X_\ell) \subset N_G(C)$ all conflicts within C are resolved. Analogously to Case 1, one can show that S' is a regular optimal solution for G': Since two vertices from different X_i's are in conflict, we have to modify the neighborhood structure of all but one X_i (requiring at least $|X_i|$ edge modifications). Thus, $\sum_{i=1}^{\ell-1} |X_i|$ is a lower bound for the solution size.

For the running time note the following. The conflict graph $G_\perp$ can be built in $O(|V_c|^2 \cdot |V_t|)$ time. Moreover, when building $G_\perp$ one can simultaneously check whether the input graph is conflict regular.

Clearly, $G_\perp$ has $|V_c|$ vertices and at most $|V_c|^2$ edges. Hence, the connected components of $G_\perp$ can be found in $O(|V_c|^2)$ time. Let $C_1, \ldots, C_\ell$ denote the connected components of $G_\perp$. Clearly, for each C_i all changes can be applied in $O(|C_i| \cdot |V_t|)$ time. Hence, the total running time is bounded by $O(|V_c|^2 \cdot |V_t|)$. □

Combining Lemma 9 and Proposition 2 we arrive at the main result of this section.

Theorem 2 Minimum-Flip Consensus Tree *can be solved in* $O(3.68^k \cdot |V_c|^2|V_t|)$ *time.*

Proof To solve MFCT apply the above branching rules as long as possible. Note that in $O(|V_c|^2 \cdot |V_t|)$ time one can check whether one of the branching rules applies

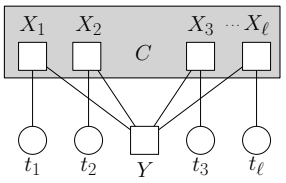


Fig. 7 The structure of $G' := G[C \cup N(C)]$ if the first statement of Lemma 11 holds. The rectangular vertices represent critical independent sets of G', the circular vertices represent single vertices, and an edge represents all edges between the (sets) of vertices. Clearly, it is optimal to delete the edges between X_i and t_i for all but one X_i for which $|X_i|$ is maximal to destroy all M-graphs in G'.

and to apply the respective rule. Moreover, if none of the branching rules apply, then the instance is conflict regular and, hence, can be solved in $O(|V_c|^2 \cdot |V_t|)$ time according to Proposition 2. Hence, for each search-tree node, all changes can be applied in $O(|V_c|^2 \cdot |V_t|)$ time. According to Lemma 9 this leads to a search-algorithm with running time $O(3.68^k \cdot |V_c|^2|V_t|)$. □

Applying the technique of interleaving [32] to our kernelization and the search tree algorithm, we obtain an "additive FPT" algorithm for MFCT.

Corollary 1 *MFCT can be solved in* $O(3.68^k + |V_c|^2 \cdot |V_t| \cdot |E|)$ *time.*

8 Outlook

There are numerous tasks for future research. Improving the polynomial running time of our data reduction rules is desirable. Obviously, obtaining data reduction rules that lead to a quadratic-vertex or linear-vertex kernel remains as an open question. Moreover, studying edge-weighted problem variants would be theoretically interesting. Furthermore, it would be interesting to adapt our data reduction to yield a full kernel (see [12]) for Minimum-Flip Consensus Tree.

Finally, recall that Minimum-Flip Consensus Tree is the problem to destroy—by a minimum number of edge modifications—all induced paths on five vertices (so-called P_5's) with the first vertex from V_t. For general graphs it has been recently shown that the problem to destroy all P_4's, called P_4-free Editing (also known as Cograph Editing), admits a cubic vertex kernel whereas for $l \geq 7$ it is very unlikely that P_l-free Editing admits a polynomial-size problem kernel [20]. To the best of our knowledge it is open whether P_5-free Editing in general graphs and P_6-free Editing in bipartite graphs admit polynomial-size problem kernels.

Acknowledgments. We are grateful to Rolf Niedermeier, Jiong Guo, and to the anonymous reviewers of FSTTCS '08 and *Algorithmica* for discussions and comments improving the presentation of this work.

References

1. F. N. Abu-Khzam and H. Fernau. Kernels: Annotated, proper and induced. In *Proceedings of the Second International Workshop on Parameterized and Exact*

Computation (IWPEC '06), volume 4169 of *Lecture Notes in Computer Science*, pages 264–275. Springer, 2006.
2. S. Bessy, C. Paul, and A. Perez. Polynomial kernels for 3-leaf power graph modification problems. *Discrete Applied Mathematics*, 158(16):1732–1744, 2010.
3. O. Bininda-Emonds, editor. *Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life*. Kluwer Academic, 2004.
4. S. Böcker, Q. B. A. Bui, F. Nicolas, and A. Truss. Intractability of the minimum-flip supertree problem and its variants. Technical report, Cornell University Library, 2011. arXiv:1112.4536v1.
5. S. Böcker, Q. B. A. Bui, and A. Truss. Improved fixed-parameter algorithms for minimum-flip consensus trees. *ACM Transactions on Algorithms*, 8(1):7:1–7:17, 2012.
6. H. L. Bodlaender. Kernelization: New upper and lower bound techniques. In *Proceedings of the 4th International Workshop on Parameterized and Exact Computation (IWPEC '09)*, volume 5917 of *Lecture Notes in Computer Science*, pages 17–37. Springer, 2009.
7. P. Burzyn, F. Bonomo, and G. Durán. NP-completeness results for edge modification problems. *Discrete Applied Mathematics*, 154(13):1824–1844, 2006.
8. D. Chen, O. Eulenstein, D. Fernández-Baca, and J. G. Burleigh. Improved heuristics for minimum-flip supertree construction. *Evolutionary Bioinformatics*, 2:347–356, 2006.
9. D. Chen, O. Eulenstein, D. Fernández-Baca, and M. Sanderson. Minimum-flip supertrees: Complexity and algorithms. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(2):165–173, 2006. Preliminary version presented at *COCOON '02*.
10. J. Chen and J. Meng. A $2k$ kernel for the cluster editing problem. *Journal of Computer and System Sciences*, 78(1):211–220, 2012.
11. M. Chimani, S. Rahmann, and S. Böcker. Exact ILP solutions for phylogenetic minimum flip problems. In *Proceedings of the First ACM International Conference on Bioinformatics and Computational Biology (ACM-BCB '10)*, pages 147–153. ACM, 2010.
12. P. Damaschke. Parameterized enumeration, transversals, and imperfect phylogeny reconstruction. *Theoretical Computer Science*, 351(3):337–350, 2006.
13. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.
14. M. R. Fellows. The lost continent of polynomial time: Preprocessing and kernelization. In *Proceedings of the Second International Workshop on Parameterized and Exact Computation (IWPEC '06)*, volume 4169 of *Lecture Notes in Computer Science*, pages 276–277. Springer, 2006.
15. M. R. Fellows, J. Guo, C. Komusiewicz, R. Niedermeier, and J. Uhlmann. Graph-based data clustering with overlaps. *Discrete Optimization*, 8(1):2–17, 2011.
16. M. R. Fellows, M. A. Langston, F. A. Rosamond, and P. Shaw. Efficient parameterized preprocessing for cluster editing. In *Proceedings of the 16th International Symposium on Fundamentals of Computation Theory (FCT '16)*, volume 4639 of *Lecture Notes in Computer Science*, pages 312–321. Springer, 2007.
17. J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.
18. J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Graph-modeled data clustering: Exact algorithms for clique generation. *Theory of Computing Systems*, 38(4):373–392, 2005.

19. J. Gramm, A. Nickelsen, and T. Tantau. Fixed-parameter algorithms in phylogenetics. *The Computer Journal*, 51(1):79–101, 2008.
20. S. Guillemot, F. Havet, C. Paul, and A. Perez. On the (non-)existence of polynomial kernels for P_l-free edge modification problems. *Algorithmica*, 2012. To appear.
21. J. Guo. Problem kernels for NP-complete edge deletion problems: split and related graphs. In *Proceedings of the 18th International Symposium on Algorithms and Computation (ISAAC '07)*, volume 4835 of *Lecture Notes in Computer Science*, pages 915–926. Springer, 2007.
22. J. Guo. A more effective linear kernelization for Cluster Editing. *Theoretical Computer Science*, 410(21–23):2045–2053, 2009.
23. J. Guo, C. Komusiewicz, R. Niedermeier, and J. Uhlmann. A more relaxed model for graph-based data clustering: s-plex cluster editing. *SIAM Journal on Discrete Mathematics*, 24(4):1662–1683, 2010.
24. J. Guo and R. Niedermeier. Invitation to data reduction and problem kernelization. *ACM SIGACT News*, 38(1):31–45, 2007.
25. D. Gusfield. Efficient algorithms for inferring evolutionary trees. *Networks*, 21:19–28, 1991.
26. W. Hsu and T. Ma. Substitution decomposition on chordal graphs and applications. In *Proceedings of the 2nd International Symposium on Algorithms (ISA '91)*, volume 557 of *Lecture Notes in Computer Science*, pages 52–60. Springer, 1991.
27. F. Hüffner, R. Niedermeier, and S. Wernicke. Techniques for practical fixed-parameter algorithms. *The Computer Journal*, 51(1):7–25, 2008.
28. S. Kratsch and M. Wahlström. Two edge modification problems without polynomial kernels. In *Proceedings of the 4th International Workshop on Parameterized and Exact Computation (IWPEC '09)*, volume 5917 of *Lecture Notes in Computer Science*, pages 264–275. Springer, 2009.
29. R. M. McConnell and J. Spinrad. Linear-time modular decomposition and efficient transitive orientation of comparability graphs. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '94)*, pages 536–545. ACM/SIAM, 1994.
30. A. Natanzon, R. Shamir, and R. Sharan. Complexity classification of some edge modification problems. *Discrete Applied Mathematics*, 113(1):109–128, 2001.
31. R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Number 31 in Oxford Lecture Series in Mathematics and Its Applications. Oxford University Press, 2006.
32. R. Niedermeier and P. Rossmanith. A general method to speed up fixed-parameter-tractable algorithms. *Information Processing Letters*, 73:125–129, 2000.
33. I. Pe'er, T. Pupko, R. Shamir, and R. Sharan. Incomplete directed perfect phylogeny. *SIAM Journal on Computing*, 33(3):590–607, 2004.
34. F. Protti, M. D. da Silva, and J. L. Szwarcfiter. Applying modular decomposition to parameterized cluster editing problems. *Theory of Computing Systems*, 44(1):91–104, 2009.
35. J. Uhlmann. *Multivariate Algorithmics in Biological Data Analysis*. Universitätsverlag der TU Berlin, 2011. Phd Thesis.
36. M. Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM Journal on Algebraic Discrete Methods*, 2(1):77–79, 1981.