# Kernelization algorithms for graph and constraint modification problems *

Anthony Perez
Under the supervision of Stéphane Bessy and Christophe Paul

September 4, 2011

**Abstract**

In this extended abstract, we give an overview of the techniques and results introduced in the thesis entitled *Algorithmes de noyau pour des problèmes d'éditions de graphes et autres structures* [58]. This work has been done under the supervision of Stéphane Bessy and Christophe Paul, at LIRMM (Montpellier, France). The results developed in this thesis belong to the framework of *parameterized complexity*, and more precisely *kernelization algorithms*. Given an instance $(I, k)$ of a parameterized problem, a kernelization algorithm is a polynomial-time algorithm (in $|I|+k$) that outputs an equivalent instance $(I', k')$ (the *kernel*) that satisfies $|I| \leqslant f(k)$ and $k' \leqslant k$ for some computable function $f$. Such a function is called the *size* of the kernel. We focus on *graph (or constraint) modification problems*, where one is given a graph $G := (V, E)$ (resp. a set of constraints $\mathcal{R}$ defined over some universe $V$) and an integer $k$ and seeks a set $F \subseteq (V \times V)$ (resp. a subset $\mathcal{F} \subseteq \mathcal{R}$) of size at most $k$ whose *modification* in the input structure satisfies some given property $\Pi$. Regarding graph modification problems, we introduce the notion of *branches*, that allows us to reduce the size of a given instance to obtain a so-called kernel. A similar idea has been used by Bodlaender et al. [15], under the name of *protrusions*, but require additional properties for the problem at hand. In particular, we obtain the first polynomial kernels for Closest 3-Leaf Power [9] and Proper Interval Completion [10], two problems finding applications in computational biology. In the second part of this paper, we introduce the notion of *safe partition* for constraint modification problems. Combined with a constant-factor approximation algorithm, this notion allows us to obtain a linear vertex-kernel for Feedback Arc Set in Tournaments [6], improving the previous bound of $O(k^2)$ vertices for the problem [5]. Next, we develop and push further a technique used in a few parameterized problems [21, 32, 67] that we call *Conflict Packing*. Combined with the notion of safe partition or *sunflower*, this method provides linear vertex-kernels for the Dense Rooted Triplet Inconsistency and Dense Betweenness problems. We would like to notice that a similar technique has been introduced by van Bevern et al. [65], namely *kernelization through tidying*, in order to deal with *vertex deletion* problems. To conclude, we mention some work that has been done *beyond* kernelization. In particular, in a joint work with Jean Daligault, Christophe Paul and Stéphan Thomassé, we proved that the Multicut problem can be reduced in FPT time to instances of treewidth bounded by a function of $k$. Part of this result, based on graph minor theory [60, 61] and the notion of *irrelevancy* [54], has been used by Bousquet et al. to obtain a parameterized algorithm for Multicut [18].

*The results presented in this extented abstract are based on joint works with Stéphane Bessy, Jean Daligault, Fedor V. Fomin, Serge Gaspers, Sylvain Guillemot, Frédéric Havet, Christophe Paul, Saket Saurabh and Stéphan Thomassé.

# Contents



# List of Figures

# 1 Introduction

**Modification problems** In this thesis, we consider *graph* (and *constraint*) *modification problems* where only the edge set (resp. collection of constraints) can be modified. Such problems constitute a broad range of $NP$-complete problems [36]. Formally, these problems can be defined as follows:

| $\mathcal{G}$-Graph Modification: |
|---|
| **Input**: A graph $G := (V, E)$, and an integer $k$. |
| **Parameter**: $k$. |
| **Question**: Does there exist a subset $F \subseteq (V \times V)$ of size at most $k$ such that the graph $H := (V, E \triangle F)$ belongs to $\mathcal{G}$? |

| $\Pi$-Constraint Modification: |
|---|
| **Input**: A set of $p$-sized *constraints* $\mathcal{R}$, $p \geqslant 2$, defined over some universe $V$, and an integer $k$. |
| **Parameter**: $k$. |
| **Question**: Does there exist a subset $\mathcal{F} \subseteq \mathcal{R}$ of size at most $k$ whose modification in $\mathcal{R}$ satisfies the property $\Pi$? |

Well-known examples of *graph modification problems* include Vertex Cover [2] , Feedback Vertex Set [64] and Cluster Editing [25], while Dense Rooted Triplet Inconsistency [37] and Dense Betweenness [48, 49] are examples of *constraint modification problems*. These problems find numerous applications in several domains, such as image processing [62], relational databases [63] and computational biology [39, 46, 52]. There exist several approaches to deal with such NP-hard problems: using heuristics, approximation algorithms, probabilistic algorithms and so on. However, these techniques measure the complexity of the given problem with respect to its size $n$ only. In particular, they do not use the information given by a *parameter* $k$ associated to the problem, such as the *size of the modification sought* for modification problems. A question that arises is thus: can we solve *exactly* a *parameterized problem* by confining the combinatorial explosion (which should not be avoided if $P \neq NP$) to the parameter $k$ *only*? Formalized by Downey and Fellows in the late 1990s, the notion of *parameterized complexity* gives a theoretical framework to obtain such results [31, 33, 55].

**Parameterized complexity** A problem *parameterized* by some integer $k$ is said to be *fixed-parameter tractable* (FPT for short) if it can be solved in time $f(k) \cdot n^{O(1)}$, where $n$ stands for the size of the instance at hand. While considering *modification problems*, a natural parameter is thus the number of allowed modifications. There are a lot of modification problems that admit parameterized algorithms when parameterized by the size of the solution [18, 22, 66]. For instance, the problems mentioned as examples in the beginning of this extended abstract are all known to admit parameterized algorithms [5, 28, 45, 46, 55]. Regarding graph modification problems, a famous result of Cai [22] states that whenever the property $\Pi$ is *hereditary* (*i.e.* closed under taking induced subgraphs) and characterized by a *finite family of forbidden induced subgraphs*, then the $\Pi$-Graph Modification problem is FPT.

**Kernelization** We follow this line of research and consider more precisely *kernelization algorithms*, one of the most powerful techniques to design parameterized algorithms [12]. A *kerneliza-*

*tion algorithm* for a parameterized problem is a *polynomial-time algorithm* (in $|I| + k$) that, given an instance $(I, k)$ of a parameterized problem, outputs an *equivalent* instance of the same problem $(I', k')$ (the *kernel*) such that $|I'| \leqslant f(k)$ and $k' \leqslant k$ for some computable function $f$. Such a function is called the *size* of the kernel. A classical result states that a problem is fixed-parameter tractable if and only if it admits a kernel [55]. However, the kernel obtained through this theoretical result is of super-polynomial size, and the aim is to improve this given size. Indeed, finding a kernel of *small* size (for instance polynomial -or even linear- in $k$) is highly desirable in practice [56]. Nevertheless, recent results of Bodlaender et al. [14] and Fortnow and Santhanam [34] give evidence that not *all* parameterized problems admit polynomial kernels, unless the polynomial hierarchy collapses to its third level (which in turn would imply $NP \subseteq coNP/poly$ [34]). This result relies on the notion of *or-composition* for parameterized problems, and has been recently generalized by the introduction of *cross-composition* [16].

**Known results** As mentioned earlier, a large number of parameterized algorithms are known for modification problems. This observation is also verified when considering polynomial kernels for such problems. For instance, the Vertex Cover and Cluster Editing problems admit a *linear vertex-kernel* (with at most $2k$ vertices) [2, 25], while the Feedback Vertex Set [64] and Dense Rooted Triplet Inconsistency [37] both admit a kernel with $O(k^2)$ vertices. Recently, Krastch and Walshtröm [51] provided evidence that there exist some $\Pi$-Graph Modification problems that *do not* admit polynomial kernels, even if $\Pi$ is a very restricted class of graphs. Actually, their result disproved a conjecture of Cai [13], stating that the $\Pi$-Graph Modification problem admits a polynomial kernel as long as $\Pi$ is hereditary and characterized by a finite family of forbidden induced subgraphs.

**Our results** We provide results in both directions, obtaining polynomial kernels for several problems, sometimes establishing the first known polynomial kernels for the considered problems. In particular, using the notion of *branches*, we obtain polynomial kernels for Closest 3-Leaf Power [9] and Proper Interval Completion [10], two well-studied parameterized problems [29, 46]. Moreover, by adapting the results of Kratsch and Walshtröm we obtain lower bounds for two graph modification problems [40], namely $C_l$-free edge deletion and $P_l$-free edge deletion for $l \geqslant 7$. Regarding constraint modification problems, introducing the notion of *safe partition*, we obtain a linear vertex-kernel for Feedback Arc Set in Tournaments [6], improving the best known bound of $O(k^2)$ vertices [5]. Finally, by developing and pushing further a technique used in a few parameterized problems [21, 32, 67], that we call *Conflict Packing*, we obtain linear vertex-kernels for the Dense Rooted Triplet Inconsistency and Dense Betweenness problems [57].

## 2 Preliminaries

**Graphs** We consider simple, loopless, undirected, graphs $G := (V, E)$, where $V(G)$ stands for its *vertex set* and $E(G)$ for its *edge set*. Given a vertex $u \in V(G)$, we denote by $N_G(u)$[1] its *open neighborhood* and by $N_G[u] := N_G(u) \cup \{u\}$ its *closed neighborhood*. Two vertices $u, v \in V$ are *true twins* iff $N[u] = N[v]$. The operation of adding a true twin to $G$ (called *true twin addition*) consists of adding a new vertex $v$ to $G$ as a true twin of some vertex $u \in V(G)$. Given a subset $S \subseteq V$, $G[S]$

[1]We forget to mention the graph $G$ whenever the context is clear.

denotes the subgraph *induced* by $S$, *i.e.* the graph $(S, E(S))$ where $E(S) := \{uv \in E \ : \ u, v \in S\}$. In a slight abuse of notation, we use $G \setminus S$ to denote the graph $G[V \setminus S]$. Moreover, $\delta(S) := \{s \in S \ : \ N_{G \setminus S}(s) \neq \emptyset\}$ denotes the *border* of $S$. We say that a subset $M \subseteq V$ is a *module* of $G$ if for every vertex $v \in V \setminus M$ we have $M \subseteq N(v)$ or $M \cap N(v) = \emptyset$. A *critical clique* of $G$ is a *maximal clique module* [52]. Given an instance $G := (V, E)$ of the $\mathcal{G}$-Graph Modication problem, a set $F$ is an *edition* of $G$ if the graph $H := (V, E \triangle F)$ belongs to $\mathcal{G}$. We say that $F$ is a $k$-edition when $|F| \leqslant k$, and that it is optimal if $|F|$ is of minimum size. We say that $\mathcal{G}$ is *hereditary* whenever any induced subgraph of a graph of $\mathcal{G}$ belongs to $\mathcal{G}$. Finally, we will also mention *tournaments* $T := (V, A)$, which are obtained by taking an arbitrary orientation of the complete unoriented graph (called *clique*).

**Constraints** We also consider *dense* collections $\mathcal{R}$ of $p$-sized *constraints*, $p \geqslant 2$, defined over some universe $V$. Here, the term *dense* means that there is *exactly* one constraint for every subset of size $p$ of $V$. Given an instance $R := (V, \mathcal{R})$ of the $\Pi$-Constraint Modification problem, an edition $\mathcal{F} \subseteq \mathcal{R}$ is a set of constraints whose modification in $\mathcal{R}$ allows to obtain a collection of constraints that satisfy $\Pi$. As before, we say that $\mathcal{F}$ is a $k$-edition when $|\mathcal{F}| \leqslant k$, and that it is optimal when $|\mathcal{F}|$ is of minimum size.

# 3 Part I. Graph modification problems

In the first part of this thesis, we look at graph modification problems. We first consider two problems with direct applications in the domain of computational biology, namely Closest 3-Leaf Power [29] and Proper Interval Completion [46, 47]. A *p-leaf power* is a graph $G := (V, E)$ that can be obtained from a tree $T$ whose leaves are in one-to-one correspondence with $V$ and such that $uv \in E(G)$ if and only if the leaves corresponding to $u$ and $v$ are at distance at most $p$ in $T$. A graph $G := (V, E)$ is a *proper interval graph* if it is the intersection graph of a finite family of intervals of the real line such that no interval strictly contains another interval.

Closest 3-Leaf Power :
**Input** : A graph $G := (V, E)$, an integer $k$.
**Parameter** : $k$.
**Question** : Does there exist a subset $F \subseteq (V \times V)$ of size at most $k$ such that the graph $H := (V, E \triangle F)$ is a 3-leaf power ?

Proper Interval Completion :
**Input** : A graph $G := (V, E)$, an integer $k$.
**Parameter** : $k$.
**Question** : Does there exist $F \subseteq (V \times V) \setminus E$ of size at most $k$ such that the graph $H := (V, E \cup F)$ is a proper interval graph ?

**Known results** While the recognition of $p$-leaf powers is polynomial-time solvable for $p \leqslant 5$ [19, 20, 23] and open for larger values of $p$, the Closest $p$-Leaf Power problem is already $NP$-Hard for $p = 2$ [62] (since it is equivalent to the Cluster Editing problem). However, this problem is known to be fixed-parameter tractable for $p = 3, 4$ [29, 30], its parameterized complexity being

opened for larger values of $p$. Regarding the PROPER INTERVAL COMPLETION problem, it is also known to be $NP$-Complete [38] and fixed-parameter tractable [22, 46, 47]. We would like to mention that the first result proving the parameterized complexity of this problem appeared in FOCS'94 [46], and thus that the existence of a polynomial kernel was opened until then.

**Our contribution** In both cases, we establish polynomial kernels when parameterized by the size of the solution sought. These results constitute the first known polynomial kernels for both problems. The former, answering a question first asked by Dom et al. [28], is a joint work with Stéphane BESSY and Christophe PAUL [8, 9], while the latter, which settles a problem opened for more than a decade, is a joint work with Stéphane BESSY [10]. To obtain polynomial kernels for these two problems, we introduce a technique based on *branches* of graphs, which are roughly speaking parts of the graph that already fulfill the desired property and are connected *properly* to the rest of the graph.

## 3.1 Reduction rules using branches

### 3.1.1 Definition and main idea

**Intuition** The general idea behind the use of *branches* in kernelization algorithms is to *reduce* a set of vertices that induce a graph that already belongs to the target class $\mathcal{G}$. We would like to observe that this idea is somehow natural when considering kernelization algorithms. Indeed, it has already been used in several kernelization algorithms [9, 40], for instance with reductions rules of the form:

> *Let $G := (V, E)$ be an instance of $\mathcal{G}$-*GRAPH MODIFICATION *problem. Remove from $G$ any connected component $C$ such that $G[C]$ belongs to $\mathcal{G}$.*

As we shall see, such a reduction rule is a particular case of *branches*. Another application of this idea lead to a linear vertex kernel for the CLUSTER EDITING [41] problem: given a *critical clique* of $G$ of arbitrary size, it is possible to preserve only $k + 1$ vertices. This observation comes from the fact that all the relevant information contained in such a clique lies in its *border*, not within the clique itself. Hence, it is safe to preserve only a few number of vertices. Based on this idea, we introduce and develop the concept of *branches* for the $\mathcal{G}$-GRAPH MODIFICATION problem, which is a non-trivial generalization of the above-mentionned rules. This method is particularly well-suited for the $\mathcal{G}$-GRAPH MODIFICATION problem where the members of $\mathcal{G}$ admit a so-called *adjacency decomposition*.

**Adjacency Decomposition** *Let $\mathcal{G}$ be a class of graphs. We say that $\mathcal{G}$ admits an* adjacency decomposition *if for any graph $G := (V, E)$ that belongs to $\mathcal{G}$, there exists a collection $\mathcal{V} := \{V_1, \ldots, V_l\}$ covering $V$ such that $V_i \subseteq V$, $i \in [l]$ and such that two sets $V_i, V_j$, $i \neq j$ are connected according to some adjacency relation $P_{\mathcal{G}}$.*

As an example, we can observe that *chordal graphs* admit an adjacency decomposition, known as *clique graph* [35]. Given a chordal graph $G := (V, E)$ (*i.e.* a graph without induced cycles of length greater than 3), we define the collection $\mathcal{V}$ as the collection of maximal cliques of $G$. Now, the clique graph $C(G) := (\mathcal{V}, E_c)$ is defined as the graph whose vertices are the maximal cliques of $G$ and the adjacency relation $P_{chordal}$ is: there is an edge between two vertices of $C(G)$ if their

intersection is a minimal separator of $G$ [35]. As we shall see later, the class of 3-leaf powers and of proper interval graphs admit such an adjacency decomposition. We now give a general definition of the notion of *branch*, which shall be adjusted to every considered problem.

**Branches** *Let $\mathcal{G}$ be a class of graphs admitting an adjacency decomposition $(\mathcal{V}, P_{\mathcal{G}})$, and $G := (V, E)$ be a graph of $\mathcal{G}$. We say that a subset $B \subseteq V$ is a* branch *of $G$ if:*

- *the graph $G[B]$ belongs to the class $\mathcal{G}$,*
- *the edges $E(\delta(B), N_{G\setminus B}(\delta(B)))$ respects the adjacency property $P_{\mathcal{G}}$.*

**Detecting the branches** To obtain a kernelization algorithm, a necessary condition is that the branches can be detected in polynomial time. Once again, this really depends on the problem at hand. For Closest 3-Leaf Power and Proper Interval Completion, this can be done in polynomial time. We do not describe the algorithms here and refer to [9] and [10] for a complete description.

**Reducing the branches** Let $G := (V, E)$ be an instance of the $\mathcal{G}$-Graph Modification problem where $\mathcal{G}$ admits an adjacency decomposition. Assume we are given a branch $B \subseteq V$ of $G$. Then, the only relevant information contained in $B$ is located in its *border*. In other words, the set $B^R := B \setminus \delta(B)$ contains *too many information*, and one can prove that it is safe to replace it by an *equivalent* graph of bounded size, containing the same amout of information. However, this particular part is problem-specific, since the proofs heavily use the adjacency decomposition of the target class $\mathcal{G}$. Roughly speaking, a branch-reduction rule must follow these lines:

> *Let $G := (V, E)$ be an instance of the $\mathcal{G}$-Graph Modification problem where $\mathcal{G}$ admits an adjacency decomposition. Let $B \subseteq V$ be a branch of $G$. Replace $B^R$ by an* equivalent graph *of size bounded in $k$.*

**Bounding the size of the kernel** We now present why such a reduction of the branches of a graph leads to polynomial kernels. Let $G := (V, E)$ be an instance of the $\mathcal{G}$-Graph Modification problem where $\mathcal{G}$ admits an adjacency decomposition $(\mathcal{V}, P_{\mathcal{G}})$. For the sake of simplicity, we assume that $\mathcal{V}$ is a partition of $V$. Let $F$ be a $k$-edition of $G$ and $H := G \triangle F$. We define a vertex of $G$ as *affected* if it is contained in some pair of $F$, and a set $V_i$ of $\mathcal{V}$ as *affected* if it contains an affected vertex. Now, we consider connected components of the graph $H \setminus A(\mathcal{V})$, where $A(\mathcal{V})$ denotes the affected sets of $\mathcal{V}$. By definition, these components define *branches* of $G$, whose border must be attached to the rest of the graph according to $P_{\mathcal{G}}$ (since they are non-affected vertices). Since $|F| \leqslant k$ by definition, there are at most $2k$ affected vertices in $H$, and hence at most $2k$ affected sets in $\mathcal{V}$, which implies that there are at most $O(k)$ branches. Hence, finding a way to reduce *efficiently* such branches would yield polynomial kernels. This particular part is problem-specific, and we will consider the Closest 3-Leaf Power problem as an example.

### 3.1.2 Branches for Closest 3-Leaf Power

To begin with, we give a characterization of 3-leaf powers in terms of critical cliques. Observe that the set of critical cliques of $G := (V, E)$ partition its vertex set $V$. We define the *critical clique graph* $\mathcal{C}_G$ of $G$ as the graph obtained by *contracting* every critical clique of $G$ (without keeping multi-edges).

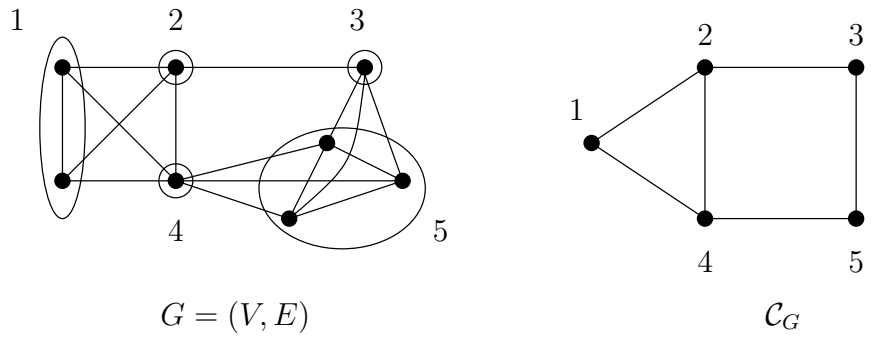


Figure 1: A graph $G := (V, E)$ and its critical clique graph $\mathcal{C}_G$.

**Theorem 3.1 ([29])** *The following properties are equivalent:*

*(i) a graph $G := (V, E)$ is a* 3-leaf power*;*

*(ii) it does not contain any* {dart,bull,gem,hole} *as an induced subgraph (see Figure 2);*

*(iii) its critical clique graph $\mathcal{C}_G$ is a forest.*

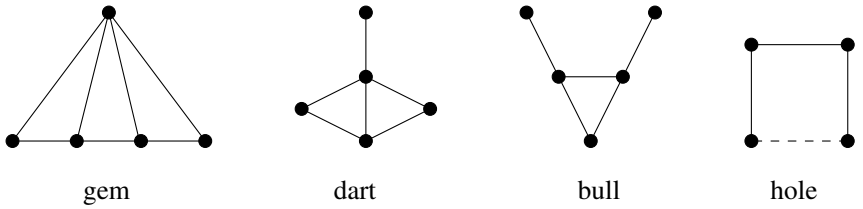


Figure 2: The forbidden induced subgraphs for the 3-leaf powers.

A first reduction rule that can be defined for the problem reduces the size of the critical cliques. Its soundness follows from the following result.

**Lemma 3.2 ([9])** *Let $G := (V, E)$ be an instance of $\mathcal{G}$-*Graph Modification *for a class $\mathcal{G}$ hereditary and closed under true twin addition. There always exists an optimal edition $F$ of $G$ that preserves the critical cliques of $G$.*

In other words, any critical clique of $G$ is a clique module (not necessarily critical) of the graph $H := (V, E \triangle F)$.

**Rule 1 (True Twins)** *Let $G := (V, E)$ be an instance of* Closest 3-Leaf Power *and $T$ be a critical clique of $G$ such that $|T| > k + 1$. Remove $|T| - (k + 1)$ arbitrary vertices from $T$.*

**Lemma 3.3 ([9])** *Rule 1 is sound and can be applied in polynomial time.*

We now give a formal definition of a *branch* for the problem Closest 3-Leaf Power. By Theorem 3.1, the *adjacency decomposition* of a 3-leaf power corresponds to its *critical clique graph*. Hence, a *branch* is a set of vertices $B \subseteq V$ whose induced critical clique graph is a forest and that is connected to the rest of the graph according to this notion of critical clique. Formally, we obtain the following definitions.

**Definition 3.4 (Branches)** *Let $G = (V, E)$ be an instance of* Closest 3-Leaf Power *and $B \subseteq V$ a subset of vertices of $V$. The set $B$ is a* branch *of $G$ if it is the union of critical cliques $\{C_1, \ldots, C_r\}$ such that the subgraph of $\mathcal{C}_G$ induced by $\{C_1, \ldots, C_r\}$ is a tree.*

**Definition 3.5 ($l$-Branches)** *Let $B := \{C_1, \ldots, C_r\}$ be a branch of an instance $G := (V, E)$ of* Closest 3-Leaf Power*. We say that a critical clique is an* attachment point *of $B$ if $N_{G \setminus B}(C_i) \neq \emptyset$. A branch containing $l$ attachment points is called a $l$-*branch*.*

**Reducing the branches** Observe in a first place that, by definition, a critical clique is a particular 1-branch, containing exactly one attachment point. Hence Rule 1 allows us to reduce special cases of 1-branches. For the sake of our other reduction rules, we only need to consider 1 and 2-branches. The intuition for the 1-branches is as follows: suppose we are given a branch containing one attachment point. Due to Lemma 3.2, we know that we can assume *without loss of generality* that any optimal solution *preserves* the critical cliques. In other words, we can assume that any critical clique of $G$ is a *clique module* of the graph $H := (V, E \triangle F)$ for any optimal edition $F$.

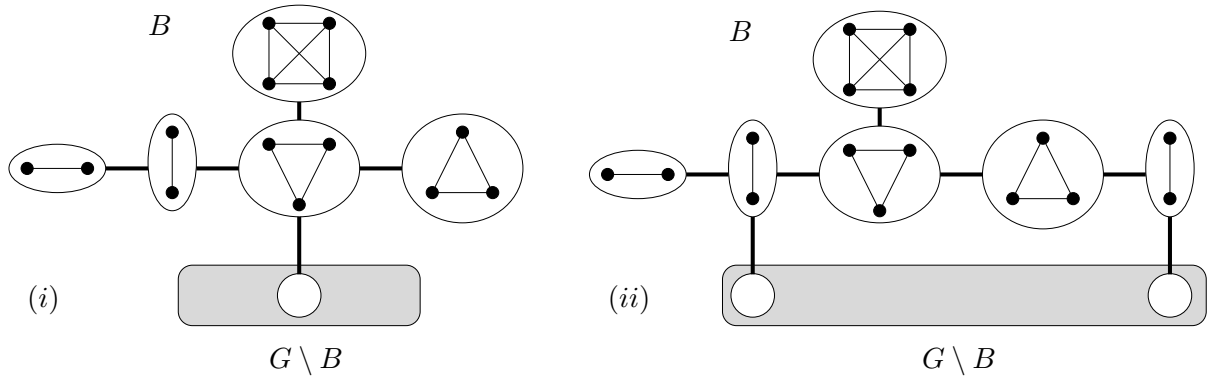


Figure 3: Illustration of the notion of 1- and 2-branches for the Closest 3-Leaf Power problem.

Hence, since the attachment point is a critical clique by definition, and since the *pendant graph* attached to this critical clique is a *branch*, we only need to preserve the attachment point and to add a special critical clique encoding its neighbourhood. We hence have the following reduction rule, where $A_1$ stands for the attachment point of any 1-branch $B \subseteq V$ and $B^R := B \setminus A_1$.

**Rule 2 (1-branche)** *Let $G := (V, E)$ be an instance of* Closest 3-Leaf Power *and $B \subseteq V$ a 1-branch of $G$. Remove from $G$ the vertices of $B^R$ and add a new critical clique neighbouring $A_1$, of size $\min\{|N_B(A_1)|, k+1\}$.*

**Lemma 3.6 ([9])** *Rule 2 is sound and can be applied in polynomial time.*

Let $G' := (V', E')$ be the graph obtained after an application of Rule 2. Due to the particular adjacency decomposition of 3-leaf powers and since the attachment point is a clique module of the graph $H' := (V', E' \triangle F')$ for any $k$-edition $F'$ of $G'$, the remaining vertices of the branch can be *plugged back* into $H'$ while preserving a 3-leaf power, and thus $F'$ is a $k$-edition for $G$.

A similar reduction rule can be designed for 2-branches, that is branches containing *exactly* two attachment points. In that case, we can reduce the length of the path composed of critical cliques

joining the two attachment points [9]. We do not define formally this particular reduction rule, but a picture of its application is given in Figure 4.

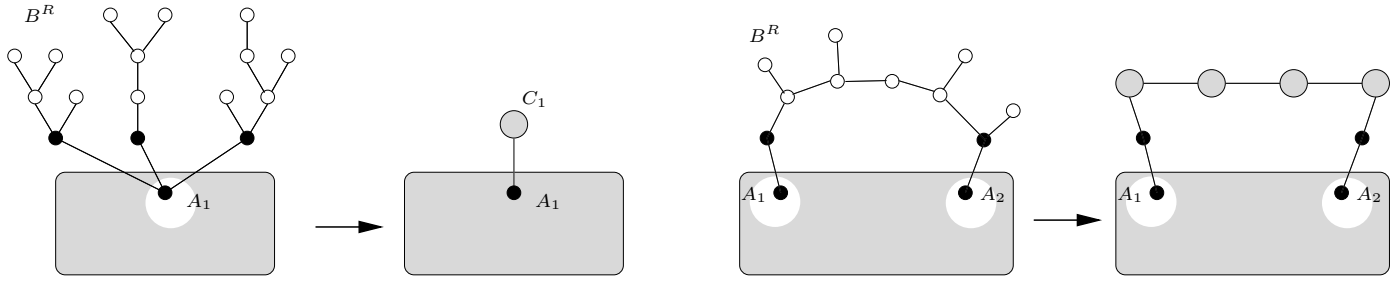


Figure 4: The branches reduction rules for Closest 3-Leaf Power.

**Bounding the size of the kernel** To conclude, we give a sketch of the proof used to bound the size of the kernel. Assume we are given a positive reduced instance $G := (V, E)$ of the Closest 3-Leaf Power problem. Let $F$ be an edition of $G$ of size at most $k$. We denote by $H := (V, E \triangle F)$ the 3-leaf power resulting from the edition of $F$ in $G$. By Theorem 3.1, we know that the critical clique graph of $\mathcal{C}_H$ is a forest. For the sake of simplicity, we will assume that it is a tree. Now, we define a critical clique of $H$ as *affected* if it contains an affected vertex. Observe that there are at most $2k$ affected critical cliques. We consider the minimal tree $T_H$ spanning these vertices in $\mathcal{C}_H$. Now, if we remove the affected critical cliques and the vertices of degree at least 3 in $T_H$ (there are at most $2k$ such vertices), the 2-branch reduction rule allows us to bound the size of the remaining connected components. Indeed, if the size of such a component is greater than some given constant (independent of $k$), then the graph $G$ could be reduced since such a set of vertices would correspond to a 2-branch of $G$. Next, by definition, we know that every set of vertices of $H \setminus V(T_H)$ pendant to a *non-affected* critical clique of $T_H$ defines a 1-branch. Since the graph is reduced, we can bound the size of such a set. Finally, using a slightly modified 1-branch reduction rule, we can also bound the size of any set of vertices of $H \setminus V(T_H)$ attached to an affected critical clique of $H$. Altogether, we can bound the number of critical cliques of $H$, which in turn allows us to bound the number of critical cliques of $G$ using some technical lemma [59]. Rule 1 finally implies the bound on the total number of vertices of $G$.

**Theorem 3.7 ([9])** *The* Closest 3-Leaf Power, 3-Leaf Power Completion *and* 3-Leaf Power Deletion *problems admit kernels with $O(k^3)$ vertices.*

### 3.1.3 Branches for Proper Interval Completion

In this section, we present a polynomial kernel for the Proper Interval Completion problem. As we shall see, the notion of branches can be used on this problem as well, but requires a more technical analysis. Recall that a graph $G := (V, E)$ is a *proper interval graph* if it is the intersection graph of a finite family of intervals of the real line such that no interval strictly contains another interval. We now give several characterisations of proper interval graphs that will be useful to define the notion of branches for the Proper Interval Completion problem. We say that an ordering of the vertices $V := \{v_1, \ldots, v_n\}$ respects the *umbrella property* if $v_iv_j \in E$, $i < j$ implies $v_iv_l, v_lv_j \in E$ for every $i < l < j$.

**Theorem 3.8 ([53, 68])** *The following properties are equivalent:*

*(i) a graph* $G := (V, E)$ *is a* proper interval graph*;*

*(ii) it does not contain any* {net,3-sun,claw,hole} *as an induced subgraph (see Figure 5);*

*(iii) its vertex set admits an ordering respecting the* umbrella property*.*

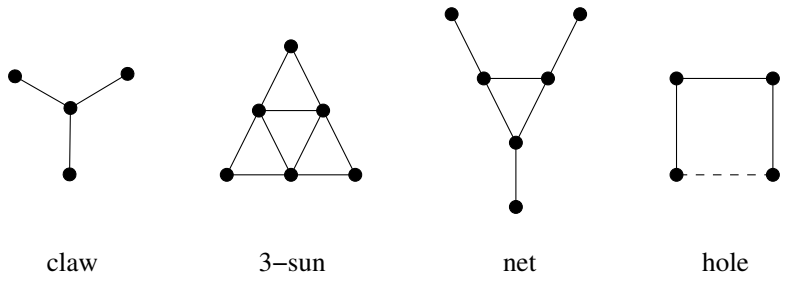


Figure 5: The forbidden induced subgraphs for the proper interval graphs.

**Adjacency Decomposition** This characterization allows us to obtain the adjacency decomposition for proper interval graphs. Indeed, let $G := (V, E)$ be a proper interval graph and assume $\sigma := v_1 \dots v_n$ is an ordering of $V$ respecting the umbrella property. By definition, the set $V_1 := \{v_1, \dots, v_l\}$ where $v_l$ is the neighbour of $v_1$ with maximal index in $\sigma$ is a clique of $G$. Similarly, the set $V_2 := \{v_{l+1}, \dots, v_{l'}\}$, where $v_{l'}$ is the neighbour of $v_{l+1}$ with maximal index in $\sigma$, is a clique of $G$, and so on. Now, by definition of the umbrella property, the set of vertices $V_1$ and $V_2$ are connected by a *join* in $G$. In other words, the graph $G[V_1 \cup V_2]$ is a *threshold graph*: for every $v_i, v_j \in V_1$, $i < j$, $N_{V_2}(v_i) \subseteq N_{V_2}(v_j)$ holds. Hence, a proper interval graph can be decomposed in a path of cliques, where every consecutive cliques are connected by a join (see Figure 6).

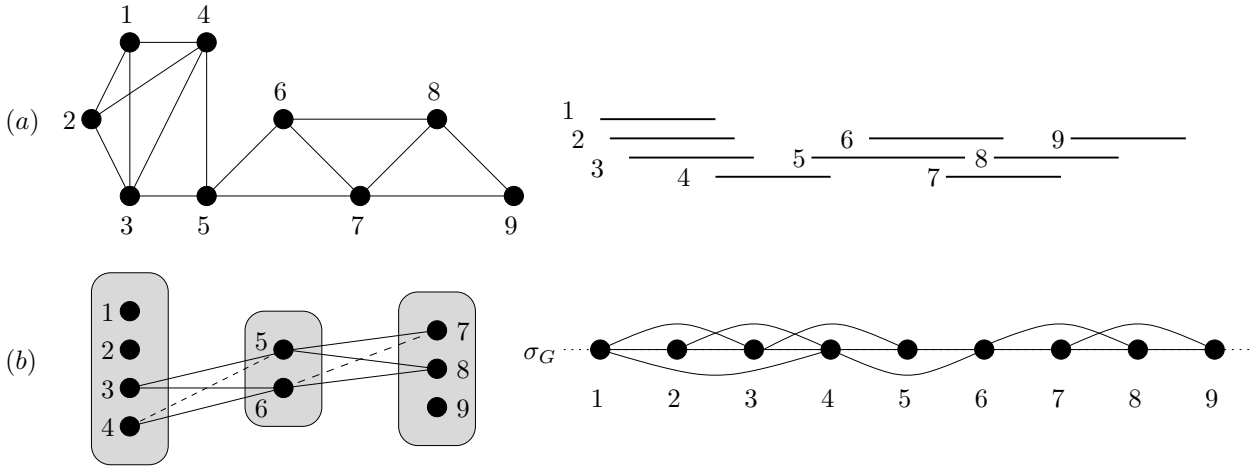


Figure 6:

The notion of branches can thus be adapted to this particular decomposition. Now, using more technical arguments, we can develop reduction rules for the so-called 1- and 2-branches, yielding to the following result.

**Theorem 3.9 ([10])** *The* Proper Interval Completion *problem admits a kernel with $O(k^5)$ vertices.*

### 3.1.4 Link with protrusions

We now briefly describe the *protrusion* technique [15], which works on a treewidth decomposition of a given graph. Using protrusions, Bodlaender et al. [15] are able to derive several polynomial kernels for edition problems, establishing a unifying framework for problems defined on graphs of bounded genus. While the concept of protrusions is similar to the notion of branches, some additional properties on the problem at hand are required in order to make use of such protrusions. As mentioned, the input graph must be embeddable in a surface of bounded genus. Moreover, the problems considered by this method have to be expressible in Counting Monadic Second Order logic (CMSO) and *compact* [15].

**Protrusions** Roughly speaking, a *protrusion* is a subgraph of constant treewidth separated from the rest of the graph by a constant number of vertices. The idea to reduce such parts of the graph is as follows [15]: *If there is a constant size separator such that after its removal we obtain a connected component of large size and of constant treewidth, then we replace this component with a graph of small size.* If the idea of this reduction rule is similar to the reduction rule dealing with branches, observe however that the notion of branches exploit the particular decomposition of the target class $\mathcal{G}$, which is not the case here. The results presented in [15] apply to $p$-min-CMSO (resp. $p$-eq-CMSO, $p$-max-CMSO), where one is given a graph $G := (V, E)$ and an integer $k$ and seeks a set of vertices/edges $S$ of size at most (resp. exactly, at least) $k$ whose modification satisfies a given CMSO-property $\Pi$. For the sake of simplicity, we only mention the results concerning $p$-min-CMSO problems. The *annotated* version of such a problem is additionnally given a subset of vertices $Y \subseteq V$ and the solution set $S$ must be a subset of $Y$. In what follows we consider annotated versions of the problem $p$-min-CMSO, since the results obtained for annotated problems can be carried out to classical problems (Corollary 1 in [15]).

**Definition 3.10 ($r$-protrusion)** *Given a graph $G := (V, E)$, a set $X \subseteq V$ is an* $r$-protrusion *of $G$ if $treewidth(G[X]) \leqslant r$ and $|\delta(X)| \leqslant r$.*

**Reducing the protrusions** Let $X \subseteq V$ be an $r$-protrusion. The aim of a reduction rule dealing with protrusions is to *replace* the set of vertices lying in $X \setminus \delta(X)$ by an equivalent graph of bounded-size. Considering the annotated version is helpful to do so, since one can then assume that the set of vertices $X \cap Y$ is a subset of $\delta(X)$, and hence $S \cap X \subseteq \delta(X)$ for any solution set $S$. Since the border of an $r$-protrusion is preserved by the reduction rule, this allows to prove the soundness of such a rule. Roughly speaking, the reduction goes as follows: for every instance $G := (V, E)$ of an annotated $p$-min-CMSO problem that can be embedded in a surface of bounded genus, there exists an algorithm that given an $r$-protrusion $X$ such that $Y \cap X \subseteq \delta(X)$ outputs in time $O(|X|)$ an equivalent instance $G' := (V', E')$ such that $|V'| < |V|$. The remaining part of the kernelization algorithm is thus to prove how to compute a protrusion satisfying the previous conditions. This can be done as long as the given protrusion contains $O(k^c)$ vertices for some constant $c > 0$. Finally, they prove how a protrusion of sufficiently large size can be computed in polynomial time (Subsection 5.1 in [15]), and using some technical and combinatorial results they obtain the following.

**Theorem 3.11 ([15])** *Let $\Pi$ be a $p$-MIN-CMSO problem defined on graphs of bounded genus such that $\Pi$ or $\overline{\Pi}$ is compact. Then the annotated version $\Pi_A$ admits a quadratic kernel*[2]*.*

**Corollary 3.12 ([15])** [3] *For $g \geqslant 0$, the* Independent Dominating Set*,* Minimum Leaf Out-Branching*,* Edge- $\mathcal{S}$-covering *and* Odd Set *problems admit polynomial kernels on graphs of genus at most $g$.*

### 3.2 Modular decomposition-based reduction rules

The last graph modification problem we consider is Cograph Edition [40], the *cographs* being the graphs that do not admit any path on four vertices as induced subgraph. To obtain a polynomial kernel for this problem, we use *modular decomposition* [42]. Since cographs are *totally decomposable* for the modular decomposition, the use of this tool to design reduction rules seems to be particularly well-suited. However, the reduction rules used can also be related to the notion of branches. In particular, if there exists a module $M$ such that $G[M]$ is a cograph, then we can safely replace $M$ by an independent set of size $\min\{|M|, k+1\}$. When $G[M]$ is not a cograph, then we do the same replacement and preserve a *copy* of $M$ to keep track of the editions needed *inside* the module. Altogether, we obtain the following results.

**Theorem 3.13 ([40])** *The* Cograph Edition*,* Cograph Edge-Deletion *and* Cograph Completion *problems admit kernels with $O(k^3)$ vertices.*

### 3.3 Kernelization lower bounds

To conclude with graph modification problems, we mention some lower bounds on kernelization obtained with Sylvain Guillemot, Frédéric Havet and Christophe Paul. These resuls improve upon reductions first published in [40]. Given a graph $\Gamma := (V, E)$ and an integer $k$, the $\Gamma$-Edge Deletion problem asks whether there exists a set $F \subseteq E$ of size at most $k$ such that the graph $H := (V, E \setminus F)$ does not contain $\Gamma$ as an induced subgraph. The graphs $C_l$ (resp. $P_l$) are cycles (resp. paths) containing $l$ vertices.

**Theorem 3.14** *The $P_l$-free edge deletion and $C_l$-free edge deletion problems do not admit a polynomial kernel for any $l \geqslant 7$, unless $NP \subseteq coNP/poly$.*

To obtain these results, we make use of several recently-introduced tools, named *or-composition* [11] and *polynomial time and parameter transformation* [17]. These particular tools, introduced by Bodlaender et al. [11, 17] and Fortnow and Saurahm [34], give a theoretical framework to prove that some parameterized problems *do not* admit polynomial kernels unless $NP \subseteq coNP/poly$. The first technique, namely *or-composition* in [11], works as follows : assume we are given $t$ instances $(I_1, k), \ldots, (I_t, k)$ of a parameterized problem $L$. An *or-composition* for $L$ is a polynomial-time algorithm (in $\sum_{i=1}^{t} |I_i| + k$) that outputs an instance $(I_c, k_c)$ of $L$ such that $(i)$ $(I_c, k_c)$ is positive iff there exists $1 \leqslant i \leqslant t$ such that $(I_i, k)$ is positive and $(ii)$ $k_c \in O(k^c)$ for some constant $c > 0$. Such a parameterized problem is said to be *or-composable*.

[2] The obtained kernels are quadratic for the $p$-MAX-CMSO and cubic for $p$-EQ-CMSO.

[3] Using the notion of *finite integer index*, Bodlaender et al.[15] improve this result to *linear* vertex-kernels for problems on graphs of genus at most $g$

**Theorem 3.15 ([11])** *Let $L$ be an or-composable parameterized problem. Then $L$ does not admit a polynomial kernel unless $NP \subseteq coNP/poly$.*

Using this result, it is possible to show that several standard problems, such as $k$-Path, do not admit polynomial kernels. Based on the notion of $NP$-complete reductions, Bodlaender et al. [17] developped the concept of *polynomial time and parameter transformation*, allowing to carry out lower bounds results from one problem to another. Formally, a polynomial time and parameter reduction from a parameterized problem $L$ to a parameterized problem $L'$ takes as input an instance $(I, k)$ of $L$ and outputs an instance $(I', k')$ of $L'$ which is positive iff $(I, k)$ is positive and such that $k' \in O(k^c)$ for some constant $c > 0$.

**Theorem 3.16 ([17])** *Let $L$ and $L'$ be two parameterized problems such that there exists a polynomial-time and parameter transformation from $L$ to $L'$. If $L'$ admits a polynomial kernel, then $L$ admits a polynomial kernel.*

In particular, this result implies that if $L$ is or-composable and polynomial-time and parameter reducible to $L'$, then $L'$ does not admit a polynomial kernel. Thus, in order to design lower bounds for parameterized problems, it is convenient to define some *toy-problem*, that can be shown to be or-composable and which reduces to the main considered problem. To conclude on lower bounds, we would like to mention that there exists a recently-introduced technique, namely *cross-composition* [16], that unifies the previous methods. This technique is in particular well-suited for proving lower bounds for problems parameterized in a non-standard way [16].

# 4 Part II. Constraint modification problems

In the second part of the thesis, we look at *constraint modification problems*, where the underlying structure is no longer a graph but a set of $p$-sized constraints $\mathcal{R}$, $p \geqslant 2$, defined over some universe $V$. The aim of these problems is to find a set of at most $k$ constraints whose modification imply that $\mathcal{R}$ satisfies the property $\Pi$. We are in particular interested in the cases where the *consistency* (*i.e.* satisfiability) of $\mathcal{R}$ can be characterized by some *conflicts*, which are subsets of $V$ of minimum (finite) size that do not fulfill the property $\Pi$. To that aim, it is convenient to consider *dense* instances, that is instances that contain *exactly* one constraint for every $p$-sized subset of $V$. As a first example, we consider the Feedback Arc Set in Tournaments [5] problem, which is defined on oriented graphs (actually tournaments). Formally, the problem is as follows:

| Feedback Arc Set in Tournaments: |
|---|
| **Input**: A tournament $T := (V, A)$, an integer $k$. |
| **Parameter**: $k$. |
| **Question**: Does there exist a subset $F \subseteq A$ of size at most $k$ such that the tournament obtained by reversing the arcs of $F$ in $T$ is acyclic? |

A *constraint* for this particular problem is thus a *dominant vertex* for every pair of vertices.

**Known results** The Feedback Arc Set in Tournaments problem is $NP$-complete [4, 24], but fixed-parameter tractable [5, 49], the best algorithm running in time $O^*(2^{O(\sqrt{k})})$ [49]. In order

to design a parameterized algorithm for the problem, Alon et al. [5] obtained a kernel with at most $O(k^2)$ vertices. However, no linear vertex-kernel was known so far.

**Our contribution** Together with Stéphane Bessy, Fedor V. Fomin, Serge Gaspers, Christophe Paul, Saket Saurabh and Stéphan Thomassé, we prove that such a kernel can be obtained for Feedback Arc Set in Tournaments [6]. To do so, we heavily use a well-known result, stating that a tournament is *acyclic* if and only if it does not contain a directed circuit on three vertices. Hence, the notion of *conflict* for this particular problem referes to *directed circuit on three vertices*. Moreover, we introduce the notion of *safe partition*, which constitutes the core tool of our kernelization algorithm. We will see in Section 4.2 that this method can be used for other problems.

## 4.1 Safe partition

**Intuition** Let $R := (V, \mathcal{R})$ be an instance of the $\Pi$-Constraint Modification problem where the consistency of $\mathcal{R}$ can be characterized by some *conflicts*. The idea of a safe partition is as follows: assume that we can partition the collection of constraints $\mathcal{R}$ into two collections $\mathcal{R}_I$ et $\mathcal{R}_O$ such that the conflicts contained in $\mathcal{R}_O$ can be solved in such a way that the remaining conflicts are contained in $\mathcal{R}_I$. Using this fact, it is then possible to show that the editions of $\mathcal{R}_I$ and $\mathcal{R}_O$ can be done *independently*. Such a partition is defined as a *safe partition* if we can *certify* that the edition of a set of constraints $\mathcal{F} \subseteq \mathcal{R}_O$ that solves *all the conflicts contained in* $\mathcal{R}_O$ can be done in a *sound* manner. Thus, finding a so-called safe partition in polynomial time will allow us to reduce the given instance.

**Safe partition for tournaments** We now formally define a *safe partition* for the problem Feedback Arc Set in Tournaments. In the following, we consider *ordered tournaments* $T_\sigma := (V, A, \sigma)$ whose vertices are fixed under some ordering $\sigma$. Given a partition $\mathcal{P} := \{V_1, \ldots, V_l\}$ of $\sigma$ into sets of consecutive vertices, with $\sigma(V_i) < \sigma(V_j$ for $i < j$, we denote by $A_B$ the set of arcs whose extremities belong to different parts of $\mathcal{P}$, and $A_I := A \setminus A_B$. Moreover, we let $\mathcal{B}(A_B)$ denote the set of *backward arcs* of $A_B$, *i.e.* the set of arcs $uv \in A$ with $\sigma(v) < \sigma(u)$. Observe that given such a partition, the reversal of all backward arcs of $A_B$ solves the conflicts contained in $A_B$, and the remaining conflicts are contained in $A_I$. Roughly speaking, this means that the reversals inside the parts and between the parts can be done *independently*. Hence, since the reversal of the arcs of $\mathcal{B}(A_B)$ would solve *all* the conflicts lying between the parts, the aim is to give necessary conditions on $\mathcal{P}$ that *certify* the existence of a solution containing $\mathcal{B}(A_B)$. This can be done with the notion of safe partition.

**Definition 4.1 (Safe partition [6])** *Let $T_\sigma := (V, A, \sigma)$ be an ordered tournament and $\mathcal{P}$ be a partition of $T_\sigma$. We say that $\mathcal{P}$ is* safe *if $\mathcal{B}(A_B) \neq \emptyset$ and there exist $|\mathcal{B}(A_B)|$ arc-disjoint directed triangles using arcs of $A_B$ only.*

**Rule 3 ([6])** *Let $T_\sigma := (V, A, \sigma)$ be an ordered tournament and $\mathcal{P}$ be a safe partition of $T_\sigma$. Reverse all the arcs of $\mathcal{B}(A_B)$ and decrease $k$ by $|\mathcal{B}(A_B)|$.*

**Lemma 4.2 ([6])** *Rule 3 is sound.*

**Lemma 4.3 ([6])** *Let $T_\sigma := (V, A, \sigma)$ be an ordered tournament having at most $p$ backward arcs, with $|V| > 2p$. There exists a safe partition of $T_\sigma$ that can be computed in polynomial time.*

In order to apply Rule 3 we need to find an ordering that satisfies the conditions of Lemma 4.3. This can be done using any constant-factor approximation algorithm for Feedback Arc Set in Tournaments (which admits even a PTAS [50]). Indeed, the ordering given by the approximation algorithm (which contains a number of backward arcs bounded by a constant times $k$ by construction) can be used to construct a safe partition provided that the number of vertices is large enough. We hence have the following result.

**Theorem 4.4 ([9])** *For every $\epsilon > 0$, the* Feedback Arc Set in Tournaments *problem admits a kernel with at most $(2+\epsilon)k$ vertices.*

### 4.2 Conflict Packing: a unifying technique

We now study two problems related to Feedback Arc Set in Tournaments, namely Dense Betweenness and Dense Rooted Triplet Inconsistency. These problems are defined on a *dense* collection of constraints $\mathcal{R}$, respectively *betweenness constraints* and *rooted binary trees on three leaves*. We first consider the Dense Betweenness problem. A *betweenness constraint* $t$ is a pair $(\{a,b,c\},m)$ where $\{a,b,c\} \subseteq V$, and $m \in \{a,b,c\}$ defines the *choosen vertex* of the constraint $t$. For the sake of simplicity, we say that $t$ *chooses* the vertex $m$. Assuming w.l.o.g. that $m = b$, we say that $t$ *consistent* with an ordering $\sigma$ over $V$ if $\sigma(a) < b < \sigma(c)$ or $\sigma(c) < b < \sigma(a)$. A set of betweenness constraints $\mathcal{R}$ is *consistent* if there exists an ordering $\sigma$ on $V$ such that every $t \in \mathcal{R}$ is consistent with $\sigma$.

Dense Betweenness:
**Input**: A set $V$, a dense collection of betweenness constraints $\mathcal{R}$ defined over $V$, an integer $k$.
**Parameter**: $k$.
**Question**: Does there exist an ordering of $V$ consistent with all but at most $k$ betweenness triplets of $\mathcal{R}$?

Next, we consider the Dense Rooted Triplet Inconsistency problem. A *rooted triplet* $t$ is a rooted binary tree on a set of three leaves $V(t) = \{a,b,c\}$. We write $t = ab|c$ if $a$ and $b$ are siblings of a child of the root of $t$, the other child of the root being $c$. We also say that $t$ *chooses* $c$. Let $t \in \mathcal{R}$ be a rooted triplet and $T$ be a tree over $V$. Then $t$ is *consistent with* $T$ if the tree spanning the leaves of $T$ corresponding to $V(t)$ is homeomorphic to $t$, and *inconsistent* otherwise. A set of rooted triplets $\mathcal{R}$ is *consistent* if there exists a rooted binary tree $T$ over $V$ such that every $t \in \mathcal{R}$ is consistent with respect to $T$.

Dense Rooted Triplet Inconsistency :
**Input** : A set $V$, a dense collection of rooted binary trees on three leaves $\mathcal{R}$ defined over $V$, an integer $k$.
**Parameter** : $k$.
**Question** : Does there exists a rooted binary tree $T$ defined over $V$ that contains all but at most $k$ rooted binary trees of $\mathcal{R}$?

**Known results** There exist (subexponential) parameterized algorithms for both problems [5, 37], and Dense Rooted Triplet Inconsistency is known to admit a quadratic vertex-kernel [37].

As Feedback Arc Set in Tournaments, Dense Betweenness admits constant-factor approximation algorithms (there exists a PTAS for this problem as well [49]). The existence of such an algorithm together with a particular *sunflower-based* reduction rule allows us to obtain a linear vertex-kernel for this problem, that we will not detail here.

**Theorem 4.5** *For every $\epsilon > 0$, the* Dense Betweenness *problem admits a kernel with at most $(4 + \epsilon)k$ vertices.*

Nevertheless, due to the very nature of the problem, the *sunflower* technique is not enough to obtain a linear-vertex kernel for the Dense Rooted Triplet Inconsistency problem. However, the notion of *safe partition* can be used, implying that any constant-factor approximation algorithm would allow us to obtain a linear vertex-kernel for this problem as well. Unfortunately, no such algorithm is known for the Dense Rooted Triplet Inconsistency problem yet [37].

**Our contribution** In a joint work with Christophe Paul and Stéphan Thomassé [57], we use this observation to develop a technique used in a few parameterized problems [21, 32, 67] that we call *Conflict Packing*. This method allows us in particular to find the first linear kernel for Dense Rooted Triplet Inconsistency, improving the best known bound of $O(k^2)$ vertices [37]. Regarding the Dense Betweenness problem, we obtain a linear vertex-kernel, which constitutes the first polynomial kernel for this problem [49].

### 4.2.1 Definition and main idea

The aim of the use of a constant-factor approximation algorithm for Feedback Arc Set in Tournaments [6] is to compute a particular ordering from which a safe partition will then be computed. The aim of a Conflict Packing is to provide such a tool for problems that are not known to admit constant-factor approximation algorithms. In particular, Conflict Packing provides a lower bound on the number of editions required to obtain a consistent instance, which in turn will imply the existence of a particular *structure* (depending on the considered problem) that will be used to obtain a kernelization algorithm. Roughly speaking, the *Conflict Packing* technique thus *replaces* the constant-factor approximation algorithms and consists in the following steps:

- **Conflict Packing** : first, we greedily (*i.e.* in polynomial time) compute a conflict packing for the problem at hand. Roughly speaking, a conflict packing is a maximal family of edges (or constraints) disjoint conflicts. As we shall see, this definition needs to be refined for several problems, in particular when two conflicts can share a constraint but still require two distinct editions. In all cases, the aim of a conflict packing is to provide a lower bound on the number of editions required to obtain an instance satisfying the property $\Pi$.
- **Size** : given a *positive* instance of the considered problem, the next step consists in showing that a conflict packing $\mathcal{C}$ contains $O(k^c)$ vertices, for some constant $c > 0$. When the conflict packing is defined as a maximal family of edges (or constraints) disjoint conflicts, this result is trivial. However, when the notion of conflict packing needs to be refined, we will see that some technical arguments are required to bound the size of $\mathcal{C}$. Moreover, when this technique is applied on the $\mathcal{G}$-Vertex Deletion problem, one needs to use a *sunflower*-based reduction rule to obtain such a result.

- **Reduction** : the remaining step is now to bound the size of the set $V \setminus V(\mathcal{C})$, where $V(\mathcal{C}) \subseteq V$ denotes the set of vertices contained in $\mathcal{C}$. By maximality of the conflict packing $\mathcal{C}$, we know that the instance induced by $G_{\mathcal{C}} := G \setminus V(\mathcal{C})$ (or $\mathcal{R}_{\mathcal{C}} := \mathcal{R} \setminus V(\mathcal{C})$ obtained by the deletion of all constraints contained in $V(\mathcal{C})$) fulfills the property $\Pi$. Moreover, for every vertex $u \in V(\mathcal{C})$, the instance $G_{\mathcal{C}} \cup \{u\}$ (or $\mathcal{R}_{\mathcal{C}} \cup \{u\}$) also fulfills the property $\Pi$. Hence, bounding the size $V \setminus V(\mathcal{C})$ can be done using these two facts. This particular step is *problem-specific*, and cannot be described in a general manner.

This technique has already been used by Brügmann et al. [21] (resp. Fernau and Raible [32]) to obtain a linear (resp. quadratic) vertex-kernel for the problem Triangle Edge Deletion (resp. $P_3$-Packing). Another exemple of graph modification problem where this technique can be applied is Cluster Vertex Deletion [44]. The Conflict Packing technique provides a quadratic vertex-kernel for this problem, but does not improve the previous kernel result for this problem [1]. In what follows, we present several constraint modification problems on which this technique can be applied (Sections 4.2.2 and 4.2.4). We also mention a similar technique introduced by van Bevern et al. [65], namely *kernelization through tidying* (Section 4.2.3).

### 4.2.2 Linear vertex-kernel for Feedback Arc Set in Tournaments

We first illustrate this notion on the Feedback Arc Set in Tournaments problem. We will show that the Conflict Packing technique allows to obtain a linear vertex-kernel for this problem. Even if such a result was known to exist [6], the kernelization algorithm presented here uses simpler arguments, and does not require a constant-factor approximation algorithm anymore. We will see that this fact can be very useful when dealing with problems that do not admit constant-factor approximation algorithm yet (Section 4.2.4). We use the following rule from [5].

**Rule 4 ([5])** *Let $T := (V, A)$ be an instance of* Feedback Arc Set in Tournaments. *Remove from $T$ any vertex that does not belong to any directed triangle.*

We now state the main definitions and results of this Section.

**Definition 4.6 ([57])** *Let $T := (V, A)$ be an instance of* Feedback Arc Set in Tournaments. *A* conflict packing *of $T$ is a maximal collection $\mathcal{C}$ of arc-disjoint directed triangles.*

**Lemma 4.7 ([57])** *Let $T := (V, A)$ be a positive instance of* Feedback Arc Set in Tournaments *and $\mathcal{C}$ be a conflict packing of $T$. Then $|\mathcal{C}| \leqslant k$ and $|V(\mathcal{C})| \leqslant 3k$.*

**Lemma 4.8 (Conflict Packing [57])** *Let $T := (V, A)$ be an instance of* Feedback Arc Set in Tournaments *and $\mathcal{C}$ be a conflict packing of $T$. There exists an ordering $\sigma$ of $V$ that can be computed in polynomial time and whose backward arcs $uv$ verify $u, v \in V(\mathcal{C})$.*

We now prove that the Feedback Arc Set in Tournaments problem admits a kernelization algorithm returning an instance containing at most $4k$ vertices. As we already mentioned, such a result can be obtained by combining the notion of safe partition and a PTAS for Feedback Arc Set in Tournaments [50]. We give a sketch of the proof of Theorem 4.9, which avoides the use of the constant-factor approximation algorithm and makes use of the ordering provided by Lemma 4.8 instead.

**Theorem 4.9 ([57])** *The* Feedback Arc Set in Tournaments *problem admits a kernel with at most* $4k$ *vertices.*

*Sketch of the proof.* Let $T := (V, A)$ be a positive instance of Feedback Arc Set in Tournaments reduced under Rule 4, and $\sigma$ be an ordering obtained through Lemma 4.8. We construct a bipartite graphe $B := (I \cup V', E)$, where:

(i) $V' := V \setminus V(\mathcal{C})$,

(ii) there is a vertex $i_{vu}$ in $I$ for every backward arc $vu$ of $T_\sigma$ and,

(iii) $i_{vu}w \in E$ if $w \in V'$ and $\{u, v, w\}$ is a directed triangle.

Now, any matching of size (at least) $k+1$ in $B$ corresponds to a collection of (at least) $k+1$ arc-disjoint directed triangles, implying that $T := (V, A)$ is a negative instance, which is not by hypothesis. Thus, the size of a maximum matching in $B$ is bounded from above by $k$, and hence $B$ admits a vertex cover $D$ of size at most $k$. Let $D_1$ and $D_2$ denote the sets $D \cap I$ and $D \cap V'$, respectively. Observe in particular that $|D_2| \leqslant k$. Now, assuming that $|V| > 4k$, since $|V(\mathcal{C})| \leqslant 3k$ (Lemma 4.7), we have $|V'| > k$ and hence $(V' \setminus D_2) \neq \emptyset$. We use this fact to design the ordered partition $\mathcal{P} := \{V_1, \ldots, V_l\}$ of $T_\sigma$ where every part $V_i$ consists of either a vertex of $V' \setminus D_2$ or a maximal subset of consecutive vertices (in $\sigma$) of $V \setminus (V' \setminus D_2)$. Using again matching arguments and Hall's Theorem [43], we finally prove the following.

**Lemma 4.10** *The partition* $\mathcal{P}$ *is a safe partition of* $T_\sigma$*.*

Hence, as long as the number of vertices is greather than $4k$, there exists a safe partition that can be computed in polynomial time, and hence the input tournament can be reduced. $\diamond$

We would like to observe that this particular counting argument is not restricted to the Feedback Arc Set in Tournaments problem: indeed, such a bipartite graph can be designed for any constraint modification problem characterized by finite *conflicts*, as long as a well-suited structure can be computed in polynomial time (Lemma 4.8). This can be done using the Conflict Packing technique. Next, with this bipartite graph in hand, it is possible to conclude using again simple matching arguments. As we shall see, the Dense Rooted Triplet Inconsistency problem fits in this framework. Before presenting this result, we connect Conflict Packing with a recently-introduced method called *kernelization through tidying* [65] , which deals with vertex-deletion problems.

### 4.2.3 Link with kernelization through tidying

In [65], van Bevern et al. consider the $\mathcal{G}$-Vertex Deletion problem for graph classes $\mathcal{G}$ that can be characterized by a finite family of forbidden induced subgraphs $\mathcal{F}$. Their method works in three steps:

- **Approximation**: first, greedily (*i.e.* in polynomial time) compute an approximation set $S$ for the problem. By definition of $\mathcal{G}$, $S$ contains at most $hk$ vertices, where $h$ denotes the maximum number of vertices contained in a forbidden induced subgraph of $\mathcal{G}$. It remains to bound the number of vertices of the graph $G \setminus S$, which can be done using the fact that $G \setminus S$ is $\mathcal{F}$-free and the *tidying* step.

- **Tidying**: In a second step, compute a *tidying set* $T \subseteq V \setminus S$ using the *sunflower* reduction rule. Roughly speaking, we will then have $|S \cup T| \in O(k^2)$ and the graph $(G \setminus (S \cup T)) \cup \{u\}$ will be $\mathcal{F}$-free for any $u \in S$ (called *local tidiness property*).
- **Shrinking**: the last step is problem-specific, and consists in shrinking the graph $G \setminus (S \cup T)$ by using data reduction rules specific to the problem at hand and the local tidiness property. If the number of vertices of this particular graph can be reduced to $p(k)$ for some polynomial $p$, then we obtain a kernel containing $O(h^2k^2 + p(k))$ vertices.

Observe that our definition of conflict packing can be adapted in a similar way for this particular problem. Here, a conflict packing $\mathcal{C}$ would be a *maximal family of forbidden induced subgraphs pairwise intersecting in at most one vertex*. By definition, this constitutes an approximation set since the graph $G \setminus V(\mathcal{C})$ is $\mathcal{F}$-free. Using the *sunflower* reduction rule, it is not hard to show that $|V(\mathcal{C})| \leqslant h^2k^2$. Now, the graph $(G \setminus V(\mathcal{C}) \cup \{u\}$ is $\mathcal{F}$-free for any $u \in V(\mathcal{C}))$, which corresponds to the local tidiness property previously mentioned. Finally, the shrinking step can be applied in a similar way that in the kernelization through tidying technique.

In what follows, we explain how the Conflict Packing technique can be refined to obtain linear-vertex kernels for several constraint modification problems. As we will see, considering a *maximal collection of constraint disjoint conflict* is no longer enough for such problems.

### 4.2.4 Linear vertex-kernel Dense Rooted Triplet Inconsistency

We describe how this technique can be used to obtain a linear vertex-kernel for the Dense Rooted Triplet Inconsistency problem, improving the best known bound of $O(k^2)$ vertices [37]. As mentioned before, the aim of a conflict packing is to provide a lower bound on the size of the solution by considering a maximal family of particular *conflicts*. Recall that a conflict is a minimum subset of $V$ of (finite) size that does not fulfill the property $\Pi$. In the following, we use the term conflict to denote both the subset of vertices of $V$ and the set of constraints contained in this particular subset. In the case of Dense Rooted Triplet Inconsistency, it is not sufficient to consider *constraint-disjoint* conflicts. Indeed, in this problem, two conflicts may *share a rooted triplet* but still require *two distinct* editions. To see this, let $\{a, b, c, d, e\}$ be a set of leaves and consider the following conflicts: $\mathcal{C} = \{ab|c, ac|d, ad|b, cd|b\}$ and $\mathcal{C}' = \{ed|c, ed|b, bc|e, bd|c\}$. Observe first that $\mathcal{C}$ remains a conflict for any choice of $\{b, c, d\}$. Since $\mathcal{C}$ and $\mathcal{C}'$ only have this rooted triplet in common, no edition on $\mathcal{C}'$ can solve $\mathcal{C}$. Hence (at least) two distinct editions are required to solve both $\mathcal{C}$ and $\mathcal{C}'$. Hence, we need to refine the definition of conflict packing, which can be done using the notion of *seed*.

**Definition 4.11 (Seed [57])** *Let $R := (V, \mathcal{R})$ be an instance of* Dense Rooted Triplet Inconsistency *and $C := \{a, b, c, d\}$ be a conflict of $R$. We say that $a$ is a* seed *of $C$ is $C$ is a conflict for any choice of $\{b, c, d\}$.*

In other words, the set $\{\{a, b, c\}, \{a, c, d\}, \{a, b, d\}\}$ is a conflict of $R$ independently from the choice of $\{b, c, d\}$.

**Definition 4.12 ([57])** *Let $R := (V, \mathcal{R})$ be an instance of* Dense Rooted Triplet Inconsistency. *A* conflict packing *of $R$ is a maximal sequence of conflicts $\mathcal{C} = \{C_1, C_2, \ldots, C_l\}$ such that for every $2 \leq i \leq l$:*

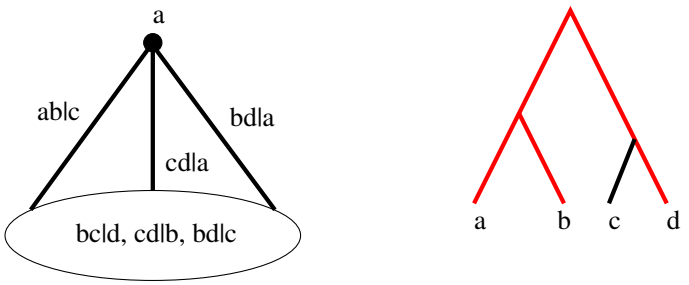


Figure 7: The notion of *seed* for Dense Rooted Triplet Inconsistency. The set $\{ab|c, cd|a, bd|a\}$ is a conflict for any choice of $\{b, c, d\}$.

- *Either $C_i$ intersects $\cup_{1 \leqslant j<i} C_j$ on at most two leaves or,*
- *$C_i$ has a unique leaf not belonging to $\cup_{1 \leqslant j<i} C_j$, which is a seed of $C_i$.*

The kernelization algorithm follows the same lines than the one for Feedback Arc Set in Tournaments: a conflict packing $\mathcal{C}$ contains $O(k)$ vertices (otherwise the instance is negative and we simply return a trivial No-instance of constant size) and the instance induced by $V \setminus V(\mathcal{C})$ is *consistent*. We can use this fact to obtain an *embedded* tree for $R$.

**Lemma 4.13 ([57])** *Let $R := (V, \mathcal{R})$ be an instance of* Dense Rooted Triplet Inconsistency *and $\mathcal{C}$ be a conflict packing of $R$. There exists a rooted binary tree $T$ over $V$ that can be computed in polynomial time and whose inconsistent triplets $t$ verify $V(t) \in V(\mathcal{C})$.*

Using Lemma 4.13, it is then possible to compute a safe partition in polynomial time, providing that $V$ contains more than $5k$ vertices. To adapt the notion of safe partition for this particular problem, we define a *tree partition* of an embedded instance $R_T := (V, \mathcal{R}, T)$. We say that $\mathcal{P} = \{T_1, \ldots, T_l\}$ is a *tree partition* of $V$ if there exist $l$ nodes and leaves $x_1, \ldots, x_l$ of $T$ such that: $(i)$ for every $i \in [l]$ $T_i = T_{x_i}$ and $(ii)$ the set of leaves in $\cup_{i=1}^{l} T_{x_i}$ partition $V$. A tree partition of $R_T$ naturally distinguishes two sets of rooted triplets: $\mathcal{R}_I = \{t \in \mathcal{R} \mid \exists i \in [l]\ V(t) \subseteq V(T_i)\}$ and $\mathcal{R}_O = \mathcal{R} \setminus \mathcal{R}_I$. Now, the definition of a safe partition is similar to Definition 4.1. Altogether, this yields to the following result.

**Theorem 4.14 ([9])** *The* Dense Rooted Triplet Inconsistency *problem admits a kernel with at most $5k$ vertices.*

**Dense Betweenness** To conclude, we would like to mention that the Conflict Packing technique can also be applied to the Dense Betweenness problem. Unlike Feedback Arc Set in Tournaments and Dense Rooted Triplet Inconsistency, this kernelization algorithm do not require the notion of safe partition. Instead, using the order providing by the Conflict Packing Lemma, it is possible to use a *sunflower-based* reduction rule in order to conclude.

**Theorem 4.15 ([9])** *The* Dense Betweenness *problem admits a kernel with at most $5k$ vertices.*

# 5 Part III. Beyond kernelization

In the last part of this thesis, we mention some techniques that can be used when no polynomial-time reduction rules can be found for the problem at hand. In particular, in a joint work with Jean Daligault, Christophe Paul and Stéphan Thomassé [26], we used the notion of *irrelevancy* [54] and graph minor theory [60, 61] to prove that the Multicut problem can be reduced in FPT time to instances of treewidth bounded by a function of $k$. Determining the parameterized complexity of this problem was a long-standing open question [27], and our result has been used by Bousquet et al. to obtain a parameterized algorithm for the problem [18]. Formally, Multicut is defined as follows:

Multicut:
**Input**: A graph $G := (V, E)$, a set of requests (*i.e.* pairs of vertices of $V$) $R$, an integer $k$.
**Parameter**: $k$.
**Question**: Does there exist a set $F \subseteq E$ of size at most $k$ whose removal *disconnects* every request of $R$?

**Theorem 5.1 ([26])** *The* Multicut *problem can be reduced to instances of treewidth bounded by a function of $k$.*

**Reducing the graph** Let $(G, R, k)$ be an instance of the Multicut problem such that $G$ has *large* treewidth (with respect to parameter $k$). Due to a famous theorem of Robertson and Seymour [60], it is known that when a graph has large treewidth then it contains a large *clique* or *grid* minor (with respect to the parameter $k$). We make use of this structural theorem to obtain structural information on the possible solutions for the problem. The core tools of our approach lie in the study of some connectivity problems of independent interest. In particular, we prove that the Triple Separation problem is FPT. Given three vertices $x, y, z$ of a graph $G := (V, E)$, we define a $(zy|x)$-*cut of size* $k$ as a set of $k$ edges whose deletion in $G$ disconnects $z$ from $x$ but does not disconnect $z$ from $y$. Formally, the Triple Separation problem can be defined as follows.

Triple Separation:
**Input**: A graph $G := (V, E)$, three vertices $x, y, z$, an integer $k$.
**Parameter**: $k$.
**Question**: Does there exist a $(zy|x)$-cut of size at most $k$?

The second connectivy problem that we consider is based on the notion of *strong connectivity*. Given a graph $G := (V, E)$, a set $T \subseteq V$, two integers $k$ and $k'$ and a vertex $z$ of $G$, we say that a vertex $x \notin T$ is $k'$-*strongly* $(z|T)$-$k$-*connected* if for every subset $S \subseteq T$ such that $|S| \geqslant |T| - k'$, there is no $(zx|S)$-cut. In other words, for every such subset $S$, every cut of size at most $k$ between $z$ and $S$ disconnects $x$ from $S$. We prove that whenever the set $T$ has sufficiently large size (with respect to parameter $k$), then there exists a vertex $x \in T$ which is $k'$-strongly $(z|T \setminus \{x\})$-$k$-connected and that can be found in FPT time. In particular, such a result allows us to obtain a parameterized reduction rule as follows. We say that a request $r \in R$ is *irrelevant* if the instances $(G, R, k)$ and $(G, R \setminus \{r\}, k)$ are equivalent. Assume there exists a vertex $u \in V$ that is incident to more than $k^{O(k)}$ requests. Hence, using the previous connectivity properties, we can find in FPT time an irrelevant request incident to that vertex, and hence reduce the graph.

These two results allow us to obtain the following reduction rules for the MULTICUT problem. A set $T \subseteq V$ is *gathered* if for every set $F \subseteq E$ of size at most $k$, there exists at most one connected component of $G \setminus F$ containing at least two vertices of $T$.

**Rule 5 ([26])** *Let $G := (V, E)$ be an instance of* MULTICUT.

- *If a vertex is incident to more than $k^{O(k)}$ requests, then it is incident to an irrelevant request.*
- *If $G$ is reduced by the previous reduction rule, and if there exists a gathered set of sufficiently large size, then there exists an irrelevant request that can be identified in FPT time.*

**Bounding the treewidth** In order to bound the treewidth of the graph, we use the graph minor theory [60, 61]. We first consider the case when the graph contains a large clique minor, using the previous reduction rules to reduce the graph. Next, if the graph contains a large grid minor but no clique minor, we are able to prove that the graph contains either a large gathered set of terminals (and hence an irrelevant request), or that there exists an edge that can safely be contracted. In all cases, as long as the treewidth of the graph $G$ is large enough, then we can reduce the graph in FPT time.

**Consequences** To conclude on the MULTICUT problem, we mention one of the main consequence of our treewidth-reduction, which has been used by Bousquet et al. [18] to design a parameterized algorithm for MULTICUT. The *request graph* $R_G$ of an instance $(G, R, k)$ of MULTICUT contains the vertices $V$ of $G$ as vertices, and there is an edge between two vertices $u, v \in V(R_G)$ iff $(u, v) \in R$.

**Theorem 5.2 ([26])** *Let $(G, R, k)$ be an instance of* MULTICUT. *There exists a function $f$ such that the degree of the request graph $R_G$ is bounded by $f(k)$.*

# 6 Conclusion

## 6.1 Our results

In this thesis we have proved several polynomial kernels for graph (and constraint) modification problems. To that aim, we developed mainly two techniques, namely *branches* for graph modification problems and *Conflict Packing* for constraint modification problems. We believe that these techniques will be useful to design kernelization algorithms for several other modification problems. In particular, we obtained polynomial kernels for CLOSEST 3-LEAF POWER, PROPER INTERVAL COMPLETION, answering two questions left open by Dom et al. [29] and Kaplan et al. [46]. Moreover, our polynomial kernels are the first known for the problems. Therefore, since it is known that not all parameterized problems admit polynomial kernels unless $NP \subseteq coNP/poly$, our results are relevant from a theoretical viewpoint.

## 6.2 Open problems

We now mention several open problems related with the work presented in this extended abstract. First of all, the bounds obtained on the size of some kernels seem to let room for improvement. Hence, this raises the following question.

**Problem 1** *Can we improve the polynomial kernels obtained for* Closest 3-Leaf Power, Proper Interval Completion *and* Cograph Edition*?*

We now mention several known open problems in kernelization, that fit to the context of *branches*. Notice that the two problems we mention are known to be fixed-parameter tractable [54, 66].

**Problem 2** *Do the* Chordal Deletion *and the* Interval Completion *problems admit polynomial kernels?*

Regarding constraint modification problems, it would be nice to determine on which problems the Conflict Packing technique can be used. In particular, there exists several $NP$-hard problems defined on hard instances [3] that seem to be candidates to fit in this framework.

**Problem 3** *Can the Conflict Packing be applied (or generalized) to other problems defined on dense instances?*

Moreover, in order to have a better understanding on the relations between problems defined on dense instances, it would be nice to answer the following question.

**Problem 4** *Does the* Dense Rooted Triplet Inconsistency *problem admit a constant-factor approximation algorithm?*

Finally, we address one of the main open problem related with the work done on Multicut.

**Problem 5** *Does the* Multicut *problem admit a polynomial kernel?*

## References


[1] F. N. Abu-Khzam. Kernelization algorithms for d-hitting set problems. In *Algorithms and Data Structures Symposium (WADS)*, volume 4619 of *Lecture Notes in Computer Science*, pages 434–445. Springer, 2007.

[2] F. N. Abu-Khzam, M. R. Fellows, M. A. Langston, and W. Henry S. Crown structures for vertex cover kernelization. *Theory of Computing Systems*, 41(3):411–430, 2007.

[3] Nir Ailon and Noga Alon. Hardness of fully dense problems. *Information and Computation*, 205(8):1117–1129, 2007.

[4] N. Alon. Ranking tournaments. *SIAM Journal on Discrete Mathematics*, 20(1):137–142, 2006.

[5] N. Alon, D. Lokshtanov, and S. Saurabh. Fast FAST. In *Automata, Languages and Programming (ICALP)*, volume 5555 of *LNCS*, pages 49–58. Springer, 2009.

[6] S. Bessy, F. V. Fomin, S. Gaspers, C. Paul, A. Perez, S. Saurabh, and S. Thomassé. Kernels for feedback arc set in tournaments. In *Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 4 of *LIPIcs*, pages 37–47, 2009.

[7] S. Bessy, F. V. Fomin, S. Gaspers, C. Paul, A. Perez, S. Saurabh, and S. Thomassé. Kernels for feedback arc set in tournaments. *Journal of Computer System and Sciences (JCSS)*, 2011. to appear.

[8] S. Bessy, C. Paul, and A. Perez. Polynomial kernels for 3-leaf power graph modification problems. In *IWOCA*, volume 5874 of *Lecture Notes in Computer Science*, pages 72–82, 2009.

[9] S. Bessy, C. Paul, and A. Perez. Polynomial kernels for 3-leaf power graph modification problems. *Discrete Applied Mathematics*, 158(16):1732–1744, 2010.

[10] S. Bessy and A. Perez. Polynomial kernels for proper interval completion and a related problem. In *to appear in FCT 2011*, 2011.

[11] H. Bodlaender, R. Downey, M. Fellows, and D. Hermelin. On problems without polynomial kernels. In *Automata, Languages and Programming (ICALP)*, number 5125 in LNCS, pages 563–574, 2008.

[12] H. L. Bodlaender. Kernelization: New upper and lower bound techniques. In *International Workshop on Parameterized and Exact Computation (IWPEC)*, pages 17–37, 2009.

[13] H. L. Bodlaender, L. Cai, J. Chen, M. R. Fellows, J. A. Telle, and D. Marx. Open problems in parameterized and exact computation. In *International Workshop on Parameterized and Exact Computation (IWPEC)*, 2006.

[14] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. On problems without polynomial kernels. *Journal of Computer and System Sciences*, 75(8):423–434, 2009.

[15] H. L. Bodlaender, F. V. Fomin, D. Lokshtanov, E. Penninkx, S. Saurabh, and D. M. Thilikos. (Meta) Kernelization. In *Symposium on Foundations of Computer Science (FOCS)*, pages 629–638, 2009.

[16] H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch. Cross-composition: A new technique for kernelization lower bounds. In *Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 9 of *Leibniz International Proceedings in Informatics*, pages 165–176, 2011.

[17] H. L. Bodlaender, S. Thomassé, and A. Yeo. Kernel bounds for disjoint cycles and disjoint paths. In *European Symposium on Algorithms (ESA)*, volume 5757 of *LNCS*, pages 635–646, 2009.

[18] N. Bousquet, J. Daligault, and S. Thomassé. Multicut is FPT. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, pages 459–468. ACM, 2011.

[19] A. Brandstädt and V.B. Le. Structure and linear time recognition of 3-leaf powers. *Information Processing Letters*, 98(4):133–138, 2006.

[20] A. Brandstädt, V.B. Le, and R. Sritharan. Structure and linear time recognition of 4-leaf powers. *ACM Transactions on Algorithms (TALG)*, 5(1):1–22, 2008.

[21] D. Brügmann, C. Komusiewicz, and H. Moser. On generating triangle-free graphs. *Electronic Notes in Discrete Mathematics*, 32:51–58, 2009.

[22] L. Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58(4):171–176, 1996.

[23] M-S. Chang and M-T. Ko. The 3-steiner root problem. In *International Workshop on Graph Theoretical Concepts in Computer Science (WG)*, pages 109–120, 2007.

[24] P. Charbit, S. Thomassé, and A. Yeo. The minimum feedback arc set problem is NP-hard for tournaments. *Combinatorics, Probability and Computing*, 16(1):1–4, 2007.

[25] J. Chen and J. Meng. A 2 k kernel for the cluster editing problem. In *International Computing and Combinatorics Conference (COCOON)*, volume 6196 of *Lecture Notes in Computer Science*, pages 459–468, 2010.

[26] J. Daligault, C. Paul, A. Perez, and S. Thomassé. Treewidth reduction for the parameterized Multicut problem. *http://www.lirmm.fr/∼daligault/MulticutTreewidthReduction.pdf*, 2010.

[27] E. D. Demaine, M. Hajiaghayi, and D. Marx. 09511 open problems – parameterized complexity and approximation algorithms. In *Parameterized complexity and approximation algorithms*, number 09511 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2010.

[28] M. Dom, J. Guo, F. Hüffner, and R. Niedermeier. Error compensation in leaf root problems. In *International Symposium on Algorithms and Computation (ISAAC)*, number 3341 in Lecture Notes in Computer Science, pages 389–401, 2004.

[29] M. Dom, J. Guo, F. Hüffner, and R. Niedermeier. Error compensation in leaf root problems. *Algorithmica*, 44:363–381, 2006.

[30] M. Dom, J. Guo, F. Hüffner, and R. Niedermeier. Closest 4-leaf power is fixed-parameter tractable. *Discrete Applied Mathematics*, 156(18):3345–3361, 2008.

[31] R.G. Downey and M.R. Fellows. *Parameterized complexity.* Springer, 1999.

[32] H. Fernau and D. Raible. A parameterized perspective on packing paths of length two. *Journal of Combinatorial Optimization*, 18(4):319–341, 2009.

[33] J. Flum and M. Grohe. *Parameterized Complexity Theory.* Text in Theoretical Computer Science. Springer, 2006.

[34] L. Fortnow and R. Santhanam. Infeasibility of instance compression and succinct PCPs for NP. In *ACM Symposium on Theory of Computing (STOC)*, pages 133–142, 2008.

[35] P. Galinier, M. Habib, and C. Paul. Chordal graphs and their clique graphs. In *International Workshop on Graph Theoretical Concepts in Computer Science (WG)*, volume 1017 of *Lecture Notes in Computer Science*, pages 358–371, 1995.

[36] M.R. Garey and D.S. Jonhson. *Computers and intractability: a guide to the theory on NP-Completeness.* W.H. Freeman and Co., 1979.

[37] S. Gaspers and M. Mnich. On feedback vertex sets in tournaments. In *European Symposium on Algorithms (ESA)*, volume 6346 of *LNCS*, pages 267–277, 2010.

[38] M. Golumbic, H. Kaplan, and R. Shamir. On the complexity of DNA physical mapping. *Advances in Applied Mathematics (ADVAM)*, 15, 1994.

[39] 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.

[40] S. Guillemot, C. Paul, and A. Perez. On the (non-)existence of polynomial kernels for $P_l$-free edge modification problems. In *International Symposium on Parameterized and Exact Computation (IPEC)*, volume 6478 of *Lecture Notes in Computer Science*, pages 147–157, 2010.

[41] J. Guo. A more effective linear kernelization for cluster editing. In *International Symposium Combinatorics, Algorithms, Probabilistic and Experimental Methodologies (ESCAPE)*, volume 4614 of *Lecture Notes in Computer Science*, pages 36–47, 2007.

[42] M. Habib and C. Paul. A survey on algorithmic aspects of modular decomposition. *Computer Science Review*, 4(1):41–59, 2010.

[43] P. Hall. On representatives of subsets. *Journal of the London Mathematical Society*, 10(37):26–30, 1935.

[44] F. Hüffner, C. Komusiewicz, H. Moser, and R. Niedermeier. Fixed-parameter algorithms for cluster vertex deletion. *Theory of Computing Systems*, 47(1):196–217, 2010.

[45] I. A. Kanj, M. J. Pelsmajer, and M. Schaefer. Parameterized algorithms for feedback vertex set. In *International Workshop on Parameterized and Exact Computation (IWPEC)*, volume 3162 of *Lecture Notes in Computer Science*, pages 235–247, 2004.

[46] H. Kaplan, R. Shamir, and R. E. Tarjan. Tractability of parameterized completion problems on chordal and interval graphs: Minimum fill-in and physical mapping. In *Symposium on Foundations of Computer Science (FOCS)*, pages 780–791, 1994.

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

[48] M. Karpinski and W. Schudy. Approximation schemes for the betweenness problem in tournaments and related ranking problems. *CoRR*, abs/0911.2214, 2009.

[49] M. Karpinski and W. Schudy. Faster algorithms for feedback arc set tournament, kemeny rank aggregation and betweenness tournament. In *International Symposium on Algorithms and Computation (ISAAC)*, pages 3–14, 2010.

[50] C. Kenyon-Mathieu and W. Schudy. How to rank with few errors. In *ACM Symposium on Theory of Computing (STOC)*, pages 95–103, 2007.

[51] S. Kratsch and M. Wahlström. Two edge modification problems without polynomial kernels. In *International Workshop on Parameterized and Exact Computation (IWPEC)*, volume 5917 of *LNCS*, pages 264–275, 2009.

[52] G.H. Lin, P.E. Kearney, and T. Jiang. Phylogenetic k-root and steiner k-root. In *International Symposium on Algorithms and Computation (ISAAC)*, number 1969 in Lecture Notes in Computer Science, pages 539–551, 2000.

[53] P. J. Looges and S. Olariu. Optimal greedy algorithms for indifference graphs. *Computers & Mathematics with Applications*, 25(7):15 – 25, 1993.

[54] D. Marx. Chordal deletion is fixed-parameter tractable. *Algorithmica*, 57(4):747–768, 2010.

[55] R. Niedermeier. *Invitation to fixed parameter algorithms*, volume 31 of *Oxford Lectures Series in Mathematics and its Applications*. Oxford University Press, 2006.

[56] R. Niedermeier and P. Rossmanith. A general method to speed up fixed-parameter-tractable algorithms. *Information Processing Letters*, 73(3-4):125–129, 2000.

[57] C. Paul, A. Perez, and S. Thomassé. Conflict packing yields linear-vertex kernels for $k$-FAST, $k$-DENSE RTI and a related problem. In *to appear in MFCS 2011*, 2011.

[58] A. Perez. *Algorithmes de noyau pour des problmes d'dition de graphes et autres structures*. PhD thesis, Universit Montpellier II - LIRMM, Montpellier, FRANCE, 2011.

[59] F. Protti, M. Dantas da Silva, and J.L. Szwarcfiter. Applying modular decomposition to parameterized cluster editing problems. *Theory of Computing Systems*, 2007.

[60] N. Robertson and P.D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Combin. Theory, Ser. B*, 63(1):65–110, 1995.

[61] N. Robertson, P.D. Seymour, and R. Thomas. Quickly excluding a planar graph. *J. Combin. Theory Ser. B*, 62(2):323–348, 1994.

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

[63] R. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 13:566–579, 1984.

[64] S. Thomassé. A $4k^2$ kernel for feedback vertex set. *ACM Transactions on Algorithms*, 6(2), 2010.

[65] R. van Bevern, H. Moser, and R. Niedermeier. Kernelization through tidying. In *Latin American Theoretical Informatics Symposium (LATIN)*, volume 6034 of *Lecture Notes in Computer Science*, pages 527–538. Springer, 2010.

[66] Y. Villanger, P. Heggernes, C. Paul, and J. A. Telle. Interval completion is fixed parameter tractable. *SIAM Journal on Computing*, 38(5):2007–2020, 2009.

[67] J. Wang, D. Ning, Q. Feng, and J. Chen. An improved kernelization for $P_3$-packing. *Information Processing Letters*, 110(5):188–192, 2010.

[68] G. Wegner. *Eigenschaften der nerven homologische-einfactor familien in* $\mathbb{R}^n$. PhD thesis, Universität Gottigen, Gottingen, Germany, 1967.

# A Publications

The works presented in this thesis have been published in several international conferences and journals. We present here a list of these publications.

**International Conferences**

- [8] **Polynomial kernels for** 3-LEAF POWER GRAPH MODIFICATION **problems**. With Stéphane BESSY and Christophe PAUL. **IWOCA 2009**, number 5874 in *Lecture Notes in Computer Science*, pages 72-80, 2009.

- [6] **Kernels for** FEEDBACK ARC SET IN TOURNAMENTS. With Stéphane BESSY, Fedor V. FOMIN, Serge GASPERS, Christophe PAUL, Saket SAURABH and Stéphan THOMASSÉ. **FSTTCS 2009**, number 4 in *Leibnitz International Proceedings in Informatics*, pages 37-47, 2009.

- [40] **On the (non-)existence of polynomial kernels for** $P_l$-FREE EDGE MODIFICATION **problems**. With Sylvain GUILLEMOT and Christophe PAUL. **IPEC 2010**, number 6478 in *Lecture Notes in Computer Science*, pages 147-157, 2010.

- [57] **Conflict Packing yields linear vertex-kernels for** DENSE ROOTED TRIPLET INCONSISTENCY **and related problems**. With Christophe PAUL and Stéphan THOMASSÉ. **MFCS 2011**, *to appear*.

- [10] **Polynomial kernels for** PROPER INTERVAL COMPLETION **and related problems**. With Stéphane BESSY. **FCT 2011**, *to appear*.

**Journals**

- [9] **Polynomial kernels for** 3-LEAF POWER GRAPH MODIFICATION **problems**. With Stéphane BESSY and Christophe PAUL. *Discrete Applied Mathematics 158* (2010) pp. 1732-1744.

- [7] **Kernels for** FEEDBACK ARC SET IN TOURNAMENTS. With Stéphane BESSY, Fedor V. FOMIN, Serge GASPERS, Christophe PAUL, Saket SAURABH and Stéphan THOMASSÉ. Accepted for publication at *Journal of Computer and System Sciences*, 2010.

**Preprint**

- [26] **Treewidth reduction for the parameterized** MULTICUT **problem**. With Jean DALIGAULT, Christophe PAUL and Stéphan THOMASSÉ. *Preprint*, 2010.

Discrete Applied Mathematics 158 (2010) 1732–1744

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

# Polynomial kernels for 3-leaf power graph modification problems☆

Stéphane Bessy [b], Christophe Paul [a], Anthony Perez [b,*]

[a] CNRS - LIRMM, Université Montpellier 2, 161 rue Ada 34932 Montpellier, France

[b] LIRMM - Université Montpellier 2, 161 rue Ada 34932 Montpellier, France

ARTICLE INFO

*Article history:*
Received 23 December 2008
Received in revised form 10 June 2010
Accepted 5 July 2010
Available online 13 August 2010

*Keywords:*
Algorithms
Data-structures
FPT
Kernel
Graph modification problems
Leaf power

ABSTRACT

A graph $G = (V, E)$ is a 3-leaf power iff there exists a tree $T$ the leaf set of which is $V$ and such that $uv \in E$ iff $u$ and $v$ are at distance at most 3 in $T$. The 3-leaf power graph edge modification problems, *i.e.* edition (also known as the CLOSEST 3-LEAF POWER), completion and edge-deletion are FPT when parameterized by the size of the edge set modification. However, polynomial kernels were known for none of these three problems. For each of them, we provide kernels with $O(k^3)$ vertices that can be computed in linear time. We thereby answer an open problem first mentioned by Dom et al. (2004) [8].



## 0. Introduction

The combinatorial analysis of experimental data-sets naturally leads to graph modification problems. For example, extracting a threshold graph from a dissimilarity on a set is a classical technique used in clustering and data analysis to move from a numerical to a combinatorial data-set [1,16]. The edge set of the threshold graph aims at representing the pairs of elements which are close to each another. As the dissimilarity reflects some experimental measures, the edge set of the threshold graph may reflect some false positive or negative errors. So for the sake of cluster identification, the edge set of the threshold graph has to be edited in order to obtain a disjoint union of cliques. This problem, known as CLUSTER EDITING, is fixed-parameter tractable (see e.g. [12,13,25]) and efficient parameterized algorithms have been proposed to solve biological instances with about 1000 vertices and several thousand edge modifications [2,7]. So, motivated by the identification of some hidden combinatorial structures on experimental data-sets, edge-modification problems cover a broad range of classical graph optimization problems, among which are *completion* problems, *edition* problems and *edge-deletion* problems (see [19] for a recent survey). Precisely, for a given graph $G = (V, E)$, in a completion problem, the set $F$ of modified edges is constrained to be disjoint from $E$, whereas in an edge deletion problem $F$ has to be a subset of $E$. If no restriction applies to $F$, then we obtain an edition problem. Though most of the edge-modification problems turn out to be NP-hard problems, efficient algorithms can be obtained to solve the natural parameterized version of some of them. Indeed, as long as the number $k$ of errors generated by the experimental process is not too large, one can afford a time complexity exponential in $k$. A problem is *fixed parameterized tractable* (FPT for short) with respect to parameter $k$ whenever it can be solved in time $f(k) \cdot n^{O(1)}$, where $f(k)$ is an arbitrary computable function. Here, the natural parameterization is the

☆ Work supported by the French research grant ANR-06-BLAN-0148-01 "Graph Decomposition and Algorithms – GRAAL".

* Corresponding author. Fax: +33 467 8500.
*E-mail addresses:* bessy@lirmm.fr (S. Bessy), paul@lirmm.fr (C. Paul), perez@lirmm.fr (A. Perez).

0166-218X/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2010.07.002

number $k = |F|$ of modified edges. The generic question is thereby whether for fixed $k$, a given edge modification problem is tractable. More formally:

Parameterized $\Pi$-modification Problem
**Input:** An undirected graph $G = (V, E)$.
**Parameter:** An integer $k \geqslant 0$.
**Question:** Is there a subset $F \subseteq V \times V$ with $|F| \leqslant k$ such that the graph $G + F = (V, E \triangle F)$ satisfies $\Pi$.

This paper studies the parameterized version of edge modification problems and more precisely the existence of a polynomial *kernel*. A problem is kernelizable if every instance $(G, k)$ can be reduced in polynomial time (using *reduction rules*) into an instance $(G', k')$ such that $k' \leqslant k$ and the size of $G'$ is bounded by a function of $k$. The membership to the FPT complexity class is equivalent to the property of having a kernel (see [20] for example). Having a kernel of small size is clearly highly desirable [15]. Indeed, preprocessing the input in order to reduce its size while preserving the existence of a solution is an important issue in the context of various applications ([15]). However, the equivalence mentioned above only provides an exponential bound on the kernel size. For a parameterized problem, the challenge is then to know whether or not it admits a polynomial – or even linear (in $k$) – kernel (see *e.g.* [20]). The $k$-vertex cover problem is the classical example of a problem with a linear kernel. Recently, parameterized problems (among which the $k$-treewidth problem) have been shown to not have polynomial kernels [3] (unless some collapse occurs in the computational complexity hierarchy).

This paper follows this line of research and studies the kernelization of edge-modification problems related to the family of *leaf powers*, graphs arising from a phylogenetic reconstruction context [17,18,21]. The goal is to extract, from a threshold graph $G$ on a set $S$ of species, a tree $T$, whose leaf set is $S$ and such that the distance between two species is at most $p$ in $T$ iff they are adjacent in $G$ ($p$ being the value used to extract $G$ from dissimilarity information). If such a tree $T$ exists, then $G$ is a *$p$-leaf power* and $T$ is its *$p$-leaf root*. For $p \leqslant 5$, the $p$-leaf power recognition is polynomial time solvable [5,6], whereas the question is still open for $p$ strictly larger than 5. Parameterized $p$-leaf power edge modification problems have been studied so far for $p \leqslant 4$. The edition problem for $p = 2$ is known as the Cluster Editing problem for which the kernel size bound has been successively improved in a series of recent papers [12,13,24], culminating in [14] with a kernel with $4k$ vertices. For larger values of $p$, the edition problem is known as the closest $p$-leaf power problem. For $p = 3$ and 4, the closest $p$-leaf power problem is known to be FPT [10,9], while its fixed parameterized tractability is still open for larger values of $p$. However, the existence of a polynomial kernel for $p \neq 2$ remained an open question [8,11]. Though the completion and edge-deletion problems are FPT for $p \leqslant 4$ [9,11], no polynomial kernel is known for $p \neq 2$ [14].

*Our results*. We prove that the closest 3-leaf power, the 3-leaf power completion and the 3-leaf power edge-deletion admit a kernel with $O(k^3)$ vertices. We thereby answer positively to the open question of Dom et al. [9,11].

*Outlines*. The first section is dedicated to some known and new structural results of 3-leaf powers and their related critical clique graphs. Section 2 describes the data-reduction rules for the closest 3-leaf power problem. The kernels for the other two variants, the 3-leaf power completion and the 3-leaf power edge-deletion problems, are presented in Section 3.

## 1. Preliminaries

The graphs we consider in this paper are undirected and loopless. The vertex set of a graph $G$ is denoted by $V(G)$, with $|V(G)| = n$, and its edge set by $E(G)$, with $|E(G)| = m$ (or $V$ and $E$ when the context is clear). The *open* neighborhood of a vertex $x$ is denoted by $N_G(x)$ (or $N(x)$), and its *closed* neighborhood (*i.e.* $N_G(x) \cup \{x\}$) by $N[x]$. Two vertices $x$ and $y$ of $G$ are *true twins* if they are adjacent and $N(x) = N(y)$. A subset $S$ of vertices is a *module* if for every distinct vertices $x$ and $y$ of $S$, $N(x) \setminus S = N(y) \setminus S$. Given a subset $S$ of vertices, $G[S]$ denotes the subgraph of $G$ induced by $S$. If $H$ is a subgraph of $G$, $G \setminus H$ stands for $G[V(G) \setminus V(H)]$. We write $d_G(u, v)$ the distance between two vertices $u$ and $v$ in $G$. For a subset $S \subseteq V$, $d_S(u, v)$ denotes the distance between $u$ and $v$ within $G[S]$, and is set to $\infty$ if $u$ and $v$ are not connected in $G[S]$. A graph family $\mathcal{F}$ is *hereditary* if for every graph $G \in \mathcal{F}$, every induced subgraph $H$ of $G$ also belongs to $\mathcal{F}$. For a set $\mathcal{S}$ of graphs, we say that $G$ is *$\mathcal{S}$-free* if none of the graphs of $\mathcal{S}$ is an induced subgraph of $G$.

As the paper deals with undirected graphs, we abusively denote by $X \times Y$ the set of unordered pairs containing one element of $X$ and one of $Y$. Let $G = (V, E)$ be a graph and $F$ be a subset of $V \times V$, we denote by $G + F$ the graph on vertex set $V$, the edge set of which is $E \triangle F$ (the symmetric difference between $E$ and $F$). Such a set $F$ is called an *edition* of $G$ (we may also abusively say that $G + F$ is an edition). We improperly speak about edges of $F$, even if the elements of $F$ are not all edges of $G$. A vertex $v \in V$ is *affected* by an edition $F$ whenever $F$ contains an edge incident to $v$. Given a graph family $\mathcal{F}$ and given a graph $G = (V, E)$, a subset $F \subseteq V \times V$ is an *optimal $\mathcal{F}$-edition* of $G$ if $F$ is a set of minimum cardinality such that $G + F \in \mathcal{F}$. If we constrain $F$ to be disjoint from $E$, we say that $F$ is a *completion*, whereas if $F$ is asked to be a subset of $E$, then $F$ is an *edge deletion*.

### 1.1. Critical cliques

The notions of critical clique and critical clique graph have been introduced in [18] in the context of phylogenetic. More recently, they have been successfully used in various modification problems such as Cluster Editing [14] and Bicluster Editing [24].

**Definition 1.1.** A *critical clique* of a graph $G$ is a clique $K$ which is a module and is maximal under this property.

It follows from the definition that the set $\mathcal{K}(G)$ of critical cliques of a graph $G$ defines a partition of its vertex set $V$.

**Definition 1.2.** Given a graph $G = (V, E)$, its *critical clique graph* $\mathcal{C}(G)$ has a vertex set $\mathcal{K}(G)$ and edge set $E(\mathcal{C}(G))$ with

$$KK' \in E(\mathcal{C}(G)) \Leftrightarrow \forall\, v \in K, v' \in K', vv' \in E(G)$$

Let us note that given a graph $G$, its critical clique graph $\mathcal{C}(G)$ can be computed in linear time with the modular decomposition algorithm (see [26] for example).

The following lemma was used in the construction of polynomial kernels for Cluster Editing and Bicluster Editing problems in [24].

**Lemma 1.3.** *Let $G = (V, E)$ be a graph. If $H$ is the graph $G + \{(u, v)\}$ with $(u, v) \in V \times V$, then $|\mathcal{K}(H)| \leqslant |\mathcal{K}(G)| + 4$.*

The next lemma shows that for a range of graph families, critical cliques are robust under optimal edition.

**Lemma 1.4.** *Let $\mathcal{F}$ be an hereditary graph family closed under true twin addition. For every graph $G = (V, E)$, there exists an optimal $\mathcal{F}$-edition (resp. $\mathcal{F}$-deletion, $\mathcal{F}$-completion) $F$ such that every critical clique of $G + F$ is the disjoint union of a subset of critical cliques of $G$.*

**Proof.** We prove the statement for the edition problem. Same arguments apply for edge deletion and edge completion problems.

Let $F$ be an optimal $\mathcal{F}$-edition of $G$ such that the number $i$ of critical cliques of $G$ which are clique modules in $H = G + F$ is maximum. Denote $\mathcal{K}(G) = \{K_1, \ldots, K_c\}$ and assume that $i < c$ (*i.e* $K_1, \ldots, K_i$ are clique modules in $H$ and $K_{i+1}, \ldots, K_c$ are no longer clique modules in $H$). Let $x$ be a vertex of $K_{i+1}$ such that the number of edges of $F$ incident to $x$ is minimum among the vertices of $K_{i+1}$. Roughly speaking, we will modify $F$ by editing all vertices of $K_{i+1}$ like $x$. Let $H_x$ be the subgraph $H \setminus (K_{i+1} \setminus \{x\})$. As $\mathcal{F}$ is hereditary, $H_x$ belongs to $\mathcal{F}$ and, as $\mathcal{F}$ is closed under true twin addition, reinserting $|K_{i+1}| - 1$ true twins of $x$ in $H_x$ results in a graph $H'$ belonging to $\mathcal{F}$. It follows that $F' = E(G) \triangle E(H')$ is an $\mathcal{F}$-edition of $G$. By the choice of $x$, we have $|F'| \leqslant |F|$. Finally let us remark that, now, $K_1, \ldots, K_i$ and $K_{i+1}$ are clique modules of $H'$, thus proving the lemma. □

In other words, for an hereditary graph family $\mathcal{F}$ which is closed under true twin addition and for every graph $G$, there always exists an optimal $\mathcal{F}$-edition $F$ (resp. $\mathcal{F}$-deletion, $\mathcal{F}$-completion) such that :

(1) every edge between two vertices of a critical clique of $G$ is an edge of $G + F$, and
(2) between two distinct critical cliques $K, K' \in \mathcal{K}(G)$, either $V(K) \times V(K') \subseteq E(G+F)$ or $(V(K) \times V(K')) \cap E(G+F) = \emptyset$.

From now on, every considered optimal edition (resp. deletion, completion) is supposed to verify these two properties.

### *1.2. Leaf powers*

**Definition 1.5.** Let $T$ be an unrooted tree whose leaves are one-to-one mapped to the elements of a set $V$. The *$k$-leaf power* of $T$ is the graph $T^k$, with $T^k = (V, E)$ where $E = \{uv \mid u, v \in V \text{ and } d_T(u, v) \leqslant k\}$. We call $T$ a *$k$-leaf root* of $T^k$.

It is easy to see that for every $k$, the $k$-leaf power family of graphs satisfies the conditions of Lemma 1.4. In this paper we focus on the class of 3-leaf powers for which several characterizations are known.

**Theorem 1.6** (*[4,10]*)**.** *For a graph $G$, the following conditions are equivalent:*

1. *$G$ is a 3-leaf power.*
2. *$G$ is {bull, dart, gem, $C_{\geqslant 4}$ }-free, $C_{\geqslant 4}$ being the cycles of length at least 4. (see Fig. 1).*
3. *The critical clique graph $\mathcal{C}(G)$ is a forest.*

The parameterized 3-leaf power edition problem, with respect to parameter $k$ being the size of the edited set, is tractable. An $O((2k+8)^k \cdot nm)$ algorithm is proposed in [10]. The proofs of our kernel for the 3-leaf power edition problem rely on the critical clique graph characterization and on the following new one which is based on the join composition of graphs.

*Join composition.* Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two disjoint graphs and let $S_1 \subseteq V_1$ and $S_2 \subseteq V_2$ be two non empty subsets of vertices. The *join composition* of $G_1$ and $G_2$ on $S_1$ and $S_2$, denoted $(G_1, S_1) \otimes (G_2, S_2)$, results in the graph $H = (V_1 \cup V_2, E_1 \cup E_2 \cup (V(S_1) \times V(S_2)))$ (see Fig. 2).

**Theorem 1.7.** *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two connected 3-leaf powers. The graph $H = (G_1, S_1) \otimes (G_2, S_2)$, with $S_1 \subseteq V_1$ and $S_2 \subseteq V_2$, is a 3-leaf power if and only if one of the following conditions holds:*

1. *$S_1$ and $S_2$ are two cliques of $G_1$ and $G_2$ respectively, and if $S_1$ (resp. $S_2$) is not critical, then $G_1$ (resp. $G_2$) is a clique or,*
2. *there exists a vertex $v \in V_1$ such that $S_1 = N[v]$ and $S_2 = V_2$ is a clique.*

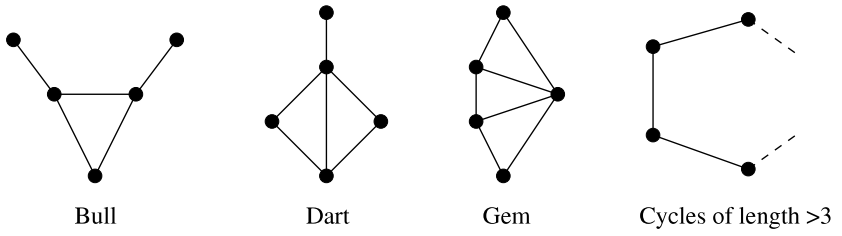


**Fig. 1.** Forbidden induced subgraphs of a 3-leaf power.

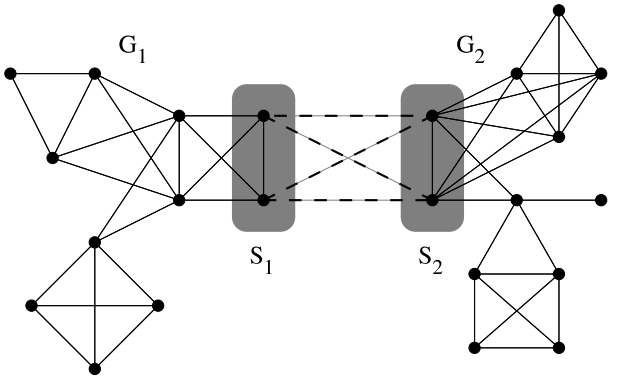


**Fig. 2.** The join composition $H = (G_1, S_1) \otimes (G_2, S_2)$ creates the doted edges. As $G_1$ and $G_2$ are two 3-leaf powers and as the subsets $S_1$ and $S_2$ of vertices are critical cliques of respectively $G_1$ and $G_2$, by Theorem 1.7, $H$ is also a 3-leaf power.

**Proof.** $\Leftarrow$ If condition (2) holds, then $H$ is obtained from $G_1$ by adding true twins to $v$, hence $H$ is a 3-leaf power. Assume $S_1$ and $S_2$ are two cliques. If $S_1$ and $S_2$ are both critical cliques of respectively $G_1$ and $G_2$, then the critical clique graph $\mathcal{C}(H)$ is clearly the tree obtained from $\mathcal{C}(G_1)$ and $\mathcal{C}(G_2)$ by adding the edges between $S_1$ and $S_2$. By Theorem 1.6, $H$ is a 3-leaf power. For $i = 1$ or 2, if $G_i$ is a clique and $S_i \subset V(G_i)$, then $S_i$ and $V(G_i) \setminus S_i$ are critical cliques in $H$. Again, it is easy to see that $\mathcal{C}(H)$ is a tree.

$\Rightarrow$ First, let us notice that if $S_1$ and $S_2$ are not cliques, then $H$ contains a $C_4$, which is forbidden. So let us assume that $S_1$ is not a clique but $S_2$ is. Then $S_1$ contains two non-adjacent vertices $x$ and $y$. First of all, if $d_{S_1}(x, y) = \infty$ (*i.e.* if $x$ and $y$ are not connected in $G_1[S_1]$), we consider $\pi_{G_1}$ a shortest path in $G_1$ between the connected component of $G_1[S_1]$ containing $x$ and the one containing $y$ (such a path exists because $G_1$ is connected). It is easy to see that $v$ together with $\pi_{G_1}$ forms an induced (chordless) cycle in $H$, which is forbidden. Now, if $d_{S_1}(x, y) > 2$, then $H$ contains a *gem*. To see this, let $\pi_{S_1}$ be a shortest $x, y$-path in $S_1$. Together with any vertex $v \in S_2$, the vertices of $\pi_{S_1}$ induce a cycle at length at least 5 in $H$. By construction the only possible chords are incident to $v$. So any 4 consecutive vertices on $\pi_{S_1}$ plus the vertex $v$ induce a *gem*. It follows that there exists in $S_1$ a vertex $u$ which dominates $x$ and $y$. Now if there exists a vertex in $V(G_2) \setminus S_2$, as $G_2$ is connected, there exists two adjacent vertices, $v \in S_2$ and $w \in V(G_2) \setminus S_2$. But, $\{w, u, x, y, v\}$, induce a *dart* in $H$, which is also forbidden. So, $S_2 = V(G_2)$ and $G_2$ is a clique. Finally, assume by contradiction again that $u$ has a neighbor $w \in V(G_1) \setminus S_1$. Considering a vertex $v$ of $S_2$, the set of vertices $\{w, x, y, u, v\}$ induces an obstruction in $H$, whatever the adjacency between $w$ and $\{x, y\}$ is. So, $N[u] \subset S_1$. Conversely, if $S_1$ contains a vertex $w \notin N(u)$, $\{w, x, y, u, v\}$ induces an obstruction in $H$. So, $S_1 = N[u]$, as expected in condition (2).

Assume now that both $S_1$ and $S_2$ are cliques. If $S_1$ and $S_2$ are not modules in respectively $G_1$ and $G_2$, then we can find a *bull* in $H$. Assume that only $S_1$ is not a module, *i.e.* there exist $x, y \in S_1$ and $u \in V(G_1) \setminus S_1$ such that *w.l.o.g.* $ux \in E(G_1)$ and $uy \notin E(G_1)$. If $S_2 \neq V(G_2)$, then again $H$ has a *bull* induced by $\{u, x, y, v, w\}$ with $v \in S_2$ and $w \in V(G_2) \setminus S_2$, $w$ neighbor of $v$. Otherwise, either condition (2) holds or $y$ has a neighbor $w$ in $V(G_1) \setminus S_1$. The latter case is impossible since we find in $H$ an obstruction induced by $\{u, x, y, v, w\}$ whatever the adjacency between $w$ and $\{u, x\}$ is. Finally assume that $S_1$ and $S_2$ are modules, but consider the case where $S_1$ is not critical (the case $S_2$ is not critical is symmetric). Then there exists a critical clique $C_1 \in \mathcal{K}(G_1)$ containing $S_1$. Denote by $x$ a vertex of $S_1$ and by $y$ a vertex of $C_1 \setminus S_1$. If $V(G_1) \neq C_1$, then $G_1$ contains two non-adjacent vertices, say $u$ and $u'$. If $u = x$ and $u' \notin C_1$, then as $G_1$ is connected, we can choose $u'$ and $w \notin C_1$ such that $\{u', w, x, y, v\}$ with $v \in S_2$ is a *bull* in $H$. Otherwise we can choose $u$ and $u'$ both adjacent to the vertices of $C_1$, and then $\{u, u', x, y, v\}$ would induce a *dart* in $H$. It follows that if $S_1$ is not critical, then condition (1) holds. □

The following observation will be helpful to apply Theorem 1.7 in the safeness' proofs of the reduction rules.

**Observation 1.8.** *Let $C$ be a critical clique of a 3-leaf power $G = (V, E)$. For every $S \subseteq V$, if the clique $C \setminus S$ is not critical in $G[V \setminus S]$, then the connected component of $G[V \setminus S]$ containing $C \setminus S$ is a clique.*

**Proof.** Assume that $C \setminus S$ is not a critical clique of $G[V \setminus S]$, *i.e.* though $C \setminus S$ is a clique module in $G[V \setminus S]$, it is not maximal. Let $x \notin S$ be a vertex such that $C \cup \{x\}$ is a clique module of $G[V \setminus S]$. Then $x$ belongs to a critical clique $C'$ of $G$ adjacent to $C$ in $\mathcal{C}(G)$. It follows that $S$ has to contain the union of all the critical cliques of $G$ adjacent to $C$ in $\mathcal{C}(G)$ but $C'$ (otherwise $C \cup \{x\}$ could not be a module of $G[V \setminus S]$), and all the critical cliques of $G$ adjacent to $C'$ in $\mathcal{C}(G)$ but $C$ (for the same reason). This means that the connected component containing $C \setminus S$ in $G[V \setminus S]$ is a subset of $C \cup C'$ which is a clique. □

Finally, let us conclude this preliminary study of 3-leaf powers by a technical lemma required in the proof of the last reduction rule.

**Lemma 1.9.** *Let $G = (V, E)$ be a 3-leaf power. Every cycle $C$ of length at least 5 in $G$ contains four distinct vertices $a, b, c, d$ (appearing in this order along $C$) with $ab$ and $cd$ edges of $C$ such that $ad \in E$, $ac \in E$ and $bd \in E$.*

**Proof.** As the 3-leaf power graphs form a hereditary family, the subgraph $H$ of $G$ induced by the vertices of the cycle $C$ is a 3-leaf power with at least 5 vertices. As $H$ is not a tree, it contains a critical clique $K$ of size at least 2. Let $a$ and $d$ be two distinct vertices of $K$. As $|C| \geq 5$, observe that there exist two distinct vertices $b$ and $c$, distinct from $a$ and $d$, such that $a$, $b$, $c$ and $d$ appear in this order along $C$ and such that $ab$ and $cd$ are edges of $C$. As $K$ is a clique module, any vertex adjacent to some vertex in $K$ neighbors all the vertices of $K$. It follows that $ad \in E$, $ac \in E$ and $bd \in E$. □

## 2. A kernel for CLOSEST 3-LEAF POWER

In this section, we present five preprocessing reduction rules the application of which leads to a kernel with $O(k^3)$ vertices for the CLOSEST 3-LEAF POWER problem. The first rule gets rid of connected components of the input graph that are already 3-leaf powers. Rule 2.1 is trivially safe.

**Rule 2.1.** *If $G$ has a connected component $C$ such that $G[C]$ is 3-leaf power, then remove $C$ from $G$.*

The next rule was already used to obtain a kernel with $O(k^2)$ vertices for the parameterized CLUSTER EDITING problem [24]. It bounds the size of every critical clique in a reduced instance by $k + 1$.

**Rule 2.2.** *If $G$ has a critical clique $K$ such that $|K| > k + 1$, then remove $|K| - k - 1$ vertices of $K$ from $V(G)$.*

**Proof.** By Lemma 1.4, we know that there always exists an optimal 3-leaf power edition that contains none or every edge incident to a critical clique $K$. Thus, if $|K| \geqslant k + 1$, this means that there is no optimal 3-leaf power edition that contains an edge incident to $K$. As this is still true if $|K| = k + 1$, it is safe to remove $|K| - (k + 1)$ vertices of $K$ from $V(G)$ (meaning that every optimal 3-leaf power edition in the reduced graph will also be an optimal 3-leaf power edition in the input graph). □

### 2.1. Branch reduction rules

We now assume that the input graph $G$ is reduced under Rule 2.1 (*i.e.* none of the connected components is a 3-leaf power) and Rule 2.2 (*i.e.* critical cliques of $G$ have size at most $k + 1$). The next three reduction rules use the fact that the critical clique graph of a 3-leaf power is a forest. The idea is to identify induced subgraphs of $G$, called *branches*, which correspond to subtrees of $\mathcal{C}(G)$. That is, a branch of $G$ is an induced subgraph which is already a 3-leaf power. More precisely:

**Definition 2.1.** Let $G = (V, E)$ be a graph. An induced subgraph $G[S]$, with $S \subseteq V$, is a *branch* if $S$ is the disjoint union of critical cliques $K_1, \ldots, K_r \in \mathcal{K}(G)$ such that the subgraph of $\mathcal{C}(G)$ induced by $\{K_1, \ldots, K_r\}$ is a tree.

Let $B = G[S]$ be a branch of a graph $G$ and let $K_1, \ldots, K_r$ be the critical cliques of $G$ contained in $S$. We say that $K_i$ $(1 \leqslant i \leqslant r)$ is an *attachment point* of the branch $B$ if it contains a vertex $x$ such that $N_G(x)$ intersects $V(G) \setminus S$. A branch $B$ is a *l-branch* if it has exactly $l$ attachment points. Our next three rules deal with 1-branches and 2-branches.

In the following, we denote by $B^R$ the subgraph of $B$ in which the vertices of the attachment points have been removed. If $P$ is an attachment point of $B$, then the set of neighbors of vertices of $P$ in $B$ is denoted $N_B(P)$.

**Lemma 2.2.** *Let $G = (V, E)$ be a graph and $B$ be a 1-branch of $G$ with attachment point $P$. There exists an optimal 3-leaf power edition $F$ of $G$ such that :*

1. *the set of affected vertices of $B$ is a subset of $P \cup N_B(P)$ and*
2. *in $G + F$, the vertices of $N_B(P)$ are all adjacent to the same vertices of $V(G) \setminus V(B^R)$.*

**Proof.** Let $F$ be an arbitrary optimal 3-leaf power edition of $G$. We construct from $F$ another optimal 3-leaf power edition which satisfies the two conditions above.

Let $C$ be the critical clique of $H = G + F$ that contains $P$ and set $C' = C \setminus B^R$. By Lemma 1.4, the set of critical cliques of $G$ whose vertices belong to $N_B(P)$ contains two kinds of cliques: $K_1, \ldots, K_c$, whose vertices are in $C$ or adjacent to the vertices of $C$ in $H$, and $K_{c+1}, \ldots, K_h$ whose vertices are not adjacent to the vertices of $C$ is $H$. For $i \in \{1, \ldots, h\}$, let $C_i$ be the connected component of $B^R$ containing $K_i$.

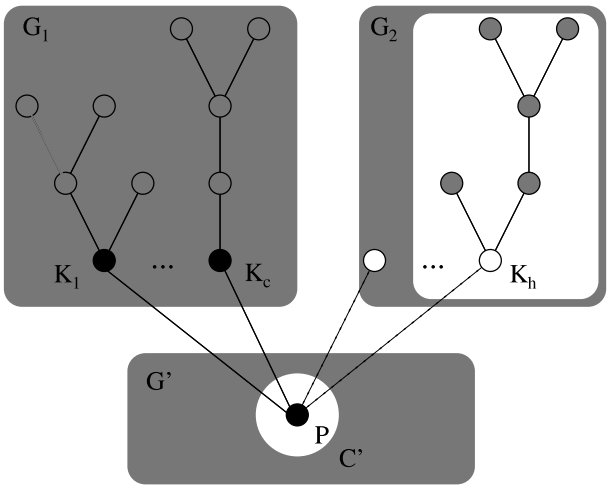


**Fig. 3.** A 1-branch of $G$ where all editions adjacent to vertices in $V(B^R) \setminus N_B(P)$ have been removed. This shows in particular the appearance of the three subgraphs we consider.

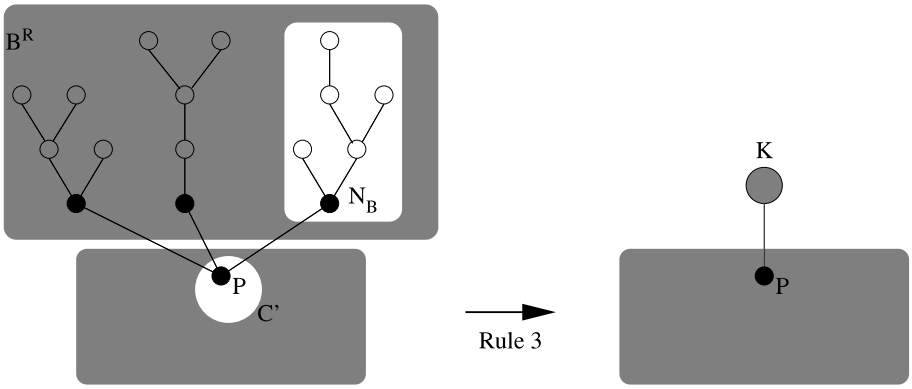


**Fig. 4.** On the left, a 1-branch $B$ with attachment point $P$. On the right, the effect of Rule 2.3 which replace $B^R$ by a clique $K$ of size $\min\{|N_B(P)|, k+1\}$.

Let us consider the three following induced subgraphs: $G_1$ the subgraph of $G$ induced by the disjoint union of $C_1, \ldots, C_c$; $G_2$ the subgraph of $G$ induced by the disjoint union of $C_{c+1}, \ldots, C_h$; and finally $G'$, the subgraph of $H$ induced by $V(G) \setminus V(B^R)$. Let us notice that these three graphs are 3-leaf powers.

By Observation 1.8, if $C'$ is not a critical clique of $G'$, then the connected component of $G'$ containing $C'$ is a clique. Similarly, if $K_i$, for every $1 \leqslant i \leqslant c$, is not a critical clique of $G_1$, then the connected component of $G_1$ containing $K_i$ is a clique. Thus, by Theorem 1.7, the disjoint union $H'$ of $G_2$ and $(G', C') \otimes (G_1, \{K_1, \ldots, K_c\})$ is a 3-leaf power (Fig. 3). By construction, the edge edition set $F'$ such that $H' = G + F'$ is a subset of $F$ and thus $|F'| \leqslant |F|$. Moreover the vertices of $B$ affected by $F'$ all belong to $P \cup N_B(P)$, which proves the first point.

To state the second point, we focus on the relationship between the critical cliques $K_i$ and $C'$ in $H' = G + F'$. If some $K_i$ is linked to $C'$ in $H'$ (i.e. $c > 1$), it means that the cost of adding the missing edges between $K_i$ and $C'$ (which, by Theorem 1.7, would also result in a 3-leaf power) is lower than the cost of removing the existing edges between $K_i$ and $C'$: $|K_i| \cdot |C' \setminus P| \leqslant |K_i| \cdot |P|$. On the other hand, if some $K_j$ is not linked to $C'$ in $H'$ (i.e. $c < h$), we conclude that $|P| \leqslant |C' \setminus P|$. Finally, if both cases occur, we have $|P| = |C' \setminus P|$, and we can choose to add all or none of the edges between $K_i$ and $C'$. In all cases, we provide an optimal edition of $G$ into a 3-leaf power in which the vertices of $N_B(P)$ are all adjacent to the same vertices of $V(G) \setminus V(B^R)$. □

We can now state the first 1-branch reduction rule whose safeness follows from Lemma 2.2 (Fig. 4).

**Rule 2.3.** *If $G$ contains a 1-branch $B$ with attachment point $P$, then remove from $G$ the vertices of $B^R$ and add a new critical clique of size $\min\{|N_B(P)|, k+1\}$ adjacent to $P$.*

Our second 1-branch reduction rule considers the case where several 1-branches are attached to the rest of the graph by a join. The following lemma shows that under certain cardinality conditions, the vertices of such 1-branches are not affected by an optimal 3-leaf power edition.

**Lemma 2.3.** *Let $G = (V, E)$ be a graph for which a 3-leaf power edition of size at most $k$ exists. Let $B_1, \ldots, B_l$ be 1-branches, the attachment points $P_1, \ldots, P_l$ of which all have the same neighborhood $N$ in $V \setminus \cup_{i=1}^{l} V(B_i)$. If $\sum_{i=1}^{l} |P_i| > 2k + 1$, then, in every 3-leaf power optimal edition $F$ of $G$, $N$ has to be a critical clique of $H = G + F$ and none of the vertices of $\cup_{i=1}^{l} V(B_i)$ is affected.*

**Proof.** We just show that every optimal 3-leaf power edition $F$ of $G$ has to transform $N$ into a critical clique, which directly implies the second part of the result. Notice that since $G$ is reduced under Rule 2.2, every attachment point $P_i$ satisfies $|P_i| \leqslant k + 1$, thus implying that $l \geqslant 2$.

First, assume that $F$ does not edit $N$ into a clique: *i.e.* there are two vertices $a$ and $b$ of $N$ such that $ab \notin E(H)$. For every pair of vertices $u_i \in P_i$ and $u_j \in P_j$ with $i \neq j$, the set $\{a, b, u_i, u_j\}$ cannot induce a chordless cycle in $H$, which implies that the vertices of $P_i$ or those of $P_j$ must be affected. It follows that among the attachment points, the vertices of at most one $P_i$ are not affected by $F$. As $|P_i| \leqslant k + 1$ for every $i$, the size of $F$ has to be at least $k + 1$: contradicting the assumptions. So $N$ is a clique in $H$.

Now, assume that $N$ is not a module of $H$: *i.e.* there exists $w \notin N$ such that for some $x, y \in N$ we have w.l.o.g. $xw \in E(H)$ and $yw \notin E(H)$. As $|F| \leqslant k$, there exist two vertices $u_i \in P_i$ and $u_j \in P_j$, non affected by $F$ and such that $u_iu_j \notin E(H)$. Together with $x$, $y$ and $w$, $u_i$ and $u_j$ induce a *dart* in $H$, contradicting Theorem 1.6. So, in $H$, the set of vertices $N$ has to be a clique module.

Finally, let us notice that $N$ has to be critical in $H$, otherwise it would imply that there exists a vertex $v \notin N$ that has been made adjacent to at least $k + 1$ vertices of $\cup_{i=1}^{l} B_i$, implying that $|F| > k$: a contradiction. □

By Lemma 2.3, if there exists a 3-leaf power edition $F$ of $G$ such that $|F| \leqslant k$, then the 1-branches $B_1, \ldots, B_l$ can be safely replaced by two critical cliques of size $k + 1$. This gives us the second 1-branch reduction rule.

**Rule 2.4.** *If $G$ has several 1-branches $B_1, \ldots, B_l$, the attachment points $P_1, \ldots, P_l$ of which all have the same neighborhood $N$ in $V \setminus \cup_{i=1}^{l} V(B_i)$ and if $\sum_{i=1}^{l} |P_i| > 2k + 1$, then remove from $G$ the vertices of $\cup_{i=1}^{l} V(B_i)$ and add two new critical cliques of size $k + 1$ neighboring exactly $N$.*

### 2.2. The 2-branch reduction rule

Let us consider a 2-branch $B$ of a graph $G = (V, E)$ with attachment points $P_1$ and $P_2$. The subgraph of $G$ induced by the critical cliques of the unique path from $P_1$ to $P_2$ in $\mathcal{C}(B)$ is called the *main path* of $B$ and denoted $path(B)$. A min-cut of $path(B)$ is a set $F$ of edges of $B$ such that $B \setminus F$ does not contain any path from $P_1$ to $P_2$ and such that $F$ has minimal cardinality for this property. We say that $B$ is *clean* if $P_1$ and $P_2$ are leaves of $\mathcal{C}(B)$, in which case we denote by $Q_1$ and $Q_2$ the critical cliques which respectively neighbor $P_1$ and $P_2$ in $B$.

**Lemma 2.4.** *Let $B$ be a clean 2-branch of a graph $G = (V, E)$ with attachment points $P_1$ and $P_2$ such that $path(B)$ contains at least 5 critical cliques. There exists an optimal 3-leaf power edition $F$ of $G$ such that :*

1. *if $path(B)$ is a disconnected subgraph of $G + F$, then $F$ may contain a min-cut of $path(B)$;*
2. *and in each case, the other affected vertices of $B$ belong to $V(P_1 \cup Q_1 \cup P_2 \cup Q_2)$.*

**Proof.** Let $F$ be an arbitrary optimal 3-leaf power edition of $G$. We call $C_1$ and $C_2$ the critical cliques of $G+F$ that respectively contain $P_1$ and $P_2$ (possibly, $C_1$ and $C_2$ could be the same), and denote $C_1 \setminus B^R$ and $C_2 \setminus B^R$ respectively by $C_1'$ and $C_2'$ see Fig. 6. We will construct from $F$ another optimal 3-leaf power edition $F'$ of $G$ satisfying the statement.

- *Assume that $F$ disconnects $path(B)$.* First of all, it is clear that for every subset $F_1$ of $F$, if $F_2$ is an optimal edition of $H_1 = G+F_1$, then $F' = F_1 \cup F_2$ is an optimal edition of $G$. We use this fact in the following different cases. Assume that $F$ contains the edges $F_1 := V(P_1) \times V(Q_1)$ and consider the graph $H_1 := G + F_1$. We call $B_1$ the 1-branch $B \setminus P_1$ of $H_1$ whose attachment point is $P_2$. Then, Lemma 2.2 applies to $B_1$ and provides from $F$ an optimal 3-leaf power edition of $H_1$, say $F_2$, where the edited vertices of $B_1$ are contained in $V(P_2 \cup Q_2)$. By the previous observation, it follows that $F_1 \cup F_2$ is an optimal edition for $G$ that respects conditions (1) and (2). We proceed similarly if $F$ contains the edges $V(P_2) \times V(Q_2)$.
  Now, assume that $F$ does not contain $V(P_1) \times V(Q_1)$ nor $V(P_2) \times V(Q_2)$. In that case, there exists $F_1 \subset F$ which is a minimal cut of $path(B)$ disjoint from $V(P_1) \times V(Q_1)$ and $V(P_2) \times V(Q_2)$. Then, there are two connected components in $B + F_1$, the one containing $P_1$, say $B_1$, and the one containing $P_2$, say $B_2$. The subgraphs $B_1$ and $B_2$ of $H_1 := G + F_1$ are 1-branches with respectively $P_1$ and $P_2$ as attachment points. So, Lemma 2.2 applies to $B_1$ and $B_2$, and provides from $F$ an optimal 3-leaf power edition of $H_1$, say $F_2$, where the edited vertices of $B_1$ and $B_2$ are contained in $V(P_1 \cup P_2 \cup Q_1 \cup Q_2)$. To conclude, remark that if $F_1$ is not a minimum (for cardinality) cut of $path(B)$, we could replace $F_1$ by such a minimum cut, and perform a similar 3-leaf power edition for $G$ with size strictly lower than $|F|$, thus contradicting the choice of $F$. It follows that $F_1 \cup F_2$ is an optimal edition for $G$ that respects conditions (1) and (2).
- *Assume that $F$ does not disconnect $path(B)$.* Let $X_1$ (resp. $X_2$) be the connected component of $(G + F) \setminus B^R$ containing $P_1$ (resp. $P_2$).

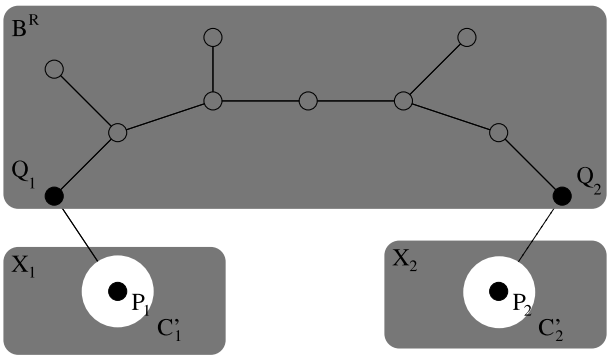


**Fig. 5.** A clean 2-branch of $G$. The bold edge represents the first join composition $H' := (X_1, C'_1) \otimes (B^R, Q_1)$ while the dotted bold edge represents the join composition which is done in a second place, namely $(H', Q_2) \otimes (X_2, C'_2)$.

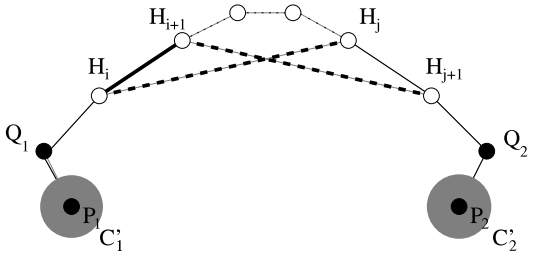


**Fig. 6.** An illustration of the first case, with $H_i \neq P_1$ and $H_{j+1} \neq P_2$. The bold edges represent the edges we put in $F'$ while the dotted bold edges denote the edges we removed from $F$ to obtain $F'$.

We first consider the case where $X_1$ and $X_2$ are two distinct connected components. By definition, $B^R$ is a 3-leaf power and $Q_1$ and $Q_2$ are two of its critical cliques (since $path(B)$ contains at least 5 critical cliques). Moreover, the subgraph $X_1$ (resp. $X_2$) is also a 3-leaf power which is a clique if $C'_1$ (resp. $C'_2$) is not a critical clique (Observation 1.8). By Theorem 1.7, it follows that the composition of these three subgraphs: $H' = (X_1, C'_1) \otimes (B^R, Q_1)$ and $(H', Q_2) \otimes (X_2, C'_2)$ yields a 3-leaf power (Fig. 5). Thus, if $F$ affects some vertices of $V(B^R) \setminus V(Q_1 \cup Q_2)$, then a smaller edition could be found by removing from $F$ the edges in $V(B^R) \times V(B^R)$. This would contradict the optimality of $F$.

So, assume that $P_1$ and $P_2$ belong to the same connected component $X$ of $(G+F) \setminus B^R$. Let $y_1$ and $y_2$ be respectively vertices of $P_1$ and $P_2$ (in the case $C_1 = C_2$, choose $y_1 = y_2$). Let $\pi_B$ and $\pi_X$ be two distinct paths between $y_1$ and $y_2$ defined as follows : $\pi_B$ is obtained by picking one vertex $b_i$ in each critical clique $H_i$ of $path(B)$ ($H_1 = P_1$ and $H_q = P_2$, with $q \geqslant 5$) and $\pi_X$ is a chordless path in $X$ (thereby its vertices $x_1, \ldots, x_p$, with $x_1 = y_1$ and $x_p = y_2$ belong to distinct critical cliques of $G + F$, say $K_1, \ldots, K_p$, with $K_1 = C'_1$ and $K_p = C'_2$). The union of these two paths results in a cycle $C$ of length at least 5. By Lemma 1.9, there are two disjoint edges $e = ab$ and $f = cd$ in $C$ such that the edges $(a, c)$ and $(b, d)$ belong to $E \triangle F$. By construction of $C$, at most one of the edges $e$ and $f$ belongs to $\pi_X$ (otherwise $\pi_X$ would not be chordless). We now study the different cases :

*Edges e and f belong to* $\pi_B$. W.l.o.g assume that $a = b_i$, $b = b_{i+1}$ and $c = b_j$, $d = b_{j+1}$ ($i + 1 < j$). By Lemma 1.4, $F$ contains the set of edges $(V(H_i) \times V(H_j)) \cup (V(H_{i+1}) \times V(H_{j+1}))$. Notice that $\min\{|H_i| \cdot |H_{i+1}|, |H_j| \cdot |H_{j+1}|\} < |H_i| \cdot |H_j| + |H_{i+1}| \cdot |H_{j+1}|$. W.l.o.g., assume that $\min\{|H_i| \cdot |H_{i+1}|, |H_j| \cdot |H_{j+1}|\} = |H_i| \cdot |H_{i+1}|$. We will 'cut' the edges between $H_i$ and $H_{i+1}$. Consider the set :

$$F' = (F \setminus (V \times V(B^R))) \cup (V(H_i) \times V(H_{i+1})).$$

Moreover, if $H_i \neq P_1$, add to $F'$ the edges $F_{C_1} := V(C'_1 \setminus P_1) \times V(Q_1)$ (which belong to $F$) and, if $H_{j+1} \neq P_2$, add to $F'$ the edges $F_{C_2} := V(C'_2 \setminus P_2) \times V(Q_2)$ (which belong to $F$). In all cases, we have $|F'| < |F|$. As in the case where $X_1$ and $X_2$ were distinct connected components, by Theorem 1.7, the graph $G + F'$ is a 3-leaf power : contradicting the optimality of $F$.

*Edge e belongs to* $\pi_B$ *and f to* $\pi_X$. W.l.o.g., assume that $a = b_i$ and $b = b_{i+1}$, and $c = x_{j+1}$, $d = x_j$. As above, by Lemma 1.4, $F$ contains $(V(H_i) \times V(K_{j+1})) \cup (V(H_{i+1}) \times V(K_j))$. Notice that $\min\{|H_i| \cdot |H_{i+1}|, |K_j| \cdot |K_{j+1}|\} < |H_i| \cdot |K_{j+1}| + |H_{i+1}| \cdot |K_j|$. If $\min\{|H_i| \cdot |H_{i+1}|, |K_j| \cdot |K_{j+1}|\} = |H_i| \cdot |H_{i+1}|$, then we consider the set :

$$F' = (F \setminus (V \times V(B^R))) \cup (V(H_i) \times V(H_{i+1})).$$

Here again, if $H_i \neq P_1$, add to $F'$ the edges $F_{C_1}$ (which belong to $F$) and, if $H_{j+1} \neq P_2$, add to $F'$ the edges $F_{C_2}$ (which belong to $F$). As previously, $|F'|$ is smaller than $|F|$ and by Theorem 1.7, we can prove that $G + F'$ is a 3-leaf power. Finally, if $\min\{|H_i| \cdot |H_{i+1}|, |K_j| \cdot |K_{j+1}|\} = |K_j| \cdot |K_{j+1}|$, then we consider the set

$$F' = (F \setminus (V \times V(B^R))) \cup (V(K_j) \times V(K_{j+1})) \cup F_{C_1} \cup F_{C_2}.$$

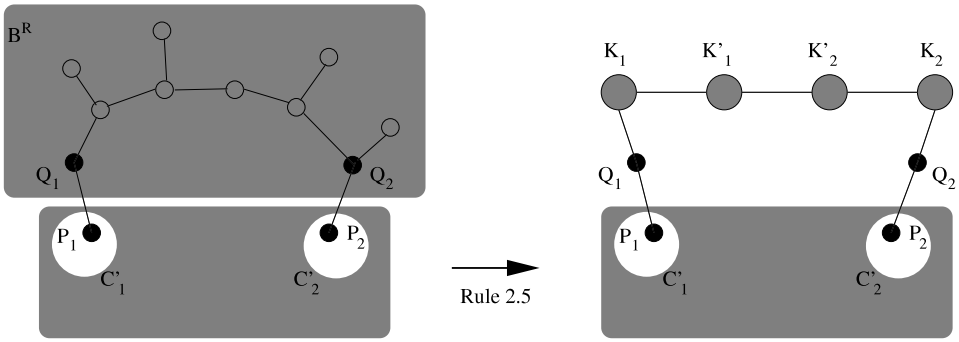


**Fig. 7.** A 2-branch $B$ on the left (only pendant critical cliques are hanging on $path(B)$ since we can assume that the graph is reduced by the previous rules). On the right, the way Rule 2.5 reduces $B$.

Again $|F'|$ is smaller than $|F|$ and by Theorem 1.7, we can prove that $G + F'$ is a 3-leaf power. In each case, we found a better 3-leaf power edition $F'$, contradicting the optimality of $F$. □

**Rule 2.5.** *Let $G$ be a graph having a clean 2-branch $B$ such that $path(B)$ is composed by at least 5 critical cliques. Remove from $G$ all the vertices of $V(B)$ except those of $V(P_1 \cup Q_1 \cup P_2 \cup Q_2)$ and add four new critical cliques :*

- *$K_1$ (resp. $K_2$) of size $k+1$ adjacent to $Q_1$ (resp. $Q_2$);*
- *$K'_1$ (resp $K'_2$) adjacent to $K_1$ (resp. $K_2$) and such that $K'_1$ and $K'_2$ are adjacent and $|K'_1| \cdot |K'_2|$ equals the min-cut of $path(B)$.*

**Proof.** Let $B'$ be the 2-branch replacing $B$ after the application of the rule. It is easy to see that by construction the min-cut of $B'$ equals the min-cut of $path(B)$. Moreover the attachment points $P_1$ and $P_2$ and their respective neighbors $Q_1$ and $Q_2$ are unchanged. It follows by Lemma 2.4 that every optimal edition $F$ of $G$ corresponds to an optimal edition $F'$ of $G'$, the graph reduced by Rule 2.5, such that $|F| = |F'|$ (Fig. 7). □

### *2.3. Kernel size and time complexity*

Let us discuss the time complexity of the reduction rules. The 3-leaf power recognition problem can be solved in $O(n+m)$ [4]. It follows that Rule 2.1 requires linear time. To implement the other reduction rules, we first need to compute the critical clique graph $\mathcal{C}(G)$. As noticed in [24], $\mathcal{C}(G)$ can be built in $O(n+m)$. For instance, to do so, we can compute in linear time the modular decomposition tree of $G$, which is a classical and well-studied problem in algorithmic graph theory (see [26] for a recent paper). Then, a critical clique is a serie-node of the decomposition tree with only leaves below it. Given $\mathcal{K}(G)$, which is linear in the size of $G$, it is easy to detect the critical cliques of size at least $k+1$. So, Rule 2.2 requires linear time. A search on $\mathcal{C}(G)$ can identify the 1-branches. It follows that the two 1-branch reduction rules (Rule 2.3 and Rule 2.4) can also be applied in $O(n+m)$ time. Let us now notice that in a graph reduced by the first four reduction rules, a 2-branch is a path to which pendant vertices are possibly attached. It follows that to detect a 2-branch $B$, such that $path(B)$ contains at least 5 critical cliques, we first prune the pendant vertices, and then identify in $\mathcal{C}(G)$ the paths containing only vertices of degree 2, and at least 5 of them. To do this, we compute the connected components of the graph induced on vertices of degree 2 in $\mathcal{C}(G)$. This shows that Rule 2.5 can be carried in linear time.

**Theorem 2.5.** *The parameterized* CLOSEST 3-LEAF POWER *problem admits a kernel with $O(k^3)$ vertices. Given a graph $G$, a reduced instance can be computed in linear time.*

**Proof.** The discussion above established the time complexity to compute a kernel. Let us determine the kernel size. Let $G = (V, E)$ be a reduced graph (*i.e.* none of the reduction rules applies to $G$) which can be edited into a 3-leaf power with a set $F \subseteq V \times V$ such that $|F| \leqslant k$. Let us denote $H = G + F$ the edited graph. We first show that $\mathcal{C}(H)$ has $O(k^2)$ vertices (*i.e.* $|\mathcal{K}(H)| \in O(k^2)$), and then Lemma 1.3 enables us to conclude.

We say that a critical clique is affected if it contains an affected vertex and denote by $A$ the set of the affected critical cliques. As each edge of $F$ affects two vertices, we have that $|A| \leqslant 2k$. Since $H$ is a 3-leaf power, its critical clique graph $\mathcal{C}(H)$ is a tree. Let $T$ be the minimal subtree of $\mathcal{C}(H)$ that spans the affected critical cliques. Let us observe that if $B$ is a maximal subtree of $\mathcal{C}(H) \setminus T$, then none of the critical cliques in $B$ contains an affected vertex and thus $B$ was the critical clique graph of a 1-branch of $G$, which has been reduced by Rule 2.3 or Rule 2.4. Let $A' \subset \mathcal{K}(H)$ be the critical cliques of degree at least 3 in $T$. As $|A| \leqslant 2k$, we also have $|A'| \leqslant 2k$. The connected components resulting from the removal of $A$ and $A'$ in $T$ are paths. There are at most $4k$ such paths. Each of these paths is composed by non-affected critical cliques. It follows that each of them corresponds to $path(B)$ for some 2-branch $B$ of $G$, which has been reduced by Rule 2.5.

From these observations, we can now estimate the size of the reduced graph. Attached to each of the critical cliques of $T \setminus A$, we can have 1 pendant critical clique resulting from the application of Rule 2.3. Remark that any 2-branch reduced

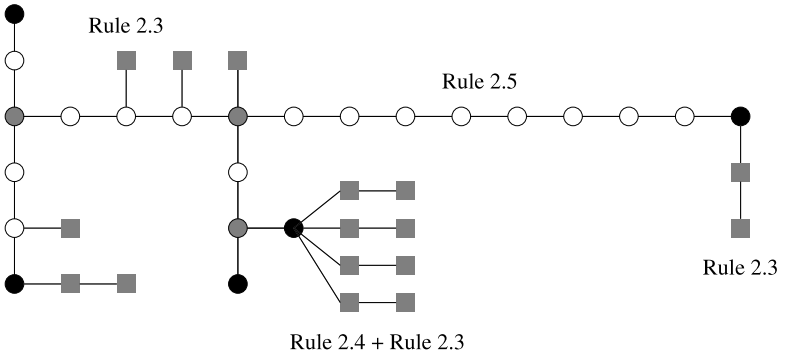


**Fig. 8.** The black circles are the critical cliques of $A$, the grey ones belong to $A'$, and the squares are the critical cliques not in $T$. On the figure, we can observe a 2-branch of size 8 reduced by Rule 2.5. There cannot be pendant critical cliques attached to its nodes. Application of Rule 2.3, may let a path of two critical cliques pendant to the elements of $A \cup A'$ and a single critical clique pendant to the elements of the small 2-branches. Finally, Rule 2.4 can only affect critical cliques of $A$.

by Rule 2.5 has no such pendant clique and that $path(B)$ contains 5 critical cliques. So, a considered 2-branch in $\mathcal{C}(H)$ is made of at most 8 critical cliques. Finally, attached to each critical clique of $A$, we can have at most $(4k + 2)$ extra critical cliques resulting from the application of Rule 2.4. See Fig. 8 for an illustration of the shape of $\mathcal{C}(H)$. Summing up everything, we obtain that $\mathcal{C}(H)$ contains at most $4k \cdot 8 + 2k \cdot 2 + 2k \cdot (4k + 3) = 8k^2 + 42k$ vertices.

By Lemma 1.3, we know that for each edited edge in a graph, the number of critical cliques increase by at most 4. It follows that $\mathcal{K}(G)$ contains at most $8k^2 + 46k$ critical cliques. By Rule 2.2, each critical clique of the reduced graph has size at most $k + 1$. This implies that the reduced graph contains at most $8k^3 + 54k^2 + 46k$ vertices, proving the $O(k^3)$ kernel size. □

We should notice that some small modifications of the branch reduction rules and a more precise analysis would improve the constants involved in the kernel size. However, the cubic bound would not change.

## 3. Kernels for edge completion and edge deletion

We now prove and adapt the previous rules to the cases where only insertions or only deletions of edges are allowed. First, observe that Rules 2.1 and 2.2 are also safe in 3-LEAF POWER COMPLETION and 3-LEAF POWER EDGE-DELETION (Rule 2.2 directly follows from Lemma 1.4). We have a similar result for the 1-branches reduction rules.

**Lemma 3.1.** *Rule 2.3 is safe for both* 3-LEAF POWER COMPLETION *and* 3-LEAF POWER DELETION.

**Proof.** In the following, we consider an optimal solution $F$ such that $H := G + F$ is a 3-leaf power, denote by $C$ the critical clique containing $P$ in $H$ and set $C' = C \setminus B^R$.

- 3-LEAF POWER COMPLETION. To show the safeness of Rule 2.3 in this case, we will build from $F$ an optimal 3-leaf power completion that respects conditions of Lemma 2.2. By Lemma 1.4, we know that the set of critical cliques $\{K_1, \ldots, K_h\}$ of $G$ whose vertices belong to $N_B(P)$ are in $C$ or adjacent to the vertices of $C$ in $H$. For $i \in \{1, \ldots, h\}$, let $C_i$ be the connected component of $B^R$ containing $K_i$. As previously, we consider $G_1$ the subgraph of $G$ induced by the disjoint union of $C_1, \ldots, C_h$ and $G'$ the subgraph of $H$ induced by $V(H) \setminus V(B^R)$. By Observation 1.8, if $C'$ is not a critical clique of $G'$, then $G'$ is a clique. Similarly, if $K_i$, for every $1 \leqslant i \leqslant h$, is not a critical clique of $G_1$, then $C_i$ is a clique. By Theorem 1.7, it follows that the graph $H' := (G', C') \otimes (G_1, \{K_1, \ldots, K_h\})$ is a 3-leaf power. By construction, the edge completion set $F'$, such that $H' = G + F'$, is a subset of $F$ and the vertices of $B$ affected by $F$ all belong to $P \cup N_B(P)$. Finally, as every $K_i$ is connected to $C'$ in $H'$, the vertices of $N_B(P)$ are all adjacent to the same vertices of $V(G) \setminus V(B^R)$.
- 3-LEAF POWER EDGE-DELETION. In the case where only edges deletion are allowed, we will build from $F$ an optimal 3-leaf power deletion respecting the conditions of Lemma 2.2 by studying the behavior of $P$ in $H$. First of all, notice that if $P$ forms a larger critical clique in $H$ with some vertex $x \in V(G) \setminus V(B^R)$, this means that $F$ contains $P \times N_B(P)$. Thus, there is no need to do extra deletions in $B^R$ and then we are done.
  Now, consider the cases where $P$ is critical in $H$ or forms a larger critical clique with some $K_i$. In both cases, we have $C' = P$. By Theorem 1.7, the graph $H' := (G', C') \otimes (G_1, \{K_1, \ldots, K_c\})$ is a 3-leaf power, and the edge set $F'$ used to transform $G$ into $H'$ is a subset of $F$ (all the edges between $C'$ and $\{K_1, \ldots, K_c\}$ are present in $G$), and then we are done. □

**Lemma 3.2.** *Rule 2.4 is safe for both* 3-LEAF POWER COMPLETION *and* 3-LEAF POWER DELETION.

**Proof.** As in Lemma 2.3, we consider $B_1, \ldots, B_l$ 1-branches of $G$, the attachment points $P_1, \ldots, P_l$ of which all have the same neighborhood $N$ and satisfy $\sum_{i=1}^{l} |P_i| > 2k + 1$. Again, as every critical clique of $G$ has at most $k + 1$ vertices, we have $l \geqslant 2$.

- 3-LEAF POWER COMPLETION. In this case, the same arguments as the ones used in the proof of Lemma 2.3 hold. We briefly detail them. First, assume that $N$ is not transformed into a clique by an optimal 3-leaf power completion $F$. To get rid

of all the $C_4$'s involving two non-adjacent vertices of $N$ and $P_i$, $P_j$, $i \neq j$, the only possibility is to transform $\cup_{i=1}^{l} P_i$ into a clique, which requires more than $k+1$ edge insertions. Thus $N$ has to be a clique. Moreover, $N$ must also become a module, otherwise we would find *darts* that would imply to transform $\cup_{i=1}^{l} P_i$ into a clique, which is impossible. Finally, $N$ must be critical (otherwise, at least one insertion for each vertex of $\cup_{i=1}^{l} P_i$ must be done), thus implying that no vertex in $\cup_{i=1}^{l} P_i$ is affected by an optimal edition.

- 3-LEAF POWER EDGE-DELETION. Firstly, observe that if $N$ is not a clique, then every optimal 3-leaf power deletion in that case would have to destroy at least $k+1$ edge disjoint $C_4$'s with edge deletion only, which is impossible. The arguments used previously hold again in this case to conclude that $N$ must become a critical clique in the modified graph. □

Now, observe that the 2-branch reduction rule can be applied directly to 3-LEAF POWER EDGE-DELETION, but will not be safe for 3-LEAF POWER COMPLETION. Indeed, in the proof of Lemma 2.4, if we look at the cycle $C$ of $G$ containing vertices of $B$, we may have to delete edges between two consecutive critical cliques along $C$ to transform $\mathcal{C}(C)$ into a tree. Nevertheless, it is possible to bound the number of vertices of $path(B)$ in the case of 3-LEAF POWER COMPLETION by looking at the edge completions required to make a cycle chordal (see Lemma 3.4).

**Lemma 3.3.** *Rule* 2.5 *is safe for* 3-LEAF POWER EDGE-DELETION.

**Proof.** Let $F$ be an arbitrary optimal 3-leaf power deletion of $G$. We call $C_1$ and $C_2$ the critical cliques of $H := G + F$ that respectively contain $P_1$ and $P_2$, and set $C_1' := C_1 \setminus B^R$ and $C_2' := C_2 \setminus B^R$. We will construct from $F$ another optimal 3-leaf power deletion $F'$ of $G$ satisfying the conditions of Lemma 2.4.

We have two cases to consider : (1) either $path(B)$ is disconnected in $H$ or (2) $path(B)$ is still connected in $H$. Case (1) works exactly as the first case studied in the proof of Lemma 2.4, and thus there exists an optimal 3-leaf power deletion on which conditions of Lemma 2.4 holds.

If case (2) holds, *i.e.* if $path(B)$ is still connected in $H$, then $P_1$ and $P_2$ must belong to distinct connected components of $H \setminus B^R$, say $X_1$ and $X_2$ (otherwise $H$ would admit a chordless cycle as induced subgraph). Furthermore, notice that we must have $C_1 = P_1$ and $C_2 = P_2$ in $H$. Indeed, if $P_1$ forms a critical clique with some vertex $x \in V(G) \setminus V(B^R)$, this means $F$ must contain $V(P_1) \times V(Q_1)$, which contradicts the hypothesis. Similarly, if $P_1$ forms a critical clique with $Q_1$, then $F$ must contain edges between $Q_1$ and $N_{B^R}(Q_1)$ which cannot be (the cases for $P_2$ are symmetric). By definition, $B^R$ is a 3-leaf power, and so are $X_1$ and $X_2$. By Theorem 1.7, it follows that the composition of these three subgraphs : $H' = (X_1, P_1) \otimes (B^R, Q_1)$ and $H'' := (H', Q_2) \otimes (X_2, P_2)$ yields a 3-leaf power. The edge set $F'$ used to obtain $H'$ from $G$ is a subset of $F$ that respects conditions of Lemma 2.4, thus implying the lemma. □

The next lemma is useful to conclude on the size of the kernel in the 3-LEAF POWER COMPLETION problem.

**Lemma 3.4.** *Let $G$ be a graph admitting a clean 2-branch $B$ such that $path(B)$ is composed by at least $k+4$ critical cliques. If $P_1$ and $P_2$ belong to the same connected component in $G \setminus B^R$, then there is no 3-leaf power completion of size at most $k$.*

**Proof.** Let $G$ be a graph with a clean 2-branch $B$ on which conditions of the Lemma 3.4 apply, and let $F$ be an optimal 3-leaf power completion of $G$. As $P_1$ and $P_2$ belong to the same connected component in $G \setminus B^R$, we have a cycle $C$ of size at least $k+4$ in $\mathcal{C}(G)$. Consider the subgraph of $\mathcal{C}(G)$ induced by the critical cliques of $C$. By Lemma 1.4 we know that there exists $F' \subseteq F$ such that $\mathcal{C}(C + F')$ is a tree. It is known that $F'$ is a triangulation of $C$ [10]. Moreover, every triangulation of a $n$-cycle needs at least $n-3$ chords, thus implying that $|F'| > k$, which is impossible. □

This result allows us to obtain a 2-branch reduction rule for the 3-LEAF POWER COMPLETION problem as well.

**Rule 3.1.** *Let $G$ be a graph having a clean 2-branch $B$ with attachment points $P_1$ and $P_2$ such that $path(B)$ is composed by at least $k+4$ critical cliques.*

- *if $P_1$ and $P_2$ belong to the same connected component in $G \setminus B^R$, then there is no completion of size at most $k$.*
- *otherwise, remove from $G$ all the vertices of $V(B^R)$ except those of $V(Q_1 \cup Q_2)$ and add all possible edges between $Q_1$ and $Q_2$.*

**Proof.** The first point follows directly from Lemma 3.4. To see the second point, we need to show that if $P_1$ and $P_2$ belong to different connected components in $G \setminus B^R$, then there exists an optimal 3-leaf power completion that affects no vertex in $V(B^R) \setminus V(Q_1 \cup Q_2)$. To show this, assume that $F$ is an optimal 3-leaf power completion of $G$ and let $C_1, C_2$ be the critical cliques containing $P_1, P_2$ in $H := G + F$. Notice that $P_1$ and $P_2$ belong to different connected components in $H \setminus B^R$, otherwise we show that, as in the proof of Lemma 3.4, $F$ has to triangulate a cycle of length at least $k+4$, thus contradicting the assumption $|F| \leqslant k$. Now, consider the subgraphs $H_1$ being the connected component of $H \setminus B^R$ containing $P_1$, and $H_2$ being the one containing $P_2$. By Theorem 1.7 and Observation 1.8, $H' := (H_1, C_1') \otimes (B^R, Q_1)$ where $C_1' := C_1 \setminus B^R$ is a 3-leaf power. With a similar argument, we can show that $H'' := (H_2, C_2') \otimes (H', Q_2)$, where $C_2' := C_2 \setminus B^R$, is a 3-leaf power. The completion used to obtain $H''$ from $G$ is a subset of $F$ respecting conditions of Rule 3.1, thus implying the result.

**Theorem 3.5.** *The parameterized* 3-LEAF POWER COMPLETION *and* 3-LEAF POWER EDGE-DELETION *problems admit kernels with $O(k^3)$ vertices. Given a graph $G$ a reduced instance can be computed in linear time.*

**Proof.** We detail separately completion and deletion.

- 3-LEAF POWER COMPLETION. As in the proof of Theorem 2.5, we consider $H := G + F$ with $G$ being reduced and $F$ being an optimal completion and we denote by $T$ the minimal subtree of $\mathcal{C}(H)$ spanning the set of affected critical cliques $A$. As noticed before, we have $|A| \leqslant 2k$.
  First, remark that the only difference between this case and 3-LEAF POWER EDITION concerns the 2-branch reduction rule. This means that the only difference will occur in the number of vertices of the paths resulting from the removal of $A$ and $A'$ in $T$ ($A'$ being critical cliques of degree at least 3 in $T$). Due to Lemma 3.4 and Rule 3.1 we know that a 2-branch in $\mathcal{C}(H)$ is made of at most $2k + 6$ critical cliques, corresponding to a path of at most $k + 4$ critical cliques, each one (except the terminal ones) having a pendant critical clique (Rule 2.3). This means that $\mathcal{C}(H)$ contains at most $4k \cdot (2k + 6) + 2k \cdot 2 + 2k \cdot (4k + 3) = 16k^2 + 34k$ critical cliques. By Lemma 1.3, we know that each edited edge creates at most 4 new critical cliques. If follows that $\mathcal{C}(G)$ contains at most $16k^2 + 38k$ vertices. By Rule 2.2, each critical clique of the reduced graph has size at most $k + 1$, thus implying that the reduced graph contains at most $16k^3 + 54k^2 + 38k$ vertices, proving the $O(k^3)$ kernel size.
- 3-LEAF POWER EDGE-DELETION. The rules used for the 3-LEAF POWER EDGE-DELETION problem are exactly the same than the one used to obtain a cubic kernel for 3-LEAF POWER EDITION. Thus, the size of a reduced instance of 3-LEAF POWER EDGE-DELETION will be exactly the same as one of a reduced instance of 3-LEAF POWER EDITION. □

## 4. Conclusion

By proving the existence of a kernel with $O(k^3)$ vertices for the 3-LEAF POWER EDITION problem, we positively answered an open problem [11,9]. The natural question is now whether the cubic bound could be improved. A strategy could be, as for the quadratic kernel of 3-HITTING SET [22] which is based on the linear kernel of VERTEX COVER [20], to consider the following subproblem:

PARAMETERIZED FAT STAR EDITION PROBLEM
**Input:** An undirected graph $G = (V, E)$.
**Parameter:** An integer $k \geqslant 0$.
**Question:** Is there a subset $F \subseteq V \times V$ with $|F| \leqslant k$ such that the graph $G + F = (V, E \triangle F)$ is a 3-leaf power and its critical clique graph $\mathcal{C}(G + F)$ is a star (we say that $G + F$ is a *fat star*).

It can be shown that small modifications of the Rules 2.1, 2.2 and 2.4 yield a kernel with $O(k^2)$ vertices for the FAT STAR EDITION problem [23]. A linear bound may be helpful to improve the kernel of the 3-LEAF POWER EDITION since it can be shown that the neighborhood of each big enough critical clique of the input graph as to be edited into a fat star.

## References


[1] J.-P. Barthélémy, A. Guénoche, Trees and proximity representations, John Wiley & Sons, 1991.
[2] S. Böcker, S. Briesemeister, G.W. Klau, Exact algorithms for cluster editing: evaluation and experiments, in: International Workshop on Experimental Algorithms, WEA, in: Lecture Notes in Computer Science, vol. 5038, 2008, pp. 289–302.
[3] H. Bodlaender, R.G. Downey, M.R. Fellows, D Hermelin, On problems without polynomial kernels (extended abstract), in: Automata, Languages and Programming, 35th International Colloquium, ICALP, in: Lecture Notes in Computer Science, vol. 5125, 2008, pp. 563–574.
[4] A. Brandstädt, V.B. Le, Structure and linear time recognition of 3-leaf powers, Information Processing Letters 98 (4) (2006) 133–138.
[5] A. Brandstädt, V.B. Le, R. Sritharan, Structure and linear time recognition of 4-leaf powers, ACM Transactions on Algorithms (TALG) 5 (1) (2008) 1–22.
[6] M.-S. Chang, M.-T. Ko, The 3-steiner root problem, in: WG, 2007, pp. 109–120.
[7] Frank K.H.A. Dehne, Michael A. Langston, Xuemei Luo, Sylvain Pitre, Peter Shaw, Yun Zhang, The cluster editing problem: Implementations and experiments, in: International Workshop on Parameterized and Exact Computation, IWPEC, in: Lecture Notes in Computer Science, vol. 4169, 2006, pp. 13–24.
[8] M. Dom, J. Guo, F. Hüffner, R. Niedermeier, Error compensation in leaf root problems, in: International Symposium on Algorithms and Computation, ISAAC, in: Lecture Notes in Computer Science, vol. 3341, 2004, pp. 389–401.
[9] M. Dom, J. Guo, F. Hüffner, R. Niedermeier, Extending the tractability border for closest leaf powers, in: International Workshop on Graph Theoretical Concepts in Computer Science WG, in: Lecture Notes in Computer Science, vol. 3787, 2005, pp. 397–408.
[10] M. Dom, J. Guo, F. Hüffner, R. Niedermeier, Error compensation in leaf root problems, Algorithmica 44 (2006) 363–381.
[11] M. Dom, J. Guo, F. Hüffner, R. Niedermeier, Closest 4-leaf power is fixed-parameter tractable, Discrete Applied Mathematics 156 (18) (2008) 3345–3361.
[12] M.R. Fellows, M. Langston, F. Rosamond, P. Shaw, Efficient parameterized preprocessing for cluster editing, in: International Symposium on Fundamentals of Computation Theory, FCT, in: Lecture Notes in Computer Science, vol. 4639, 2007, pp. 312–321.
[13] J. Gramm, J. Guo, F. Hüffner, R. Niedermeier, Graph-modeled data clustering: exact algorithms for clique generation, Theory of Computing Systems 38 (4) (2005) 373–392.
[14] J. Guo, A more effective linear kernelization for cluster editing, in: International Symposium Combinatorics, Algorithms, Probabilistic and Experimental Methodologies, ESCAPE, in: Lecture Notes in Computer Science, vol. 4614, 2007, pp. 36–47.
[15] J. Guo, R. Niedermeier, Invitation to data reduction and problem kernelization, SIGACT News 38 (1) (2007) 31–45.
[16] P. Hansen, B. Jaumard, Cluster analysis and mathematical programming, Mathematical Programming 79 (1–3) (1997) 191–215.
[17] P. Kearney, D. Corneil, Tree powers, Journal of Algorithms 29 (1) (1998) 111–131.
[18] G.H. Lin, P.E. Kearney, T. Jiang, Phylogenetic $k$-root and steiner $k$-root, in: International Symposium on Algorithms and Computation, ISAAC, in: Lecture Notes in Computer Science, vol. 1969, 2000, pp. 539–551.
[19] A. Natazon, R. Shamir, R. Sharan, Complexity classification of some edge modification problems, Discrete Applied Mathematics 113 (2001) 109–128.
[20] R. Niedermeier, Invitation to fixed parameter algorithms, in: Oxford Lectures Series in Mathematics and its Applications, vol. 31, Oxford University Press, 2006.
[21] N. Nishimura, P. Ragde, D. Thilikos, On graph powers for leaf-labeled trees, Journal of Algorithms 42 (1) (2002) 69–108.

[22] N. Nishimura, P. Ragde, D. Thilikos, Smaller kernels for hitting set problems of constant arity, in: International Workshop on Parameterized and Exact Computation, IWPEC, in: Lecture Notes in Computer Science, vol. 3162, 2004, pp. 121–126.
[23] A. Perez, Complexité paramétrique et recherche de noyaux. Master's thesis, Université de Montpellier 2, France, 2008.
[24] F. Protti, M. Dantas da Silva, J.L. Szwarcfiter, Applying modular decomposition to parameterized cluster editing problems, Theory of Computing Systems (2007).
[25] Ron Shamir, Roded Sharan, Dekel Tsur, Cluster graph modification problems, Discrete Applied Mathematics 144 (1–2) (2004) 173–182.
[26] M. Tedder, D. Corneil, M. Habib, C. Paul, Simpler linear-time modular decomposition via recursive factorizing permutations, in: Automata, Languages and Programming, ICALP, in: Lecture Notes in Computer Science, vol. 5115, 2008, pp. 634–645.

Journal of Computer and System Sciences ••• (••••) •••–•••

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

# Kernels for feedback arc set in tournaments

Stéphane Bessy [c], Fedor V. Fomin [a], Serge Gaspers [b], Christophe Paul [c], Anthony Perez [c,*], Saket Saurabh [d], Stéphan Thomassé [c]

[a] Department of Informatics, University of Bergen, N-5020 Bergen, Norway
[b] CMM, Universidad de Chile, Av. Blanco Encalada 2120, 8370459 Santiago de Chile, Chile
[c] LIRMM – Université Montpellier 2, CNRS, 161 rue Ada, 34932 Montpellier, France
[d] The Institute of Mathematical Sciences, Chennai 600 113, India

ARTICLE INFO

*Article history:*
Received 10 January 2010
Received in revised form 9 September 2010
Accepted 4 October 2010
Available online xxxx

*Keywords:*
Feedback arc set
Tournaments
Kernelization
Parameterized algorithms
Graph algorithms

ABSTRACT

A tournament $T = (V, A)$ is a directed graph in which there is exactly one arc between every pair of distinct vertices. Given a digraph on $n$ vertices and an integer parameter $k$, the Feedback Arc Set problem asks whether the given digraph has a set of $k$ arcs whose removal results in an acyclic digraph. The Feedback Arc Set problem restricted to tournaments is known as the $k$-Feedback Arc Set in Tournaments ($k$-FAST) problem. In this paper we obtain a linear vertex kernel for $k$-FAST. That is, we give a polynomial time algorithm which given an input instance $T$ to $k$-FAST obtains an equivalent instance $T'$ on $O(k)$ vertices. In fact, given any fixed $\epsilon > 0$, the kernelized instance has at most $(2+\epsilon)k$ vertices. Our result improves the previous known bound of $O(k^2)$ on the kernel size for $k$-FAST. Our kernelization algorithm solves the problem on a subclass of tournaments in polynomial time and uses a known polynomial time approximation scheme for $k$-FAST.



## 1. Introduction

Given a directed graph $G = (V, A)$ on $n$ vertices and an integer parameter $k$, the Feedback Arc Set problem asks whether the given digraph has a set of $k$ arcs whose removal results in an acyclic directed graph. In this paper, we consider this problem in a special class of directed graphs, *tournaments*. A tournament $T = (V, A)$ is a directed graph in which there is exactly one directed arc between every pair of vertices. More formally the problem we consider is defined as follows.

$k$-Feedback Arc Set in Tournaments ($k$-FAST): Given a tournament $T = (V, A)$ and a positive integer $k$, does there exist a subset $F \subseteq A$ of at most $k$ arcs whose removal makes $T$ acyclic.

In the weighted version of $k$-FAST, we are also given integer weights (each weight is at least one) on the arcs and the objective is to find a feedback arc set of weight at most $k$. This problem is called $k$-Weighted Feedback Arc Set in Tournaments ($k$-WFAST).

Feedback arc sets in tournaments are well studied from the combinatorial [20,22,29,30,33,37], statistical [31] and algorithmic [2,3,14,26,35,36] points of view. The problems $k$-FAST and $k$-WFAST have several applications. In *rank aggregation* we are given several rankings of a set of objects, and we wish to produce a single ranking that on average is as consistent as possible with the given ones, according to some chosen measure of consistency. This problem has been studied in the context of voting [8,11,13], machine learning [12], and search engine ranking [18,19]. A natural consistency measure for rank

* Corresponding author.
*E-mail address:* perez@lirmm.fr (A. Perez).

0022-0000/$ – see front matter 
doi:10.1016/j.jcss.2010.10.001

Please cite this article in press as: S. Bessy et al., Kernels for feedback arc set in tournaments, J. Comput. System Sci. (2010), doi:10.1016/j.jcss.2010.10.001

aggregation is the number of pairs that occur in a different order in the two rankings. This leads to *Kemeny rank aggregation* [24,25], a special case of $k$-WFAST.

The $k$-FAST problem is known to be NP-complete by recent results of Alon [3] and Charbit et al. [10] while $k$-WFAST is known to be NP-complete by Bartholdi III et al. [5]. From an approximation perspective, $k$-WFAST is APX-hard [32] but admits a polynomial time approximation scheme when the edge weights are bounded by a constant [26]. The problem is also well studied in parameterized complexity. In this area, a problem with input size $n$ and a parameter $k$ is said to be fixed parameter tractable (FPT) if there exists an algorithm to solve this problem in time $f(k)\cdot n^{O(1)}$, where $f$ is an arbitrary function of $k$. Raman and Saurabh [28] showed that $k$-FAST and $k$-WFAST are FPT by obtaining an algorithm running in time $O(2.415^k \cdot k^{4.752} + n^{O(1)})$. Recently, Alon et al. [4] have improved this result by giving an algorithm for $k$-WFAST running in time $O(2^{O(\sqrt{k}\log^2 k)} + n^{O(1)})$. This algorithm runs in sub-exponential time, a trait uncommon to parameterized algorithms. Moreover, a new algorithm due to Karpinsky and Schudy [23] with running time $O(2^{O(\sqrt{k})} + n^{O(1)})$ improves again the complexity of $k$-WFAST. Finally, Fomin et al. [21] provided a sub-exponential local search algorithm for $k$-WFAST. In this paper we investigate $k$-FAST from the view point of kernelization, currently one of the most active subfields of parameterized algorithms.

A parameterized problem is said to admit a *polynomial kernel* if there is a polynomial (in $n$) time algorithm, called a *kernelization* algorithm, that reduces the input instance to an instance whose size is bounded by a polynomial $p(k)$ in $k$, while preserving the answer. This reduced instance is called a $p(k)$ *kernel* for the problem. When $p(k)$ is a linear function of $k$ then the corresponding kernel is a linear kernel. Kernelization has been at the forefront of research in parameterized complexity in the last couple of years, leading to various new polynomial kernels as well as tools to show that several problems do not have a polynomial kernel under some complexity-theoretic assumptions [6,7,9,15,17,34]. In this paper we continue the current theme of research on kernelization and obtain a *linear vertex* kernel for $k$-FAST. That is, we give a polynomial time algorithm which given an input instance $T$ to $k$-FAST obtains an equivalent instance $T'$ on $O(k)$ vertices. More precisely, given any fixed $\epsilon > 0$, we find a kernel with a most $(2+\epsilon)k$ vertices in polynomial time. The reason we call it a linear *vertex* kernel is that, even though the number of vertices in the reduced instance is at most $O(k)$, the number of arcs is still $O(k^2)$. Our result improves the previous known bound of $O(k^2)$ on the vertex kernel size for $k$-FAST [4,16]. For our kernelization algorithm we find a subclass of tournaments where one can find a minimum sized feedback arc set in polynomial time (see Lemma 3.8) and use the known polynomial time approximation scheme for $k$-FAST by Kenyon-Mathieu and Schudy [26]. The polynomial time algorithm for a subclass of tournaments could be of independent interest.

The paper is organized as follows. In Section 2, we give some definition and preliminary results regarding feedback arc sets. In Section 3 we give a linear vertex kernel for $k$-FAST. Finally we conclude with some remarks in Section 4.

## 2. Preliminaries

Let $T = (V, A)$ be a tournament on $n$ vertices. We use $T_\sigma = (V_\sigma, A)$ to denote a tournament whose vertices are ordered under a fixed ordering $\sigma = v_1, \ldots, v_n$ (we also use $D_\sigma$ for an ordered directed graph). We say that an arc $v_i v_j$ of $T_\sigma$ is a *backward arc* if $i > j$, otherwise we call it a *forward arc*. Moreover, given any partition $\mathcal{P} := \{V_1, \ldots, V_l\}$ of $V_\sigma$, where every $V_i$ is an interval according to the ordering of $T_\sigma$, we use $A_B$ to denote all arcs between the intervals (having their endpoints in different intervals), and $A_I$ for all arcs within the intervals. If $T_\sigma$ contains no backward arc, then we say that it is *transitive*.

For a vertex $v \in V$ we denote its *in-neighborhood* by $N^-(v) := \{u \in V \mid uv \in A\}$ and its *out-neighborhood* by $N^+(v) := \{u \in V \mid vu \in A\}$. A set of vertices $M \subseteq V$ is a *module* if and only if $N^+(u) \setminus M = N^+(v) \setminus M$ for every $u, v \in M$. For a subset of arcs $A' \subseteq A$, we define $T[A']$ to be the digraph $(V', A')$ where $V'$ is the union of endpoints of the arcs in $A'$. Given an ordered digraph $D_\sigma$ and an arc $e = v_i v_j$, $S(e) = \{v_i, \ldots, v_j\}$ denotes the *span* of $e$. The number of vertices in $S(e)$ is called the *length* of $e$ and is denoted by $l(e)$. Thus, for every arc $e = v_i v_j$, $l(e) = |i - j| + 1$. Finally, for every vertex $v$ in the span of $e$, we say that $e$ is *above* $v$.

In this paper, we will use the well-known fact that every acyclic tournament admits a transitive ordering. In particular, we will consider *maximal transitive modules*. We also need the following result for our kernelization algorithm.

**Lemma 2.1.** *(See [28].) Let $D = (V, A)$ be a directed graph and $F$ be a minimal feedback arc set of $D$. Let $D'$ be the graph obtained from $D$ by reversing the arcs of $F$ in $D$, then $D'$ is acyclic.*

We now introduce a definition which is useful for a lemma we prove later.

**Definition 2.2.** Let $D_\sigma = (V_\sigma, A)$ be an ordered directed graph and let $f = vu$ be a backward arc of $D_\sigma$. We call *certificate* of $f$, and denote it by $c(f)$, any directed path from $u$ to $v$ using only forward arcs in the span of $f$ in $D_\sigma$.

Observe that such a directed path together with the backward arc $f$ forms a directed cycle in $D_\sigma$ whose only backward arc is $f$.

**Definition 2.3.** Let $D_\sigma = (V_\sigma, A)$ be an ordered directed graph, and let $F \subseteq A$ be a set of backward arcs of $D_\sigma$. We say that we can *certify* $F$ whenever it is possible to find a set $\mathcal{F} = \{c(f):\ f \in F\}$ of arc-disjoint certificates for the arcs in $F$.

Please cite this article in press as: S. Bessy et al., Kernels for feedback arc set in tournaments, J. Comput. System Sci. (2010), doi:10.1016/j.jcss.2010.10.001

Let $D_\sigma = (V_\sigma, A)$ be an ordered directed graph, and let $F \subseteq A$ be a subset of backward arcs of $D_\sigma$. We say that we can certify the set $F$ using *only arcs from* $A' \subseteq A$ if $F$ can be certified by a collection $\mathcal{F}$ such that the union of the arcs of the certificates in $\mathcal{F}$ is contained in $A'$. In the following, $fas(D)$ denotes the *size* of a minimum feedback arc set, that is, the cardinality of a minimum sized set $F$ of arcs whose removal makes $D$ acyclic.

**Lemma 2.4.** *Let $D_\sigma$ be an ordered directed graph, and let $\mathcal{P} = \{V_1, \ldots, V_l\}$ be a partition of $D_\sigma$ into intervals. Assume that the set $F$ of all backward arcs of $D_\sigma[A_B]$ can be certified using only arcs from $A_B$. Then $fas(D_\sigma) = fas(D_\sigma[A_I]) + fas(D_\sigma[A_B])$. Moreover, there exists a minimum sized feedback arc set of $D_\sigma$ containing $F$.*

**Proof.** For any bipartition of the arc set $A$ into $A_1$ and $A_2$, $fas(D_\sigma) \geqslant fas(D_\sigma[A_1]) + fas(D_\sigma[A_2])$. Hence, in particular for a partition of the arc set $A$ into $A_I$ and $A_B$ we have that $fas(D_\sigma) \geqslant fas(D_\sigma[A_I]) + fas(D_\sigma[A_B])$. Next, we show that $fas(D_\sigma) \leqslant fas(D_\sigma[A_I]) + fas(D_\sigma[A_B])$. This follows from the fact that once we reverse all the arcs in $F$, each remaining directed cycle lies in $D_\sigma[V_i]$ for some $i \in \{1, \ldots, l\}$. In other words once we reverse all the arcs in $F$, every cycle is completely contained in $D_\sigma[A_I]$. This concludes the proof of the first part of the lemma. In fact, what we have shown is that there exists a minimum sized feedback arc set of $D_\sigma$ containing $F$. This concludes the proof of the lemma. □

## 3. Kernels for *k*-FAST

In this section we first give a subquadratic vertex kernel of size $O(k\sqrt{k})$ for $k$-FAST and then improve on it to get our final vertex kernel of size $O(k)$. We start by giving a few reduction rules that will be needed to bound the size of the kernels.

**Rule 3.1.** If a vertex $v$ is not contained in any triangle, delete $v$ from $T$.

**Rule 3.2.** If there exists an arc $uv$ that belongs to more than $k$ distinct triangles, then reverse $uv$ and decrease $k$ by 1.

We say that a reduction rule is *sound*, if whenever the rule is applied to an instance $(T, k)$ to obtain an instance $(T', k')$, $T$ has a feedback arc set of size at most $k$ if and only if $T'$ has a feedback arc set of size at most $k'$. Moreover, *applying* a reduction rule in polynomial time means that the structure sought by the reduction rule can be identified in polynomial time and the instance can be updated in polynomial time. Finally, we say that an instance $(T, k)$ is *reduced* according to a set of reduction rules whenever none of the reduction rules can be applied to $(T, k)$.

**Lemma 3.1.** *(See [4,16].) Rules* 3.1 *and* 3.2 *are sound and can be applied in polynomial time.*

Rules 3.1 and 3.2 together led to a quadratic kernel for $k$-WFAST [4]. Earlier, these rules were used by Dom et al. [16] to obtain a quadratic kernel for $k$-FAST. We now add a new reduction rule that will allow us to obtain the claimed bound on the kernel sizes for $k$-FAST. Given an ordered tournament $T_\sigma = (V_\sigma, A)$, we say that $\mathcal{P} = \{V_1, \ldots, V_l\}$ is a *safe partition* of $V_\sigma$ into intervals whenever it is possible to certify the backward arcs of $T_\sigma[A_B]$ using only arcs from $A_B$.

**Rule 3.3.** Let $T_\sigma$ be an ordered tournament, and $\mathcal{P} = \{V_1, \ldots, V_l\}$ be a safe partition of $V_\sigma$ into intervals such that $F \neq \varnothing$, where $F$ denotes the set of backward arcs of $T_\sigma[A_B]$. Then reverse all the arcs of $F$ and decrease $k$ by $|F|$.

**Lemma 3.2.** *Rule* 3.3 *is sound.*

**Proof.** Let $\mathcal{P}$ be a safe partition of $T_\sigma$. Observe that it is possible to certify all the backward arcs, that is $F$, using only arcs in $A_B$. Hence using Lemma 2.4 we have that $fas(T_\sigma) = fas(T_\sigma[A_I]) + fas(T_\sigma[A_B])$. Furthermore, by Lemma 2.4 we also know that there exists a minimum sized feedback arc set of $D_\sigma$ containing $F$. Thus, $T_\sigma$ has a feedback arc set of size at most $k$ if and only if the tournament $T'_\sigma$ obtained from $T_\sigma$ by reversing all the arcs of $F$ has a feedback arc set of size at most $k - |F|$. □

### 3.1. A subquadratic kernel for *k*-FAST

In this section, we show how to obtain an $O(k\sqrt{k})$ sized vertex kernel for $k$-FAST. To do so, we introduce the following reduction rule (see Fig. 1).

**Rule 3.4.** Let $T_m$ be a *maximal transitive module* of size $p$, and $I$ and $O$ be the set of in-neighbors and out-neighbors of the vertices of $T_m$ in $T$, respectively. Let $Z$ be the set of arcs $uv$ such that $u \in O$ and $v \in I$. If $q = |Z| < p$ then reverse all the arcs in $Z$ and decrease $k$ by $q$.

Please cite this article in press as: S. Bessy et al., Kernels for feedback arc set in tournaments, J. Comput. System Sci. (2010), doi:10.1016/j.jcss.2010.10.001

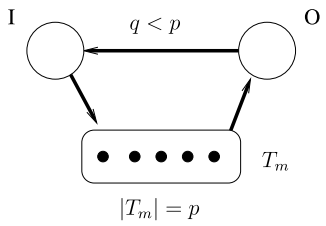


**Fig. 1.** A transitive module on which Rule 3.4 applies.

**Lemma 3.3.** *Rule* 3.4 *is sound and can be applied in* $O(n+m)$ *time.*

**Proof.** We first prove that the partition $\mathcal{P} = \{I, T_m, O\}$ forms a safe partition of the input tournament. Let $T'_m = \{w_1, \ldots, w_q\} \subseteq T_m$ be an arbitrary subset of size $q$ of $T_m$ and let $Z = \{u_i v_i \mid 1 \leqslant i \leqslant q\}$. Consider the collection $\mathcal{F} = \{v_i w_i u_i \mid u_i v_i \in Z,\ w_i \in T'_m\}$ and notice that it certifies all the arcs in $Z$. In fact we have managed to certify all the backwards arcs of the partition using only arcs from $A_B$ and hence $\mathcal{P}$ forms a safe partition. Thus, by Rule 3.3, it is safe to reverse all the arcs from $O$ to $I$.

The time complexity follows from the fact that computing the modular decomposition tree can be done in $O(n+m)$ time on directed graphs [27]. It is well known that the modular decomposition tree of a tournament has nodes labelled either *prime* or *transitive* and that each maximal transitive module corresponds to the set of leaves attached to some transitive node of the modular decomposition tree. □

We show that any YES-instance to which none of Rules 3.1, 3.2 and 3.4 could be applied has at most $O(k\sqrt{k})$ vertices.

**Theorem 3.4.** *Let* $(T = (V, A), k)$ *be a* YES*-instance to* $k$-FAST *which has been reduced according to Rules* 3.1, 3.2 *and* 3.4. *Then* $T$ *has at most* $O(k\sqrt{k})$ *vertices.*

**Proof.** Let $S$ be a feedback arc set of size at most $k$ of $T$ and let $T'$ be the tournament obtained from $T$ by reversing all the arcs in $S$. Let $\sigma$ be the transitive ordering of $T'$ and $T_\sigma = (V_\sigma, A)$ be the ordered tournament corresponding to the ordering $\sigma$. We say that a vertex is *affected* if it is incident to some arc in $S$. Thus, the number of affected vertices is at most $2|S| \leqslant 2k$. The reduction Rule 3.1 ensures that the first and last vertex of $T_\sigma$ are affected. To see this note that if the first vertex in $V_\sigma$ is not affected then it is a source vertex (vertex with in-degree 0) and hence it is not part of any triangle and thus Rule 3.1 would have applied. We can similarly argue for the last vertex. Next we argue that there is no backward arc $e$ of length greater than $2k+2$ in $T_\sigma$. Assume to the contrary that $e = uv$ is a backward arc with $S(e) = \{v, x_1, x_2, \ldots, x_{2k+1}, \ldots, u\}$ and hence $l(e) > 2k+2$. Consider the collection $\mathcal{T} = \{vx_iu \mid 1 \leqslant i \leqslant 2k\}$ and observe that at most $k$ of these triples can contain an arc from $S \setminus \{e\}$ and hence there exist at least $k+1$ triplets in $\mathcal{T}$ which corresponds to distinct triangles all containing $e$. But then $e$ would have been reversed by an application of Rule 3.2. Hence, we have shown that there is no backward arc $e$ of length greater than $2k+2$ in $T_\sigma$. Thus $\sum_{e \in S} l(e) \leqslant 2k^2 + 2k$.

We also know that between two consecutive affected vertices there is exactly one maximal transitive module. Let us denote by $t_i$ the number of vertices in these modules, where $i \in \{1, \ldots, 2k-1\}$. The objective here is to bound the number of vertices in $V_\sigma$ or $V$ using $\sum_{i=1}^{2k-1} t_i$. To do so, observe that since $T$ is reduced under Rule 3.4, there are at least $t_i$ backward arcs above every module with $t_i$ vertices, each of length at least $t_i$. This implies that $\sum_{i=1}^{2k-1} t_i^2 \leqslant \sum_{e \in S} l(e) \leqslant 2k^2 + 2k$. Now, using the Cauchy–Schwarz inequality we can show the following:

$$\sum_{i=1}^{2k-1} t_i = \sum_{i=1}^{2k-1} t_i \cdot 1 \leqslant \sqrt{\sum_{i=1}^{2k-1} t_i^2 \cdot \sum_{i=1}^{2k-1} 1} \leqslant \sqrt{(2k^2+2k)\cdot(2k-1)} = \sqrt{4k^3+2k^2-2k}.$$

Thus every reduced YES-instance has at most $\sqrt{4k^3+2k^2-2k} + 2k = O(k\sqrt{k})$ vertices. □

### 3.2. A linear kernel for $k$-FAST

We begin this subsection by showing some general properties about tournaments which will be useful in obtaining a linear kernel for $k$-FAST.

#### 3.2.1. Backward weighted tournaments

Let $T_\sigma$ be an ordered tournament with weights on its backward arcs. We call such a tournament a *backward weighted tournament* and denote it by $T_\omega$, and use $\omega(e)$ to denote the weight of a backward arc $e$. For every interval $I := [v_i, \ldots, v_j]$ we use $\omega(I)$ to denote the total weight of all backward arcs having both their endpoints in $I$, that is, $\omega(I) = \sum_{e=uv} w(e)$ where $u, v \in I$ and $e$ is a backward arc.

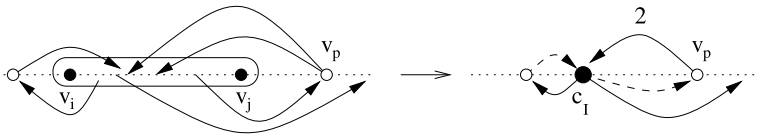


**Fig. 2.** Illustration of the contraction step for the interval $I := [v_i, \ldots, v_j]$.

**Definition 3.5** *(Contraction).* Let $T_\omega = (V_\sigma, A)$ be an ordered tournament with weights on its backward arcs and $I = [v_i, \ldots, v_j]$ be an interval. The contracted tournament is defined as $T_{\omega'} = (V_{\sigma'} = V_\sigma \setminus \{I\} \cup \{c_I\}, A')$. The arc set $A'$ is defined as follows.

- It contains all the arcs $A_1 = \{uv \mid uv \in A, u \notin I, v \notin I\}$.
- Add $A_2 = \{uc_I \mid uv \in A, u \notin I, v \in I\}$ and $A_3 = \{c_I v \mid uv \in A, u \in I, v \notin I\}$.
- Finally, we remove every forward arc involved in a 2-cycle after the addition of arcs in the previous step.

The order $\sigma'$ for $T_{\omega'}$ is provided by $\sigma' = v_1, \ldots, v_{i-1}, c_I, v_{j+1}, \ldots, v_n$. We define the weight of a backward arc $e = xy$ of $A'$ as follows:

$$w'(xy) = \begin{cases} w(xy) & \text{if } xy \in A_1, \\ \sum_{\{xz \in A \mid z \in I\}} w(xz) & \text{if } xy \in A_2, \\ \sum_{\{zy \in A \mid z \in I\}} w(zy) & \text{if } xy \in A_3. \end{cases}$$

We refer to Fig. 2 for an illustration.

Next we generalize the notions of certificate and certification (Definitions 2.2 and 2.3) to backward weighted tournaments.

**Definition 3.6.** Let $T_\omega = (V_\sigma, A)$ be a backward weighted tournament, and let $f = vu \in A$ be a backward arc of $T_\omega$. We call *$\omega$-certificate* of $f$, and denote it by $\mathcal{C}(f)$, a collection of $\omega(f)$ arc-disjoint directed paths going from $u$ to $v$ and using only forward arcs in the span of $f$ in $T_\omega$.

**Definition 3.7.** Let $T_\omega = (V_\sigma, A)$ be a backward weighted tournament, and let $F \subseteq A$ be a subset of backward arcs of $T_\omega$. We say that we can *$\omega$-certify* $F$ whenever it is possible to find a set $\mathcal{F} = \{\mathcal{C}(f) \colon f \in F\}$ of arc-disjoint $\omega$-certificates for the arcs in $F$.

**Lemma 3.8.** *Let $T_\omega = (V_\sigma, A)$ be a backward weighted tournament such that for every interval $I := [v_i, \ldots, v_j]$ the following holds*:

$$2 \cdot \omega(I) \leqslant |I| - 1. \tag{1}$$

*Then it is possible to $\omega$-certify the backward arcs of $T_\omega$.*

**Proof.** Let $V_\sigma = v_1, \ldots, v_n$. The proof is by induction on $n$, the number of vertices. Note that by applying (1) to the interval $I = [v_1, \ldots, v_n]$, we have that there exists a vertex $v_i$ in $T_\omega$ that is not incident to any backward arc. Let $T'_\omega = (V'_\sigma, A')$ denote the tournament $T_\omega \setminus \{v_i\}$. We say that an interval $I$ is *critical* whenever $|I| \geqslant 2$ and $2 \cdot \omega(I) = |I| - 1$. We now consider several cases, based on different types of critical intervals.

(i) Suppose that there are no critical intervals. Thus, in $T'_\omega$, every interval satisfies (1), and hence by induction on $n$ the result holds.

(ii) Suppose now that the only critical interval is $I = [v_1, \ldots, v_n]$, and let $e = vu$ be a backward arc above $v_i$ with the maximum length. Note that since $v_i$ does not belong to any backward arc, we can use it to form a directed path $c(e) = uv_iv$, which is a certificate for $e$. We now consider $T'_\omega$ where the weight of $e$ has been decreased by 1. In this process if $\omega(e)$ becomes 0 then we reverse the arc $e$. We now show that every interval of $T'_\omega$ respects (1). If an interval $I' \in T'_\omega$ does not contain $v_i$ in the corresponding interval in $T_\omega$, then by our assumption we have that $2 \cdot \omega(I') \leqslant |I'| - 1$. Now we assume that the interval corresponding to $I'$ in $T_\omega$ contains $v_i$ but either $u \notin I' \cup \{v_i\}$ or $v \notin I' \cup \{v_i\}$. Then we have $2 \cdot \omega(I') = 2 \cdot \omega(I) < |I| - 1 = |I'|$ and hence we get that $2 \cdot \omega(I') \leqslant |I'| - 1$. Finally, we assume that the interval corresponding to $I'$ in $T_\omega$ contains $v_i$ and $u, v \in I' \cup \{v_i\}$. In this case, $2 \cdot \omega(I') = 2 \cdot (\omega(I) - 1) \leqslant |I| - 1 - 2 < |I'| - 1$. Thus, by the induction hypothesis, we obtain a family of arc-disjoint $\omega$-certificates $\mathcal{F}'$ which $\omega$-certify the backward arcs of $T'_\omega$. Observe that the maximality of $l(e)$ ensures that if $e$ is reversed then it will not be used in any $\omega$-certificate of $\mathcal{F}'$, thus implying that $\mathcal{F}' \cup c(e)$ is a family $\omega$-certifying the backward arcs of $T_\omega$.

(iii) Finally, suppose that there exists a critical interval $I \subsetneq V_\sigma$. Roughly speaking, we will show that $I$ and $V_\sigma \setminus I$ can be certified separately. To do so, we first show the following.

**Claim 1.** *Let $I \subset V_\sigma$ be a critical interval. Then the tournament $T_{\omega'} = (V_{\sigma'}, A')$ obtained from $T_\omega$ by contracting $I$ satisfies the conditions of the lemma.*

**Proof.** Let $H'$ be any interval of $T_{\omega'}$. As before if $H'$ does not contain $c_I$ then the result holds by hypothesis. Otherwise, let $H$ be the interval corresponding to $H'$ in $T_\omega$. We will show that $2\omega(H') \leqslant |H'| - 1$. By hypothesis, we know that $2\omega(H) \leqslant |H| - 1$ and that $2\omega(I) = |I| - 1$. Thus we have the following.

$$2\omega(H') = 2 \cdot \big(\omega(H) - \omega(I)\big) \leqslant |H| - 1 - |I| + 1 = \big(|H| + 1 - |I|\big) - 1 = |H'| - 1.$$

Thus, we have shown that the tournament $T_{\omega'}$ satisfies the conditions of the lemma. □

We now consider a minimal critical interval $I$. By induction, and using the claim, we know that we can obtain a family of arc-disjoint $\omega$-certificates $\mathcal{F}'$ which $\omega$-certifies the backward arcs of $T_{\omega'}$ without using any arc within $I$. Now, by minimality of $I$, we can use (ii) to obtain a family of arc-disjoint $\omega$-certificates $\mathcal{F}''$ which $\omega$-certifies the backward arcs of $I$ using only arcs within $I$. Thus, $\mathcal{F}' \cup \mathcal{F}''$ is a family $\omega$-certifying all backward arcs of $T_\omega$.

This concludes the proof of the lemma. □

In the following, any interval that does not respect condition (1) is said to be a *dense interval*.

**Lemma 3.9.** *Let $T_\omega = (V_\sigma, A)$ be a backward weighted tournament reduced under Rule* 3.1 *with $|V_\sigma| \geqslant 2p + 1$ and $\omega(V_\sigma) \leqslant p$. Then there exists a safe partition of $V_\sigma$ with at least one backward arc between the intervals and it can be computed in polynomial time.*

**Proof.** The proof is by induction on $n = |V_\sigma|$. Observe that by hypothesis every vertex of $T_\omega$ belongs to the span of some backward arc (since otherwise it would not be contained in any triangle). It follows that the statement is true for $n = 3$: in such a case, $T_\omega$ is a triangle with exactly one backward arc, and hence the partition into singletons is a safe partition. This constitutes our base case.

For the inductive step, we assume first that there is no dense interval in $T_\omega$. In this case Lemma 3.8 ensures that the partition of $V_\sigma$ into singletons of vertices is a safe partition. So from now on we assume that there exists at least one dense interval.

Let $I$ be a dense interval. By definition of $I$, we have that $\omega(I) \geqslant \frac{1}{2} \cdot |I|$. We now contract $I$ and obtain the backward weighted tournament $T_{\omega'} = (V_{\sigma'}, A')$. In the contracted tournament $T_{\omega'}$, we have:

$$\begin{cases} |V_{\sigma'}| \geqslant 2p + 1 - \big(|I| - 1\big) = 2p - |I| + 2; \\ \omega'(V_{\sigma'}) \leqslant p - \dfrac{1}{2} \cdot |I|. \end{cases}$$

Thus, if we set $r := p - \frac{1}{2} \cdot |I|$, we get that $|V_{\sigma'}| \geqslant 2r + 1$ and $\omega'(V_{\sigma'}) \leqslant r$. Since $|V_{\sigma'}| < |V_\sigma|$, by the induction hypothesis we can find a safe partition $\mathcal{P}$ of $T_{\omega'}$, and thus obtain a family $\mathcal{F}_{\omega'}$ that $\omega$-certifies the backward arcs of $T_{\omega'}[A_B]$ using only arcs in $A_B$. Observe that every vertex of $T_{\omega'}$ still belongs to the span of some backward arc, and hence it is safe to apply our induction hypothesis.

We claim that $\mathcal{P}'$ obtained from $\mathcal{P}$ by substituting $c_I$ by its corresponding interval $I$ is a safe partition in $T_\omega$. To see this, first observe that if $c_I$ has not been used to $\omega$-certify the backward arcs in $T_{\omega'}[A_B]$, that is, if $c_I$ is not an end point of any arc in the $\omega$-certificates, then we are done. So from now on we assume that $c_I$ has been part of an $\omega$-certificate for some backward arc. Let $e = vu$ be such a backward arc in $T_{\omega'}[A_B]$, and let $c_{\omega'}(e) \in \mathcal{F}_{\omega'}$ be a $\omega$-certificate of $e$. First we assume that $c_I$ is neither the first nor the last vertex of the certificate $c_{\omega'}(e)$ (with respect to ordering $\sigma'$), and let $c_1$ and $c_2$ be the left (in-) and right (out-) neighbors of $c_I$ in $c_{\omega'}(e)$. By definition of the contraction step together with the fact that there is a forward arc between $c_1$ and $c_I$ and between $c_I$ and $c_2$ in $T_{\omega'}$, we have that there were no backward arcs between any vertex in the interval corresponding to $c_I$ and $c_1$ and $c_2$ in the original tournament $T_\omega$. So we can always find a vertex in $I$ to replace $c_I$ in $c_{\omega'}(e)$, thus obtaining a certificate $c(e)$ for $e$ in $T_\omega[A_B]$ (observe that $e$ remains a backward arc even in $T_\omega$). Now we assume that $c_I$ is either the first or last vertex in the certificate $c_{\omega'}(e)$. Let $e'$ be an arc in $T_\omega$ corresponding to $e$ in $T_{\omega'}$ with one of its endpoints being $e_I \in I$. To certify $e'$ in $T_\omega[A_B]$, we need to show that we can construct a certificate $c(e')$ using only arcs of $T_\omega[A_B]$. We have two cases to deal with.

(i) If $c_I$ is the first vertex of $c_{\omega'}(e)$, then let $c_1$ be the right neighbor of $c_I$ for any directed path $P$ between $u = c_I$ and $v$ in $c_{\omega'}(e)$. Using the same argument as before, there are only forward arcs between any vertex in $I$ and $c_1$. In particular, there is a forward arc $e_I c_1$ in $T_\omega$, meaning that we can construct a $\omega$-certificate for $e'$ in $T_\omega$ by setting $c(e') := (c_{\omega'}(e) \setminus \{c_I\}) \cup \{e_I\}$. (See Fig. 3.)

(ii) If $c_I$ is the last vertex of $c_{\omega'}(e)$, then let $c_q$ be the left neighbor of $c_I$ for any directed path $P$ between $u$ and $v = c_I$ in $c_{\omega'}(e)$. Once again, we have that there are only forward arcs between $c_q$ and vertices in $I$, and thus between $c_q$ and $e_I$. So using this we can construct a $\omega$-certificate for $e'$ in $T_\omega$.

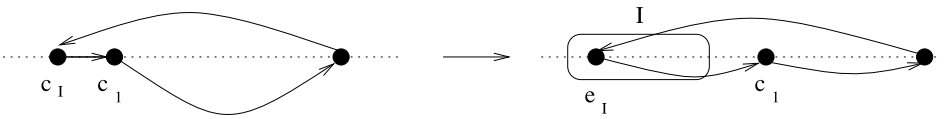


**Fig. 3.** On the left, the $\omega$-certificate $c_{\omega'}(e) \in \mathcal{F}_{\omega'}$. On the right, the corresponding $\omega$-certificate obtained in $T_\omega$ by replacing $c_I$ by the interval $I$.

**Algorithm 1**: Kernelization algorithm for the $(2+\epsilon)k$-vertex kernel of $k$-FAST.

**Input**: An instance $T = ((V, A), k)$ of $k$-FAST, where $k \in \mathbb{N}$, and a fixed $\epsilon > 0$.
**Output**: An equivalent instance $T' = ((V', A'), k')$ with $|V'| \leqslant (2+\epsilon)k$.

```
1  Reduce (T, k) according to Rule 3.1;
2  Compute a feedback arc set S using the (1 + ε/2)-PTAS for k-FAST [26];
3  if (|S| > (1 + ε/2)k) then
4      Return a small trivial No-instance;
5  else
6      T_σ ← the transitive ordering obtained by reversing every arc of S in T (observe that ω(T_σ) ⩽ p := (1 + ε/2)k);
7      repeat
8          if |V_σ| ⩾ 2p + 1 > (2 + ε)k then
9              P ← the safe partition (with at least one backward arc between its intervals) obtained by Lemma 3.9;
10             T_σ ← reverse all backward arcs between intervals of P in T_σ (Rule 3.3);
11             k ← decrease the value of k accordingly;
12             if (k ⩾ 0) then
13                 Reduce (T_σ, k) according to Rule 3.1;
14                 if (V_σ = ∅) then
15                     Return a small trivial YES-instance;
16     until (k ⩽ 0) or (|V_σ| ⩽ (2 + ε)k);
17     if (k ⩽ 0) then
18         Return a small trivial No-instance;
19     else
20         Return T_σ;
```

Notice that the fact that all $\omega$-certificates are pairwise arc-disjoint in $T_{\omega'}[A_B]$ implies that the corresponding $\omega$-certificates are arc-disjoint in $T_\omega[A_B]$, and so $\mathcal{P}'$ is indeed a safe partition of $V_\sigma$. □

We are now ready to give the linear size kernel for $k$-FAST. To do so, we make use of the fact that there exists a polynomial time approximation scheme for this problem [26]. The kernelization algorithm is depicted in Algorithm 1.

**Theorem 3.10.** *For every fixed $\epsilon > 0$, there exists a vertex kernel for $k$-FAST with at most $(2+\epsilon)k$ vertices that can be computed in polynomial time.*

**Proof.** Let $(T = (V, A), k)$ be an instance of $k$-FAST. First, we reduce $(T, k)$ according to Rule 3.1. Then, for a fixed $\epsilon > 0$, we compute a feedback arc set $S$ using the known $(1 + \frac{\epsilon}{2})$-polynomial time approximation scheme for $k$-FAST [26]. If $|S| > (1 + \frac{\epsilon}{2})k$, then there is no feedback arc set of size at most $k$ for $T$. Hence we return a trivial small No-instance. Otherwise, $S$ has size at most $(1 + \frac{\epsilon}{2})k$. We then order $T$ with the transitive ordering of the tournament obtained by reversing every arc of $S$ in $T$. Let $T_\sigma$ denote the resulting ordered tournament. By the upper bound on the size of $S$, we know that $T_\sigma$ has at most $(1+\frac{\epsilon}{2})k$ backward arcs. Thus, if $T_\sigma$ has more than $(2+\epsilon)k$ vertices then Lemma 3.9 ensures that we can find a safe partition with at least one backward arc between the intervals in polynomial time. Hence we can reduce the tournament by applying Rule 3.3, decreasing the value of $k$ accordingly. Finally, if $k \geqslant 0$, we reduce the tournament according to Rule 3.1; notice that if we get $V = \varnothing$ doing so, then we return a small trivial YES-instance. We repeat the previous steps until we do not find a safe partition or $k \leqslant 0$. In the former case, we know by Lemma 3.9 that $T$ can have at most $(2+\epsilon)k$ vertices, thus implying the result. In the latter case we return a trivial small No-instance. □

## 4. Conclusion

In this paper we obtained linear vertex kernel for $k$-FAST, in fact, a vertex kernel of size $(2+\epsilon)k$ for any fixed $\epsilon > 0$. The new bound on the kernel size improves the previous known bound of $O(k^2)$ on the vertex kernel size for $k$-FAST given in [4,16]. Moreover, it would be interesting to see if one can obtain kernels for other problems using either polynomial time approximation schemes or a constant factor approximation algorithm for the corresponding problem. An interesting problem which remains unanswered is, whether there exists a linear or even an $o(k^2)$ vertex kernel for the $k$-FEEDBACK VERTEX SET IN TOURNAMENTS ($k$-FVST) problem. In the $k$-FVST problem we are given a tournament $T$ and a positive integer $k$

Please cite this article in press as: S. Bessy et al., Kernels for feedback arc set in tournaments, J. Comput. System Sci. (2010), doi:10.1016/j.jcss.2010.10.001

and the aim is to find a set of at most $k$ vertices whose deletion makes the input tournament acyclic. The smallest known kernel for $k$-FVST has size $O(k^2)$ [1].

## References

[1] F.N. Abu-Khzam, A kernelization algorithm for d-hitting set, J. Comput. System Sci. 76 (7) (2010) 524–531.
[2] N. Ailon, M. Charikar, A. Newman, Aggregating inconsistent information: ranking and clustering, in: ACM Symposium on Theory of Computing (STOC), 2005, pp. 684–693.
[3] N. Alon, Ranking tournaments, SIAM J. Discrete Math. 20 (1) (2006) 137–142.
[4] N. Alon, D. Lokshtanov, S. Saurabh, Fast FAST, in: ICALP, in: Lecture Notes in Comput. Sci., vol. 5555, Springer, 2009, pp. 49–58.
[5] J. Bartholdi III, C.A. Tovey, M.A. Trick, Voting schemes for which it can be difficult to tell who won the election, Soc. Choice Welf. 6 (2) (1989) 157–165.
[6] H.L. Bodlaender, R.G. Downey, M.R. Fellows, D. Hermelin, On problems without polynomial kernels, J. Comput. System Sci. 75 (8) (2009) 423–434.
[7] H.L. Bodlaender, F.V. Fomin, D. Lokshtanov, E. Penninkx, S. Saurabh, D.M. Thilikos, (Meta) Kernelization, in: FOCS, 2009, pp. 629–638.
[8] J. Borda, Mémoire sur les élections au scrutin, Histoire de l'Académie Royale des Sciences, 1781.
[9] N. Bousquet, J. Daligault, S. Thomassé, A. Yeo, A polynomial kernel for multicut in trees, in: Symposium on Theoretical Aspects of Computer Science (STACS), 2009, pp. 183–194.
[10] P. Charbit, S. Thomassé, A. Yeo, The minimum feedback arc set problem is NP-hard for tournaments, Combin. Probab. Comput. 16 (1) (2007) 1–4.
[11] I. Charon, O. Hudry, A survey on the linear ordering problem for weighted or unweighted tournaments, 4OR 5 (1) (2007) 5–60.
[12] W.W. Cohen, R.E. Schapire, Y. Singer, Learning to order things, in: Advances in Neural Information Processing Systems (NIPS), 1997, pp. 451–457.
[13] M. Condorcet, Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix, 1785.
[14] D. Coppersmith, L. Fleischer, A. Rudra, Ordering by weighted number of wins gives a good ranking for weighted tournaments, in: ACM–SIAM Symposium on Discrete Algorithms (SODA), 2006, pp. 776–782.
[15] H. Dell, D. van Melkebeek, Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses, in: STOC, ACM, 2010, pp. 251–260.
[16] M. Dom, J. Guo, F. Hüffner, R. Niedermeier, A. Truß, Fixed-parameter tractability results for feedback set problems in tournaments, in: Conference on Algorithms and Complexity (CIAC), in: Lecture Notes in Comput. Sci., vol. 3998, 2006, pp. 320–331.
[17] M. Dom, D. Lokshtanov, S. Saurabh, Incompressibility through colors and IDs, in: ICALP, in: Lecture Notes in Comput. Sci., vol. 5555, Springer, 2009, pp. 378–389.
[18] C. Dwork, R. Kumar, M. Naor, D. Sivakumar, Rank aggregation revisited, Technical report.
[19] C. Dwork, R. Kumar, M. Naor, D. Sivakumar, Rank aggregation methods for the web, in: World Wide Web Conference (WWW), 2001.
[20] P. Erdös, J.W. Moon, On sets on consistent arcs in tournaments, Canad. Math. Bull. 8 (1965) 269–271.
[21] F.V. Fomin, D. Lokshtanov, V. Raman, S. Saurabh, Fast local search algorithm for weighted feedback arc set in tournaments, in: AAAI, 2010, pp. 65–70.
[22] H.A. Jung, On subgraphs without cycles in tournaments, Combinatorial Theory and Its Applications II, 1970, pp. 675–677.
[23] M. Karpinski, W. Schudy, Faster algorithms for feedback arc set tournament, Kemeny rank aggregation and betweenness tournament, CoRR, arXiv:1006.4396, 2010.
[24] J. Kemeny, Mathematics without numbers, Daedalus 88 (1959) 571–591.
[25] J. Kemeny, J. Snell, Mathematical Models in the Social Sciences, Blaisdell, 1962.
[26] C. Kenyon-Mathieu, W. Schudy, How to rank with few errors, in: ACM Symposium on Theory of Computing (STOC), 2007, pp. 95–103.
[27] R.M. McConnell, F. de Montgolfier, Linear-time modular decomposition of directed graphs, Discrete Appl. Math. 145 (2) (2005) 198–209.
[28] V. Raman, S. Saurabh, Parameterized algorithms for feedback set problems and their duals in tournaments, Theoret. Comput. Sci. 351 (3) (2006) 446–458.
[29] K.D. Reid, E.T. Parker, Disproof of a conjecture of Erdös and Moser on tournaments, J. Combin. Theory 9 (1970) 225–238.
[30] S. Seshu, M.B. Reed, Linear Graphs and Electrical Networks, Addison–Wesley, 1961.
[31] P. Slater, Inconsistencies in a schedule of paired comparisons, Biometrika 48 (1961) 303–312.
[32] E. Speckenmeyer, On feedback problems in digraphs, in: Workshop on Graph-Theoretic Concepts in Computer Science (WG), in: Lecture Notes in Comput. Sci., vol. 411, Springer, 1989, pp. 218–231.
[33] J. Spencer, Optimal ranking of tournaments, Networks 1 (1971) 135–138.
[34] S. Thomassé, A $4k^2$ kernel for feedback vertex set, ACM Trans. Algorithms 6 (2) (2010).
[35] A. van Zuylen, Deterministic approximation algorithms for ranking and clusterings, Technical Report 1431, Cornell ORIE, 2005.
[36] A. van Zuylen, R. Hegde, K. Jain, D.P. Williamson, Deterministic pivoting algorithms for constrained ranking and clustering problems, in: ACM–SIAM Symposium on Discrete Algorithms (SODA), 2007, pp. 405–414.
[37] D.H. Younger, Minimum feedback arc sets for a directed graph, IEEE Trans. Circuit Theory 10 (1963) 238–245.

Please cite this article in press as: S. Bessy et al., Kernels for feedback arc set in tournaments, J. Comput. System Sci. (2010), doi:10.1016/j.jcss.2010.10.001

# On the (non-)existence of polynomial kernels for $P_l$-free edge modification problems *

Sylvain Guillemot[1] Christophe Paul[2]
Anthony Perez[2]

[1] Lehrstuhl für Bioinformatik, Friedrich-Schiller Universität Jena
[2] Université Montpellier II - CNRS, LIRMM

**Abstract**

Given a graph $G = (V, E)$ and an integer $k$, an edge modification problem for a graph property $\Pi$ consists in deciding whether there exists a set of edges $F$ of size at most $k$ such that the graph $H = (V, E \triangle F)$ satisfies the property $\Pi$. In the $\Pi$ *edge-completion problem*, the set $F$ of edges is constrained to be disjoint from $E$; in the $\Pi$ *edge-deletion problem*, $F$ is a subset of $E$; no constraint is imposed on $F$ in the $\Pi$ *edge-edition problem*. A number of optimization problems can be expressed in terms of graph modification problems which have been extensively studied in the context of parameterized complexity. When parameterized by the size $k$ of the edge set $F$, it has been proved that if $\Pi$ is an hereditary property characterized by a finite set of forbidden induced subgraphs, then the three $\Pi$ edge-modification problems are FPT [4]. It was then natural to ask [4] whether these problems also admit a polynomial size kernel. Using recent lower bound techniques, Kratsch and Wahlström answered this question negatively [15]. However, the problem remains open on many natural graph classes characterized by forbidden induced subgraphs. Kratsch and Wahlström asked whether the result holds when the forbidden subgraphs are paths or cycles and pointed out that the problem is already open in the case of $P_4$-free graphs (i.e. cographs). This paper provides positive and negative results in that line of research. We prove that parameterized cograph edge modification problems have cubic vertex kernels whereas polynomial kernels are unlikely to exist for the $P_l$-free and $C_l$-free edge-deletion problems for large enough $l$.

## 1 Introduction

An edge modification problem aims at changing the edge set of an input graph $G = (V, E)$ in order to get a certain property $\Pi$ satisfied (see [16] for a recent study). Edge modification problems cover a broad range of graph optimization problems among which completion problems (*e.g.* MINIMUM FILL-IN, *a.k.a* CHORDAL GRAPH COMPLETION [19, 21]), edition problems (*e.g.* CLUSTER EDITING [20]) and edge deletion problems (*e.g.* MAXIMUM PLANAR SUBGRAPH [10]). In a completion problem, the set $F$ of modified edges is constrained to be disjoint from $E$; in an edge deletion problem, $F$ has to be a subset of $E$; and in an edition problem, no restriction applies to $F$. These

*Research supported by the AGAPE project (ANR-09-BLAN-0159).

problems are fundamental in graph theory and play an important role in computational complexity theory (indeed they represent a large number of the earliest NP-Complete problems [10]). Edge modification problems are also relevant in the context of applications as graphs are often used to model data sets which may contain errors. Adding or deleting an edge thereby corresponds to fixing some false negatives or false positives (see *e.g.* [20] in the context of CLUSTER EDITING). Different variants of edge modification problems have been studied in the literature such as graph sandwich problems [11]. Most of the edge modification problems turns out to be NP-Complete [16] and approximation algorithms exist for some known graph properties (see *e.g.* [14, 22]). But for those who want to compute an exact solution, fixed parameter algorithms [5, 8, 17] are a good alternative to cope with such hard problems. In the last decades, edge modification problems have been extensively studied in the context of fixed parameterized complexity (see [4, 7, 13]).

A parameterized problem $Q$ is *fixed parameter tractable* (FPT for short) with respect to parameter $k$ whenever it can be solved in time $f(k).n^{O(1)}$, where $f(k)$ is an arbitrary computable function [5, 17]. In the context of edge modification problems, the size $k$ of the set $F$ of modified edges is a natural parameterization. The generic question is thereby whether a given edge modification problem is FPT for this parameterization. More formally:

PARAMETERIZED $\Pi$ EDGE–MODIFICATION PROBLEM
**Input:** An undirected graph $G = (V, E)$.
**Parameter:** An integer $k \geqslant 0$.
**Question:** Is there a subset $F \subseteq V \times V$ with $|F| \leqslant k$ such that the graph $H = (V, E \triangle F)$ satisfies $\Pi$.

A classical result of parameterized complexity states that a parameterized problem $Q$ is FPT if and only if it admits a *kernelization*. A *kernelization* of a parameterized problem $Q$ is a polynomial time algorithm $\mathcal{K}$ that given an instance $(x, k)$ computes an equivalent instance $\mathcal{K}(x, k) = (x', k')$ such that the size of $x'$ and $k'$ are bounded by a computable function $h()$ depending only on the parameter $k$. The reduced instance $(x', k')$ is called a *kernel* and we say that $Q$ admits a *polynomial kernel* if the function $h()$ is a polynomial. The equivalence between the existence of an FPT algorithm and the existence of a kernelization only yields kernels of (at least) exponential size. Determining whether an FPT problem has a polynomial (or even linear) size kernel is thus an important challenge. Indeed, the existence of such polynomial time reduction algorithm (or pre-processing algorithm or *reduction rules*) really speed-up the resolution of the problem, especially if it is interleaved with other techniques [18]. However, recent results proved that not every fixed parameter tractable problem admits a polynomial kernel [1].

Cai [4] proved that if $\Pi$ is an hereditary graph property characterized by a finite set of forbidden subgraphs, then the PARAMETERIZED $\Pi$ MODIFICATION problems (edge-completion, edge-deletion and edge-edition) are FPT. It was then natural to ask [4] whether these $\Pi$ edge-modification problems also admit a polynomial size kernel. Using recent lower bound techniques, Kratsch and Wahlström answered negatively this question [15]. However, the problem remains open on many natural graph classes characterized by forbidden induced subgraphs. Kratsch and Wahlström asked whether the result holds when the forbidden subgraphs are paths or cycles and pointed out that the

problem is already open in the case of $P_4$-free graphs (i.e. cographs). In this paper, we prove that PARAMETERIZED COGRAPH EDGE MODIFICATION problems have cubic vertex kernels whereas polynomial kernels are unlikely to exist for $P_l$-FREE and $C_l$-FREE EDGE DELETION problems for large enough $l$. The NP-Completeness of the cograph edge-deletion and edge-completion problems have been proved in [6].

**Outline of the paper.** We first establish structural properties of optimal edge-modification sets with respect to modules of the input graph (Section 2). These properties allow us to design general reduction rules (Section 3.1). We then establish cubic kernels using an extra sunflower rule (Section 3.2 and 3.3). Finally, we show it is unlikely that the $C_l$-FREE and the $P_l$-FREE EDGE-DELETION problems have polynomial kernels (Section 4).

# 2 Preliminaries

## 2.1 Notations

We only consider finite undirected graphs without loops nor multiple edges. Given a graph $G = (V, E)$, we denote by $xy$ the edge of $E$ between the vertices $x$ and $y$ of $V$. We set $n = |V|$ and $m = |E|$ (subscripts will be used to avoid possible confusion). The neighbourhood of a vertex $x$ is denoted by $N(x)$. If $S$ is a subset of vertices, then $G[S]$ is the subgraph induced by $S$ (i.e. any edge $xy \in E$ between vertices $x, y \in S$ belongs to $E_{G[S]}$). Given a set of pairs of vertices $F$ and a subset $S \subseteq V$, $F[S]$ denotes the pairs of $F$ with both vertices in $S$. Given two sets $S$ and $S'$, we denote by $S \triangle S'$ their symmetric difference.

## 2.2 Fixed parameter complexity and kernelization

We let $\Sigma$ denote a finite alphabet and $\mathbb{N}$ the set of natural numbers. A *(classical) problem* $Q$ is a subset of $\Sigma^*$, and a string $x \in \Sigma^*$ is an *input* of $Q$. A *parameterized problem* $Q$ over $\Sigma$ is a subset of $\Sigma^* \times \mathbb{N}$. The second component of an input $(x, k)$ of a parameterized problem is called the *parameter*. Given a parameterized problem $Q$, one can derive its unparameterized (or classical) version $\tilde{Q}$ by $\tilde{Q} = \{x\#1^k : (x, k) \in Q\}$, where $\#$ is a symbol that does not belong to $\Sigma$.

A parameterized problem $Q$ is *fixed parameter tractable* (FPT for short) if there is an algorithm which given an instance $(x, k) \in \Sigma^* \times \mathbb{N}$ decides whether $(x, k) \in Q$ in time $f(k).n^{O(1)}$ where $f(k)$ is an arbitrary computable function (see [5, 8, 17]). A *kernelization* of a parameterized problem $Q$ is a polynomial time algorithm $\mathcal{K} : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ which given an instance $(x, k) \in \Sigma^* \times \mathbb{N}$ outputs an instance $(x', k') \in \Sigma^* \times \mathbb{N}$ such that

1. $(x, k) \in Q \Leftrightarrow (x', k') \in Q$ and
2. $|x'|, k' \leqslant h(k)$ for some computable function $h : \mathbb{N} \to \mathbb{N}$.

The reduced instance $(x', k')$ is called a *kernel* and we say that $Q$ admits a *polynomial kernel* if the function $h()$ is a polynomial. It is well known that a parameterized problem $Q$ is FPT if and only if it has a kernelization [17]. But this equivalence only yields (at

least) exponential size kernels. Recent results proved that it is unlikely that every fixed parameter tractable problem admits a polynomial kernel [1]. These results rely on the notion of *(or-)composition algorithms* for parameterized problems, which together with a polynomial kernel would imply a collapse on the polynomial hierarchy [1]. An *or-composition algorithm* for a parameterized problem $Q$ is an algorithm that receives as input a sequence of instances $(x_1, k) \dots (x_t, k)$ with $(x_i, k) \in \Sigma^* \times \mathbb{N}$ for $1 \leqslant i \leqslant t$, runs in time polynomial in $\sum_{i=1}^{t} |x_i| + k$ and outputs an instance $(y, k')$ of $Q$ such that:

1. $(y, k') \in Q \Leftrightarrow (x_i, k) \in Q$ for some $1 \leqslant i \leqslant t$ and
2. $k'$ is polynomial in $k$.

A parameterized problem admitting an *or-composition algorithm* is said to be *or-compositional*.

**Theorem 2.1** *[1, 9] Let $Q$ be an or-compositional parameterized problem whose unparameterized version $\tilde{Q}$ is NP-complete. The problem $Q$ does not admit a polynomial kernel unless $NP \subseteq coNP/Poly$.*

Let $P$ and $Q$ be parameterized problems. A *polynomial time and parameter transformation* from $P$ to $Q$ is a polynomial time computable function $\mathcal{T} : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ which given an instance $(x, k) \in \Sigma^* \times \mathbb{N}$ outputs an instance $(x', k') \in \Sigma^* \times \mathbb{N}$ such that

1. $(x, k) \in P \Leftrightarrow (x', k') \in Q$ and
2. $k' \leqslant p(k)$ for some polynomial $p$.

**Theorem 2.2** *[2] Let $P$ and $Q$ be parameterized problems and let $\tilde{P}$ and $\tilde{Q}$ be their unparameterized versions. Suppose that $\tilde{P}$ is NP-complete and $\tilde{Q}$ belongs to NP. If there is a polynomial time and parameter transformation from $P$ to $Q$ and if $Q$ admits a polynomial kernel, then $P$ also admits a polynomial kernel.*

### 2.3 Modular decomposition and cographs

A *module* in a graph $G = (V, E)$ is a set of vertices $M \subseteq V$ such that for any $x \notin M$ either $M \subseteq N(x)$ or $M \cap N(x) = \emptyset$. Clearly if $M = V$ or $|M| = 1$, then $M$ is a *trivial* module. A graph without any non-trivial module is called *prime*. For two disjoint modules $M$ and $M'$, either all the vertices of $M$ are adjacent to all the vertices of $M'$ or none of the vertices of $M$ is adjacent to any vertex of $M'$. A partition $\mathcal{P} = \{M_1, \dots M_k\}$ of the vertex set $V(G)$ whose parts are modules is a *modular partition*. A *quotient graph* $G_{/\mathcal{P}}$ is associated with any modular partition $\mathcal{P}$: its vertices are the parts of $\mathcal{P}$ and there is an edge between $M_i$ and $M_j$ iff $M_i$ and $M_j$ are adjacent in $G$.

A module $M$ is *strong* if for any module $M'$ distinct from $M$, either $M \cap M' = \emptyset$ or $M \subset M'$ or $M' \subset M$. It is clear from definition that the family of strong modules arranges in an inclusion tree, called the *modular decomposition tree* and denoted $MD(G)$. Each node $N$ of $MD(G)$ is associated with a quotient graph $G_N$ whose vertices correspond to the children $N_1, \dots N_k$ of $N$. (see Figure 1 for an example). We say that a node $N$ of $MD(G)$ is *parallel* if $G_N$ has no edge, *series* if $G_N$ is complete, and *prime* otherwise. For a survey on modular decomposition theory, refer to [12].

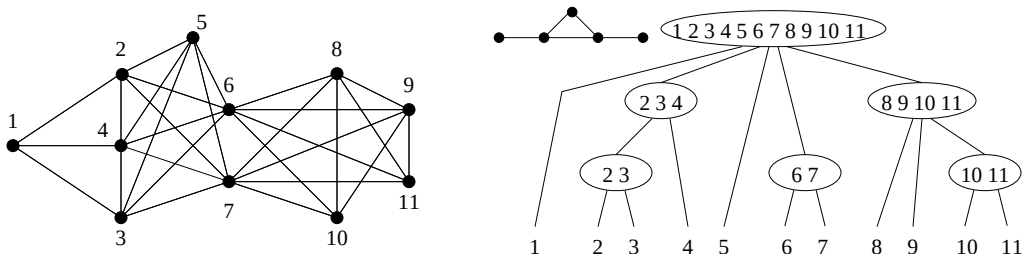


Figure 1: A graph $G$ and its modular decomposition tree $MD(G)$. The root of $MD(G)$ is prime and its quotient graph is the 5 vertex graph depicted eside. Every other node is either parallel or series.

**Definition 2.3** *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two vertex disjoint graphs. The* series composition *of $G_1$ and $G_2$ is the graph $G_1 \otimes G_2 = (V_1 \cup V_2, E_1 \cup E_2 \cup V_1 \times V_2)$. The* parallel composition *of $G_1$ and $G_2$ is the graph $G_1 \oplus G_2 = (V_1 \cup V_2, E_1 \cup E_2)$*

Parallel and series nodes in the modular decomposition tree respectively correspond to a parallel and series composition of their children.

Cographs are commonly known as $P_4$-free graphs (a $P_4$ is an induced path on four vertices). However, they were originally defined as follows:

**Definition 2.4 ([3])** *A graph is a cograph if it can be constructed from single vertex graphs by a sequence of parallel and series composition.*

In particular, this means that the modular decomposition tree of a *cograph* does not contain any prime node. It follows that cographs are also known as the totally decomposable graphs for the modular decomposition.

# 3 Polynomial kernels for cograph modification problems

## 3.1 Modular decomposition based reduction rules

Since cographs correspond to $P_4$-free graphs, cograph edge-modification problems consist in adding or deleting at most $k$ edges to the input graph in order to make it $P_4$-free. The use of the modular decomposition tree in our algorithms follows from the following observation:

**Observation 3.1** *[Folklore] Let $M$ be a module of a graph $G = (V, E)$ and $\{a, b, c, d\}$ be four vertices inducing a $P_4$ of $G$, then $|M \cap \{a, b, c, d\}| \leqslant 1$ or $\{a, b, c, d\} \subseteq M$.*

This means that given a modular partition $\mathcal{P}$ of a graph $G$, any induced $P_4$ of $G$ is either contained in a part of $\mathcal{P}$ or intersects the parts of $\mathcal{P}$ in at most one vertex. This observation allows us to show that a cograph edge-modification problem can be solved independently on modules of the partition $\mathcal{P}$ and on the quotient graph $G_{/\mathcal{P}}$, as stated by the following results:

**Observation 3.2** *Let $M$ be a non-trivial module of a graph $G = (V, E)$. Let $F_M$ be an optimal edge-deletion (resp. edge-completion, edge-edition) set of $G[M]$ and let $F_{opt}$ be an optimal edge-deletion (resp. edge-completion, edge-edition) set of $G$. Then*

$$F = (F_{opt} \setminus F_{opt}[M]) \cup F_M$$

*is an optimal edge-deletion (resp. edge-completion, edge-edition) set of $G$.*

*Proof:* By Observation 3.1, it follows that $H = (V, E \triangle F)$ is $P_4$-free, thereby $F$ is an edge-deletion set. As being a cograph is an hereditary property, $F_{opt}[M]$ is an edge-deletion set of $G[M]$. Now observe that $|F| = |F_{opt}|$ since otherwise $|F_M| > |F_{opt}[M]|$, which would contradict the optimality of $F_M$. The same argument holds for edge-completion and edge-edition sets. □

**Lemma 3.3** *Let $M$ be a module of a graph $G = (V, E)$. There exists an optimal edge-deletion (resp. edge-completion, edge-edition) set $F$ such that $M$ is a module of the cograph $H = (V, E \triangle F)$.*

*Proof:* Let $F_{opt}$ be an optimal edge-deletion set and denote $H_{opt} = (V, E \triangle F_{opt})$. Let $x$ be a vertex of $M$ such that $|\{xy \in F : y \notin M\}|$ is minimum. We argue that the following set of edges is an optimal edge-deletion set:

$$F = F_{opt}[M] \cup F_{opt}[V \setminus M] \cup \{zy : z \in M, y \notin M, xy \in F_{opt}\}$$

First observe that by construction $M$ is a module in the graph $H = (V, E \triangle F)$ and that by the choice of $x$, $|F| \leqslant |F_{opt}|$. Let us prove that $H$ is $P_4$-free. As $H[M]$ and $H[V \setminus M]$ are respectively isomorphic to $H_{opt}[M]$ and $H_{opt}[V \setminus M]$, they are $P_4$-free. So if $H$ contains an induced $P_4$, its vertices $\{a, b, c, d\}$ intersect $M$ and $V \setminus M$. As $M$ is a module of $H$ it follows by Observation 3.1 that $|M \cap \{a, b, c, d\}| = 1$ (say $a \in M \cap \{a, b, c, d\}$). It follows by construction of $F$, that $\{x, b, c, d\}$ also induces a $P_4$ in $H_{opt}$, contradicting the assumption that $F_{opt}$ is an edge-deletion set. So we proved that $F$ is an edge-deletion set of $G$ which preserves the module $M$ and is not larger than $F_{opt}$. The same proof holds for edge-completion and edge-edition sets. □

**Lemma 3.4** *Let $G = (V, E)$ be an arbitrary graph. There exists an optimal edge-deletion (resp. edge-completion, edge-edition) set $F$ such that every module $M$ of $G$ is module of the cograph $H = (V, E \triangle F)$.*

*Proof:* We prove the statement for edge-deletion sets by induction on the number of modules of a graph. The same proof applies for edge-completion and edge-edition sets. Observe that the result trivially holds if $G$ is a prime graph and follows from Lemma 3.3 if $G$ contains a unique non-trivial module.

Let us now assume that the property holds for every graph with at most $t$ non-trivial modules. Let $G$ be a graph with $t + 1$ non-trivial modules and let $M$ be a non-trivial module of $G$ which is minimal for inclusion. By induction hypothesis, the statement holds on $G[M]$ (since it is prime) and on the graph $G_{M \to x}$ where $M$ has been contracted to a single vertex $x$ (since it contains at most $t$ non-trivial modules). The conclusion follows from Observation 3.2. □

We now present three reduction rules which apply to the three cograph edge-modification problems we consider. The second reduction rule is not required to obtain a polynomial kernel for each of these problems. However, it will ease the analysis of the structure of a reduced graph.

**Rule 1** *Remove the connected components of $G$ which are cographs.*

**Rule 2** *If $C = G_1 \otimes G_2$ is a connected component of $G$, then replace $C$ by $G_1 \oplus G_2$.*

**Rule 3** *If $M$ is a non-trivial module of $G$ which is strictly contained in a connected component and is not an independent set of size at most $k+1$, then return the graph $G' \oplus G[M]$ where $G'$ is obtained from $G$ by replacing $M$ by an independent set module of size $min\{|M|, k+1\}$.*

Observe that if $G[M]$ is a cograph, adding a disjoint copy to the graph is useless since it will then be removed by Rule 1.

**Lemma 3.5** *The reduction rules 1, 2 and 3 are safe and can be carried out in linear time.*

*Proof:* The three rules can be computed in linear time using any linear time modular decomposition algorithm [12]. The first rule is trivially safe. The second rule is safe by Lemma 3.4. The safeness of Rule 3 also follows from Lemma 3.4: there always exists an optimal solution that updates all or none of the edges between any two disjoint modules. Thereby if a module $M$ has size larger than $k+1$, none of the edges (or non-edges) $xy$ with $x \in M$, $y \notin M$ can be changed in such a solution. Shrinking $M$ into an independent set of size $k+1$ and adding a disjoint copy of $G[M]$ (to keep track of the edge modification inside the module) is thereby safe. □

The analysis of the size of the kernel relies on the following structural property of the modular decomposition tree of an instance reduced under Rule 1, Rule 2 and Rule 3.

**Observation 3.6** *Let $G$ be a graph reduced under Rule 1, Rule 2 and Rule 3. If $C$ is a non prime connected component of $G$, then the modules of $C$ are independent sets of size at most $k+1$.*

*Proof:* By Rule 2, none of the connected components of $G$ results from a series composition. By Rule 3, a module which is not the union of some connected components of $G$ has size at most $k+1$ and is an independent set. □

Observe that these three reduction rules preserve the parameter. However, Rule 3 increases the number of vertices of the instance. Nevertheless, we will be able to bound the number of vertices of a reduced instance.

It remains to show that computing a reduced graph requires polynomial time. Let us mention that it is safe to apply Rule 2 and Rule 3 only on strong modules (in Rule 2, $G_1$ can be chosen as a strong module).

**Lemma 3.7** *Given a graph $G = (V, E)$, computing a graph reduced under Rule 1, Rule 2 and Rule 3 requires polynomial time.*

*Proof:* Let us say that a module $M$ of $G$ is *reduced* if it is an independent set of size at most $k + 1$ or the disjoint union of some connected components of $G$ (observe that connected components of $G$ are also modules of $G$). By Observation 3.6, if $G$ is reduced under Rule 1, Rule 2 and Rule 3, then every module of $G$ is reduced. Notice that if every strong module of $G$ is reduced, then every module of $G$ is reduced. So to prove the statement, we count the number of strong modules (*i.e.* nodes of the modular decomposition tree $MD(G)$) which are not reduced.

Let us also remark that if a connected component $C$ is a cograph with at least two vertices, then a series of applications of Rule 2 eventually transforms $C$ in a set of isolated vertices. This means that we can assume that the applications of Rule 1 is postponed to the end of the reduction process. This will ease the argument below.

When Rule 3 is applied, then by definition the number of non-reduced strong modules decreases by one. When Rule 2 is applied, unless $G_1$ is an independent set of size at most $k + 1$, then the number of non-reduced strong modules also decreases by one. But observe that if $G_1$ is an independent set of size at most $k + 1$, then its vertices will be removed by Rule 1 as they will become isolated vertices. As the number of strong modules of a graph is bounded by the number of vertices, this proves that a series of at most $n$ applications of Rule 2 and Rule 3 is enough to compute a reduced graph. □

## 3.2 Cograph edge-deletion (and edge-completion)

In addition to the previous reduction rules, we need the classical *sunflower* rule to obtain a polynomial kernel for the parameterized cograph edge-deletion problem.

**Rule 4** *If $e$ is an edge of $G$ that belongs to a set $\mathcal{P}$ of at least $k + 1$ $P_4$'s such that $e$ is the only common edge of any two distinct $P_4$'s of $\mathcal{P}$, then remove $e$ and decrease $k$ by one.*

**Observation 3.8** *The reduction rule 4 is safe and can be carried out in polynomial time.*

*Proof:* It is clear that the edge $e$ has to be deleted as otherwise at least $k + 1$ edge deletions would be required to break all the $P_4$'s of the set $\mathcal{P}$. Such an edge, if it exists, can be found in polynomial time if one computes the set of all $P_4$'s of the input graph (which can be done in $O(n^4)$ time). □

To analyse the size of a reduced graph $G = (V, E)$, we study the structure of the cograph $H = (V, E \triangle F)$ resulting from the removal of an optimal (of size at most $k$) edge-deletion set $F$. The modular decomposition tree (or cotree) is the appropriate tool for this analysis.

**Theorem 3.9** *The parameterized cograph edge-deletion problem admits a cubic vertex kernel.*

*Proof:* Let $G = (V, E)$ be a graph reduced under Rule 1, Rule 2, Rule 3 and Rule 4 that can be turned into a cograph by deleting at most $k$ edges. Let $F$ be an optimal edge-deletion set and denote by $H = (V, E \triangle F)$ the cograph resulting from the deletion of $F$ and by $T$ its cotree. We will count the number of leaves of $T$ (or equivalently of vertices of $G$ and $H$).

Observe that since a set of $k$ edges covers at most $2k$ vertices, $T$ contains at most $2k$ affected leaves (i.e. leaves corresponding to a vertex incident to a removed edge). We say that an internal node of the cotree $T$ is *affected* if it is the least common ancestor of two affected leaves. Notice that there are at most $2k$ affected nodes.

We first argue that the root of $T$ is a parallel node and is affected. Assume that the root of $T$ is a series node: since no edges are added to $G$, this would imply that $G$ is not reduced under Rule 2, a contradiction. Moreover, since $G$ is reduced under Rule 1, none of its connected components is a cograph. It follows that every connected component of $G$ contains a vertex incident to a removed edge, and thus that every subtree attached to the root contains an affected leaf as a descendant. Hence the root of $T$ is an affected node.

**Claim 3.10** *Let $p$ be an affected leaf or an affected node different from the root, and $q$ be the least affected ancestor of $p$. The path between $p$ and $q$ has length at most $2k + 3$.*

*Proof.* Observe first that the result trivially holds if $q$ is the root of $T$ and $p$ one of its children. In all other cases, let $M$ be the set of leaves descendant of $p$ in $T$. We claim that $M$ contains a leaf $x$ which is incident to a removed edge $xy$, with $y \notin M$. If $p$ is an affected leaf then this is true by definition. Otherwise, if $p$ is an affected node different from the root, assume by contradiction that all the removed edges in $M$ are of the form $uv$ with $u, v \in M$. In particular, this implies that $M$ is a module of $G$ strictly contained in a connected component. By Observation 3.6, it follows that $M$ is an independent set and hence contains no edges, a contradiction. Let $t$ be the least common ancestor of $x$ and $y$. The node $t$ is a parallel node which is an ancestor of $p$ and $q$ (observe that we may have $t = q$). Assume by contradiction that the path between $x$ and $t$ in $T$ contains a sequence of $2k + 3$ consecutive non-affected nodes. The type of these nodes is alternatively series and parallel. So we can find a sequence $s_1, p_1 \dots s_{k+1}, p_{k+1}$ of consecutive non-affected nodes with $s_i$ (resp. $p_i$) being the father of $p_i$ (resp. $s_{i+1}$) and with $s_i$'s being series nodes and the $p_i$'s being parallel node. Now each of the $s_i$'s (resp. $p_i$) has a non-affected leaf $a_i$ (resp. $b_i$) which is not a descendant of $p_i$ (resp. $s_{i+1}$). Observe that for every $i \in [1, k + 1]$ the vertex set $\{b_i, a_i, x, y\}$ induces a $P_4$ in $G$. Thereby we found a set of $k + 1$ $P_4$'s in $G$ pairwise intersecting on the edge $xy$. It follows that $G$ is not reduced by the Rule 4: contradiction. This implies that the path between $p$ and $q$ contains at most $2k + 3$ non-affected nodes. $\diamond$

Since there are at most $2k$ affected nodes and $2k$ affected leaves, $T$ contains at most $(4k-1)(2k+3)+2k$ internal nodes. As $G$ is reduced, Observation 3.6 implies that each of these $O(k^2)$ nodes is attached to a set of at most $k+1$ leaves or a parallel node with $k+1$ children. It follows that $T$ contains at most $2k + (k + 1)[(4k - 1)(2k + 3) + 2k] \leqslant 8k^3 + 20k^2 + 11k$ leaves, which correspond to the number of vertices of $G$.

We now conclude with the time complexity needed to compute the kernel. Since the application of Rule 4 decreases the value of the parameter (which is not changed by the other rules), Rule 4 is applied at most $k \leqslant n^2$ times. It then follows from Lemma 3.7 that a reduced instance can be computed in polynomial time. □

The following corollary simply follows from the observation that the family of cographs is closed under complementation (since the complement of a $P_4$ is a $P_4$).

**Corollary 3.11** *The parameterized cograph edge-completion problem admits a cubic vertex kernel.*

### 3.3 Cograph edge-edition

The lines of the proof for the cubic kernel of the edge-edition problem are essentially the same as for the edge-deletion problem. But since edges can be added and deleted, the reduction Rule 4 has to be refined in order to avoid that a single edge addition breaks an arbitrary large set of $P_4$'s.

**Rule 5** *If $\{x, y\}$ is a pair of vertices of $G$ that belongs to a set $S$ of $t \geqslant k+1$ quadruples $P_i = \{x, y, a_i, b_i\}$ such that for $1 \leqslant i \leqslant t$, every $P_i$ induces a $P_4$ and for any $1 \leqslant i < j \leqslant t$, $P_i \cap P_j = \{x, y\}$, then change $E$ into $E \triangle \{xy\}$ and decrease $k$ by one.*

As for reduction Rule 4, it is clear that reduction Rule 5 is safe and can be applied in polynomial time. The kernelization algorithm of cograph edge-edition consists of an exhaustive application of Rules 1, 2, 3 and 5.

**Theorem 3.12** *The parameterized cograph edge-edition problem has a cubic vertex kernel.*

*Proof:* Let $G = (V, E)$ be a graph reduced under Rule 1, Rule 2, Rule 3 and Rule 5 that can be turned into a cograph by editing at most $k$ edges. Let $H$ be the cograph obtained by an optimal edge-edition. The cotree of $H$ is denoted by $T$. Unlike in the edge-deletion problem, the root of $T$ is not necessary a parallel node. However it is still true that the root of $T$ is affected. Indeed, assume first that the root of $T$ is a series node. Then it is affected since otherwise $G$ would not be reduced under Rule 2. Now, assume that the root is a non affected parallel node. This means that at most one of its children contains an affected leaf as descendant, and hence that $G$ is not reduced under Rule 1: contradiction.

In the following we assume w.l.o.g. that the root of $T$ is a parallel node. We prove that Claim 3.10 still holds in this case. Let $p$ be an affected leaf or an affected node different from the root, and $q$ be the least affected ancestor of $p$. Observe that the result is trivially true if $q$ is the root of $T$ and $p$ one of its children. In all other cases, let $M$ be the set of leaves descendant of $p$ in $T$. As in the proof of Theorem 3.9, there must exist an edited edge $xy$ with $x \in M, y \notin M$ (otherwise $M$ would be a module of $G$, i.e. an independent set by Observation 3.6 and would thus not be edited by Observation 3.2).

Now the proof follows the arguments of the proof of Theorem 3.9, if one can find in $T$ a path of $2k + 3$ consecutive non-affected nodes between $p$ and $q$, then $G$ is not reduced under Rule 5. Proving that $T$ contains $O(k^2)$ nodes and thereby $O(k^3)$ leaves.

The fact that a reduced instance can be computed in polynomial time follows from Lemma 3.7 and the observation that Rule 5 decreases the value of the parameter and requires polynomial time. □

For the deletion (resp. edition) problem there exists a graph reduced under Rule 1, Rule 2, Rule 3 and Rule 4 (resp. Rule 5) that achieves the cubic bound (see Figure 2).

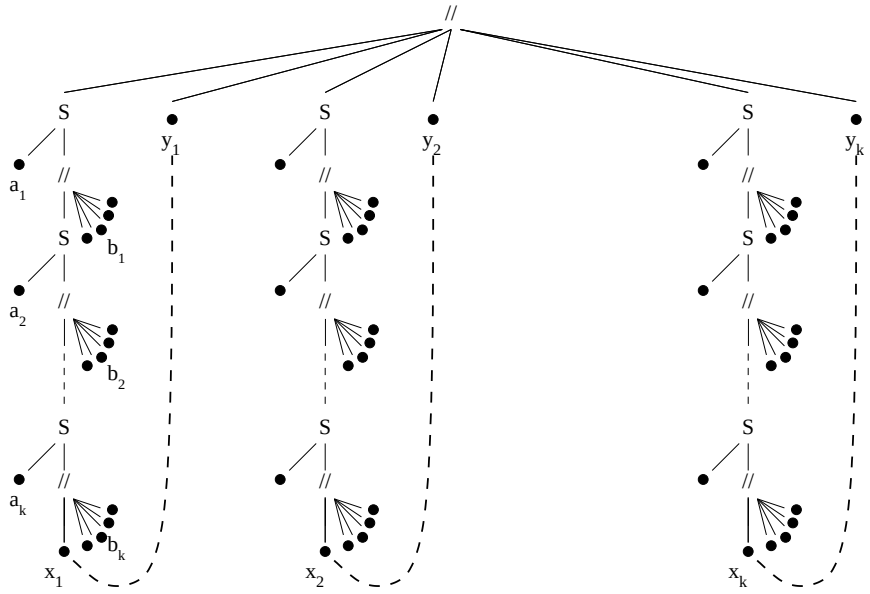


Figure 2: A reduced graph $G$ with $k(k+1)^2+k$ vertices for which $k$ edge deletions, namely the $x_iy_i$'s for $i \in [1,k]$, are required to obtain a cograph $H$. The cotree $T$ of $H$ is represented. Each parallel node of $T$ which is not the root has $k+2$ children, $k+1$ of which are leaves. The root of $T$ has $2k$ children.

# 4 Kernel lower bounds for $P_l$-free edge-deletion problems

In [15], Kratsch and Wahlström show that the Not-1-in-3-sat problem has no polynomial kernelization under a complexity-theoretic assumption ($NP \not\subseteq coNP/poly$). We observe that their argument still applies to a graph restriction of Not-1-in-3-sat where the constraints arise from the triangles of an input graph.

## 4.1 A graphic version of the Not-1-in-3-sat problem

For a graph $G=(V,E)$, an edge-bicoloring is a function $B : E \to \{0,1\}$. A *partial edge-bicoloring* of $G$ is an edge-bicoloring of a subset of edges of $E$. An edge colored 1 (resp. 0) is called a 1-edge (resp. 0-edge). We say that the edge-bicoloring $B'$ *extends* a partial edge-bicoloring $B$ if for every $e \in E$ colored by $B$, then $B(e) = B'(e)$. The weight of an edge-bicoloring is the number $\omega(B)$ of 1-edges. An edge-bicoloring is *valid* if every triangle of $G$ contains either zero, two or three 1-edges. We consider the following problem:

NOT-1-IN-3-EDGE-TRIANGLE
**Input:** An undirected graph $G = (V, E)$ and a partial edge-bicoloring $B : E \to \{0, 1\}$.
**Parameter:** An integer $k \in \mathbb{N}$.
**Question:** Can we extend $B$ to a valid edge-bicoloring $B'$ of weight at most $k$?

**Proposition 4.1** NOT-1-IN-3-EDGE-TRIANGLE *is NP-complete and or-compositional.*

*Proof:* The NP-hardness follows from a reduction from VERTEX COVER. Let $(G, k)$ be an instance of VERTEX COVER [10], where $G = (V, E)$. We create an instance $(G', B, k')$ of NOT-1-IN-3-EDGE-TRIANGLE as follows. The graph $G'$ is obtained from $G$ by adding a dominating vertex $q$, the partial edge-bicoloring $B$ is such that $B(e) = 1$ for every $e \in E$, and we let $k' = |E| + k$. As the triangles of $G$ are monochromatic, the constraints to obtain a valid extension of $B$ are carried by the triangles of the form $quv$ with $uv \in E$. It is easy to observe that $(G', B, k')$ has a valid edge-bicoloring extension of weight $k'$ iff $G$ has a vertex cover of size $k$. As NOT-1-IN-3-EDGE-TRIANGLE clearly belongs to NP, the NP-completeness follows.

We now show that NOT-1-IN-3-EDGE-TRIANGLE is or-compositional. The proof closely follows the proof of [15] for NOT-1-IN-3-SAT. We first need the following result:

**Claim 4.2** *Given an instance $(G, B, k)$ of* NOT-1-IN-3-EDGE-TRIANGLE*, and two positive integers $r$ and $k'$ such that $k' \geq k + r$, we can compute in polynomial time an equivalent instance $(G', B', k')$ of* NOT-1-IN-3-EDGE-TRIANGLE *such that $\omega(B') = \omega(B) + r$.*

*Proof.* To build $G'$, we first add to $G$ a set $F$ of $r$ new isolated edges $e_1 \dots e_r$ such that $B'(e_i) = 1$ for all $i \in [r]$. Then we add to the resulting graph $k' - (k + r)$ gadgets as follows: let $e_j = u_j v_j$ (with $j \in [k' - (k + r)]$) be an arbitrary 1-edge of $G$; add the triangles $u_j v_j x_j, v_j x_j y_j$ with $B'(v_j y_j) = B'(x_j y_j) = 0$. The edges $e_j$'s are not necessarily distinct. Observe that in any valid edge-bicoloring of $G'$ extending $B'$, the edge $v_j x_j$ (for every $j \in [k' - (k + r)]$) is a 0-edge while the edge $u_j x_j$ is a 1-edge. It follows that $(G, B, k)$ is a positive instance if and only if $(G', B', k')$ is a positive instance as the set $F$ increases the weight by $r$ and the added triangles by $k' - (k + r)$. ◇

Consider a sequence $(G_1, B_1, k) \dots (G_t, B_t, k)$ of instances of NOT-1-IN-3-EDGE-TRIANGLE. We denote by $E_1(j)$ the set of 1-edges of $(G_j, B_j, k)$. By Claim 4.2, we can assume w.l.o.g. that $|E_1(j)| = s \leqslant k$, for $1 \leqslant j \leqslant t$. We can also assume that $t \leqslant 3^k$ since otherwise an exact branching algorithm could solve the problem. Moreover, for the sake of the construction, we assume $t = 2^l$ (duplicating some instance $(G_i, B_i, k)$ if necessary).

Intuitively, the graph $G$ of the composed instance $(G, B, k')$ is built on the disjoint union of the $G_j$'s, $1 \leqslant j \leqslant t$. Then, as a selection gadget, we add a "tree-like graph" $T$ connecting a "root edge" $r$ to edges $e_j$ for $j = 1, ..., t$. Finally, for every $1 \leqslant j \leqslant t$, the 1-edges of the graph $G_j$ are connected via a propagation gadget to the edge $e_j$ in $T$. The root edge is the unique 1-edge of $G$. The copies of the $G_j$'s inherit the 0-edges of the $G_j$'s. The idea is that the selection gadget guarantees that at least one of the $e_j$'s edge gets colored 1. Then the propagation gadgets attached to that edge $e_j$ transmit color 1 to the copies of every 1-edge of $G_j$.

Formally, we do the following: (i) we start with a complete binary tree $T_0$ with $t$ leaves; (ii) to each node $u$ of $T_0$, we associate an edge $e_u$ in $T$ as follows: if $u$ is associated to the edge $xy$ and if $u$ has two children $v, v'$, we create a new vertex $z$ and we let $e_v = xz, e_{v'} = yz$. The leaves of $T_0$ are then associated to edges $e_1, ..., e_t$. Now, for every $1 \leqslant j \leqslant t$, the propagation gadget $S_j$ consists of vertex-disjoint graphs $S_{j,e}$ for every edge $e$ of $E_1(j)$. If $e = uv$ and $e_j = xy$, then $S_{j,e}$ consists of four triangles $uva, vab, abx, bxy$, with edges $ua, vb, ax, by$ colored 0 by $B$ (the other edges remain uncolored). Again the unique 1-edge of $B$ is the root edge of $T$, in particular the edges of the $E_1(j)$ are uncolored by $B$. However, the 0-edge sets of the $G_j$'s are inherited by $B$. (see Figure 3)

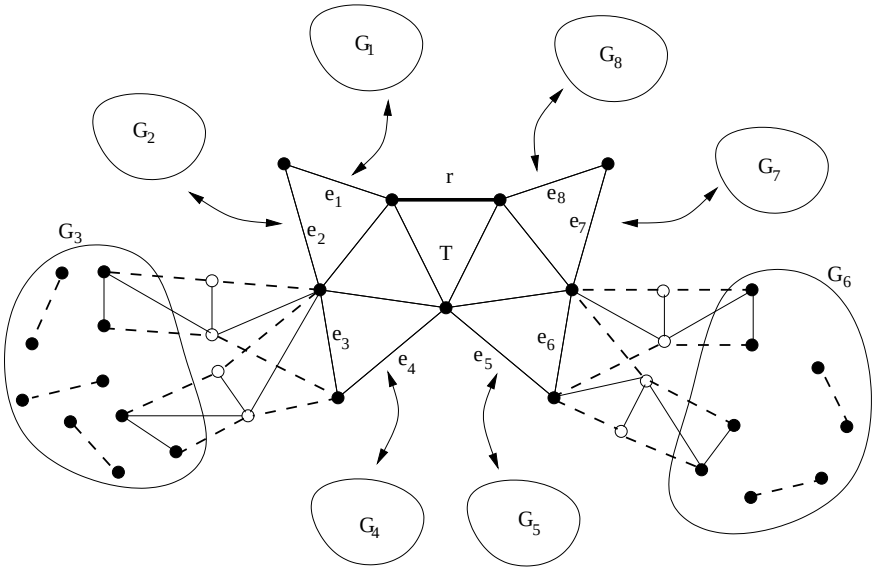


Figure 3: The instance $(G, B, k')$ built from a sequence $(G_1, B_1, k), \ldots, (G_t, B_t, k)$ with $t = 2^3$. The unique 1-edge is $r$. Every "leaf edge" $e_j$ of $T$ is linked to the copies of the 1-edges of $(G_j, B_j, k)$ via the propagation gadget. The 0-edges are depicted as dotted lines: they either belong to a propagation gadget or correspond to a 0-edge of some $(G_j, B_j, k)$.

Observe first that every valid edge-bicoloring extending $B$ has to assign color 1 to at least one edge $e_j$, for $1 \leqslant j \leqslant t$. Then the edges of $E_1(j)$ and the $3s$ non 0-edges of $S_j$ are also assigned color 1. It follows that if we choose $k' = k + 3s + l$, then $(G, B, k')$ is a positive instance if and only if there exists $1 \leqslant j \leqslant t$ such that $(G_j, B_j, k)$ is a positive instance. □

The following corollary follows from Theorem 2.1:

**Corollary 4.3** *The* Not-1-in-3-edge-triangle *problem does not admit a polynomial kernel unless* $NP \subseteq coNP/poly$.

The problem Tripartite-Not-1-in-3-edge-triangle is the restriction of Not-1-in-3-edge-triangle where the input graph $G$ is 3-colorable. The hardness results obtained for Not-1-in-3-edge-triangle carry over to this restriction:

**Lemma 4.4** *The* Tripartite-Not-1-in-3-edge-triangle *problem does not admit a polynomial kernel unless* $NP \subseteq coNP/poly$.

*Proof:* The proof uses Theorem 2.2, that is we provide a polynomial parameter-preserving transformation from Not-1-in-3-edge-triangle to Tripartite-Not-1-in-3-edge-triangle. By Proposition 4.1, Not-1-in-3-edge-triangle is NP-complete. Observe that Tripartite-Not-1-in-3-edge-triangle clearly belongs to NP.

Let $(G, B, k)$ be an instance of Not-1-in-3-edge-triangle. We build an instance $(G', B', 6k)$ of Tripartite-Not-1-in-3-edge-triangle in the following way. Suppose that $G = (V, E)$, then $G'$ has vertex set $V' = \{v_1, v_2, v_3 : v \in V\}$, and has edge set $E' = \{u_1v_2, u_1v_3, u_2v_3 : u = v \text{ or } uv \in E\}$. The partial edge-bicoloring $B'$ is defined as follows: $B'(u_iu_j) = 0$ for $1 \leqslant i < j \leqslant 3$; if the edge $uv$ of $G$ is colored, then $B'(u_iv_j) = B(uv)$ for $1 \leqslant i, j \leqslant 3$; the other edges of $G'$ are uncolored.

Observe that every valid edge-bicoloring extending $B'$ assigns the same color to the six edges of $G'$ associated with an edge $uv$ of $G$: indeed, given $u_iv_j, u_kv_l$ $1 \leqslant i, j, k, l \leqslant 3$, if $i = j$ this holds since $B'(v_kv_l) = 0$, if $k = l$ this holds since $B'(u_iu_j) = 0$, and otherwise this follows by transitivity. It is then easy to see that solutions of $(G, B, k)$ and solutions of $(G', B', 6k)$ are in one-to-one correspondence. □

## 4.2 Negative results for Γ-free edge deletion problems

In this section, we show that unless $NP \subseteq coNP/poly$, the $C_l$-free edge-deletion and the $P_l$-free edge-deletion problems have no polynomial kernel for large enough $l \in \mathbb{N}$. To that aim, we provide polynomial time and parameter transformations from Tripartite-Not-1-in-3-edge-triangle to the Annotated $C_l$-free edge-deletion problem and to the Annotated $P_l$-free edge-deletion problem. For a graph $\Gamma$, the Annotated $\Gamma$-free edge-deletion problem is defined as follows:

Annotated Γ-free edge-deletion
**Input:** An undirected graph $G = (V, E)$ and a subset $S$ of vertices.
**Parameter:** An integer $k \in \mathbb{N}$.
**Question:** Is there a subset $F \subseteq E \cap (S \times S)$ such that $H = (V, E \setminus F)$ is $\Gamma$-free?

Observe that the Annotated $\Gamma$-free edge-deletion problem reduces to the (unannotated) $\Gamma$-free edge-deletion problem whenever $\Gamma$ is closed under twin addition: it suffices to add for every vertex $v \in V \setminus S$ a set of $k + 1$ twin vertices. Clearly this transformation also preserves the parameter.

Observe also that we can restrict the Tripartite-Not-1-in-3-edge-triangle problem to instances $(G, B, k)$ not containing any 0-edge (*i.e.* $B(e) = 1$ whenever it is defined). The reason is that any uncolored edge $e = uw$ of $G$ can be forced to be assigned color 0 in every valid edge-bicoloring extending $B$ by adding to $G$ $k + 1$ new vertices $v_1, \ldots, v_k$ such that $uv_iw$, $1 \leqslant i \leqslant k$, is an uncolored triangle. Clearly if $e$ is a 1-edge of an edge-bicoloring $B'$ extending $B$, $B'$ needs at least $k + 1$ 1-edges to be valid: $e$ plus one edge per triangle. The same argument was used in [15] for the Not-1-in-3-sat problem.

**Theorem 4.5** *The $C_l$-FREE EDGE-DELETION problem has no polynomial kernel for any $l \geqslant 12$, unless $NP \subseteq coNP/poly$.*

*Proof:* We describe a polynomial time and parameter transformation from the restriction of TRIPARTITE-NOT-1-IN-3-EDGE-TRIANGLE without 0-edges to ANNOTATED $C_l$-FREE EDGE-DELETION. The statement then follows from Theorem 2.2 and the fact that ANNOTATED $C_l$-FREE EDGE-DELETION reduces to $C_l$-FREE EDGE-DELETION.

Let $(G, B, k)$ be an instance of the TRIPARTITE-NOT-1-IN-3-EDGE-TRIANGLE problem, where $V_1, V_2, V_3$ are disjoint independent sets of $G = (V, E)$. The construction of the instance $(H, S, k')$ of ANNOTATED $C_l$-FREE EDGE-DELETION works as follows. First the sets $V_1$, $V_2$ and $V_3$ are turned into cliques and the 1-edges of $G$ are removed. In addition to $V$, the graph $H$ contains a set $U$ of new vertices. For each pair $t = (e, v)$ with $e = uw$ an edge of $G$ and $v$ a vertex of $G$, such that $\{u, v, w\}$ induces a triangle in $G$, we create a path $P_t$ of length $l-1$ between $u$ and $w$ in $H$ (the internal vertices of $P_t$ are added to $U$). Notice that each triangle of $G$ generates three such paths in $H$. It remains to add some *safety* edges incident to the vertices of $U$. Every two vertices $x$ and $y$ of $U$ that do not belong to the same path are made adjacent. In every path $P_t$, we select an internal vertex $c_t$, called its *centre*, at distance $(l-1)/2$ from $u$. Every centre vertex $c_t$ is made adjacent to $V \setminus \{u, v, w\}$. We denote by $H = (V_H, E_H)$ the resulting graph. To complete the description of $(H, S, k')$ we set $S = V$ and the parameter $k' = k - k_1$ where $k_1$ is the number of 1-edges of $(G, B, k)$.

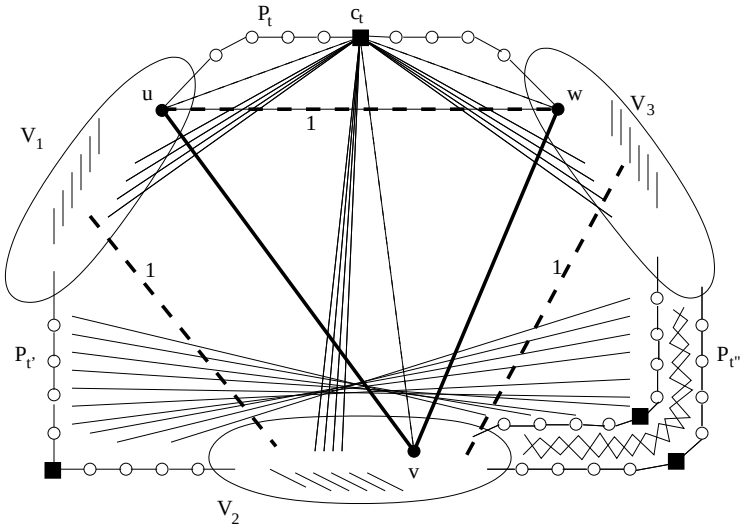


Figure 4: The graph $H = (V_H, E_H)$ built from an instance $(G, B, k)$ of the TRIPARTITE-NOT-1-IN-3-EDGE-TRIANGLE problem for $l = 12$. The white and the square vertices form the set $U$ of new vertices. The independent sets $V_1$, $V_2$ and $V_3$ of $G$ are turned into cliques. The thick dotted edges are the removed 1-edges of $(G, B, k)$. The non 1-edges of $(G, B, k)$ are preserved in $H$.

**Claim 4.6** *A subset of vertices $C \in V_H$ induces a cycle of length $l$ iff $G$ contains a triangle $uvw$, with $e = uw$ a 1-edge and $uv$, $vw$ uncolored edges, such that $C = P_t \cup \{v\}$ with $t = (e, v)$.*

*Proof.* By construction, if $G$ contains a triangle $uvw$ with a unique 1-edge $e = uw$, then $C = P_t \cup \{v\}$ (with $t = (e, v)$) induces a cycle of length $l$ in $H$ (keep in mind that the 1-edges of $G$ are removed from $H$). Let $C$ be an induced $C_l$ in $H$. Observe that as $V_1$, $V_2$ and $V_3$ are turned into cliques, $|C \cap V| \leqslant 6$. Thereby $C$ intersects the vertex set $U$. We now argue that there exists a path $P_t$, with $t = (e, v)$ and $e = uw$, containing the vertices of $C \cap U$. Otherwise, since every pair of vertices of $U$ belonging to two distinct paths $P_t$ and $P_{t'}$ (with $t \neq t'$) are adjacent, we would have $|C \cap U| \leqslant 4$ and thus $|C| \leqslant 4 + 6 < l$. It follows that $u$ or $w$ belongs to $C$. We prove that they both belong to $C$. Assume $w \notin C$, then $C$ uses a safety edge incident to the centre vertex $c_t$ and half of the internal vertices of $P_t$ does not belong to $C$. Thereby $|C| \leqslant 6 + (l-3)/2 + 1$: contradiction with the hypothesis $l \geqslant 12$. Finally as $P_t$ contains $l - 1$ vertices, $C$ contains an extra vertex and $uw$ are not adjacent. As $c_t$ is adjacent to every vertex of $V$ except $u$, $v$ and $w$, we have that $C = P_t \cap \{v\}$ as announced and $uv, wv \in E_H$. Now the existence of $P_t$ witnesses the existence of the triangle $uvw$ in $G$. As $uv, wv \in E_H$ and $uw \notin E_H$, $uw$ is the only 1-edge of the triangle $uvw$. $\diamond$

We now argue for the correctness of the transformation. Suppose that there exists a set $F$ of allowed edges of size at most $k'$ such that $H' = (V_H, E_H \setminus F)$ is $C_l$-free. Define the edge-bicoloring $B'$ of $E$ as follows: $B'(e) = 1$ if $e \in F$, $B'(e) = 0$ otherwise. As by assumption $B$ does not assign color 0 to any edge, $B'$ extends $B$ and has weight at most $|F| + k_1 \leqslant k' + k_1 = k$. Besides, $B'$ is a valid edge-bicoloring of $G$. Let $t = (e, v)$ with $e = uw$ be a pair such that $\{u, v, w\}$ induces a triangle in $G$. If we had $B(uw) = 1$, $B'(uv) = B'(vw) = 0$, we would obtain that $P_t \cup \{v\}$ induces a $C_l$ in $H'$, impossible. Conversely, suppose that $B'$ is valid edge-bicoloring of weight at most $k$ of $G$ which extends $B$. Let $F \subseteq E$ be the set of edges such that $B'(e) = 1$ but are uncolored by $B$. By construction $F$ is a set of allowed edges of $H$ of size at most $k - k_1$. Since $B'$ is a valid edge-bicoloring of $G$, Claim 4.6 implies that $H' = (V_H, E_H \setminus F)$ is $C_l$-free. $\square$

A slight modification of the above construction yields the following:

**Theorem 4.7** *The $P_l$-*FREE EDGE-DELETION *problem has no polynomial kernel for any $l \geqslant 13$, unless $NP \subseteq coNP/poly$.*

*Proof:* Let $(G, B, k)$ be an instance of the TRIPARTITE-NOT-1-IN-3-EDGE-TRIANGLE problem not containing any 0-edge and such that $V_1, V_2, V_3$ are disjoint independent sets of $G = (V, E)$. We modify the construction given in Theorem 4.5 to obtain an instance $(H, S, k')$ of ANNOTATED $P_l$-FREE EDGE-DELETION problem. The vertex set $V_H$ of $H$ consists of the union of $V$ and a set $U$ of new vertices. The sets $V_1$, $V_2$ and $V_3$ are again turned into cliques and the 1-edges of $E$ are not duplicated in $E_H$. But for each pair $t = (e, v)$, with $e = uw \in E$ and $v \in V$ such that $\{u, v, w\}$ is a triangle of $G$, the associated gadget $Q_t$ is no longer a path. Instead, $Q_t$ consist of two paths $Q_t^u$ and $Q_t^w$: $Q_t^u$ is a path of length $(l-1)/3$ containing $u$ as extremity and $Q_t^w$ is a path of length $2(l-1)/3$ containing $w$ as extremity. The vertices of $Q_t \setminus \{u, w\}$ are added to $U$. As before for every $t \neq t'$ we add all the edges between vertices of $Q_t$ and $Q_{t'}$. The centre vertex $c_t$ of $Q_t$ is the vertex of $Q_t^w$ at distance $(l-1)/3$ from $w$. The centre vertex is made adjacent to every vertex of $V$ except $u$, $v$ and $w$. To complete the description of $(H, S, k')$ we set $S = V$ and $k' = k - k_1$ where $k_1$ is the number of 1-edges of $(G, B, k)$.

The correctness proof of the construction follows the same lines than the proof of Proposition 4.5. It now relies on the following claim that characterizes the possible induced $P_l$'s.

**Claim 4.8** *A subset of vertices $Q \in V_H$ induces a path of length $l$ iff $G$ contains a triangle $uvw$, with $e = uw$ a 1-edge and $uv$, $vw$ uncolored edges, such that $Q = Q_t \cup \{v\}$ with $t = (e, v)$.*

*Proof.* By construction, if $G$ contains a triangle $uvw$ with a unique 1-edge $e = uw$, then $Q = Q_t \cup \{v\}$ (with $t = (e, v)$) induces a path of length $l$ in $H$ (keep in mind that the 1-edges of $G$ are removed from $H$). Let $Q$ be an induced $P_l$ in $H$. As in the proof of Claim 4.6, observe that $|Q \cap V| \leqslant 6$ and thereby $Q$ intersects the vertex set $U$ and that there exists a unique pair $t = (e, v)$ with $e = uw$ such that $Q_t$ contains $Q \cap U$. By the choice of the length of $Q_t^u$ and $Q_t^w$, $Q \cap U$ intersects both $Q_t^u$ and $Q_t^w$. It follows that $u$ and $w$ belongs to $Q$. Assume that $Q$ uses an edge $xc_t$ such that $x \notin Q_t$. Then half of the vertices of $Q_t^w$ does not belong to $Q$, which would contradict the hypothesis $l \geqslant 13$. Finally as by construction $Q_t$ contains $l - 1$ vertices, we need at least one extra vertex from $V$. Since $c_t$ is adjacent to all vertices of $V$ except $u$, $v$ and $v$, that extra vertex can only be $v$. Moreover the chord $uw$ cannot exist in $H$, meaning that $uw$ is a 1-edge of $(G, B, k)$. □

# 5 Conclusion

In this paper we have shown that the PARAMETERIZED COGRAPH EDGE MODIFICATION problems admit vertex cubic kernels. Moreover, we provide evidence that the $C_l$-FREE EDGE-DELETION and the $P_l$-FREE EDGE-DELETION problems do not admit polynomial kernels for large enough $l$ (under a complexity-theoretic assumption [1]). These problems were left open by Kratsch and Wahlström in [15]. The value of $l$ being respectively (at least) 12 and 13, one remaining question is thus to determine whether the $C_p$-FREE EDGE-DELETION and the $P_p$-FREE EDGE-DELETION problems admit polynomial kernels for $4 \leqslant p < 12$ in the former case, and $4 < p < 13$ in the latter.

# References


[1] H. Bodlaender, R. Downey, M. Fellows, and D. Hermelin. On problems without polynomial kernels. In *ICALP*, number 5125 in LNCS, pages 563–574, 2008.

[2] H. L. Bodlaender, S. Thomassé, and A. Yeo. Kernel bounds for disjoint cycles and disjoint paths. In *ESA*, volume 5757 of *LNCS*, pages 635–646, 2009.

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

[4] L. Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58(4):171–176, 1996.

[5] R. Downey and M. Fellows. *Parameterized complexity.* Springer, 1999.

[6] E. S. El-Mallah and C. Colbourn. The complexity of some edge deletion problems. *IEEE Transactions on Circuits and Systems*, 35(3):354–362, 1988.

[7] M. Fellows, M. Langston, F. Rosamond, and P. Shaw. Efficient parameterized preprocessing for cluster editing. In *Fundamentals of Computation Theory, 16th International Symposium, (FCT)*, number 4639 in LNCS, pages 312–321, 2007.

[8] J. Flum and M. Grohe. *Parameterized complexity theorey.* Texts in Theoretical Computer Science. Springer, 2006.

[9] L. Fortnow and R. Santhanam. Infeasibility of instance compression and succinct PCPs for NP. In *STOC*, pages 133–142, 2008.

[10] M. Garey and S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness.* Freeman, 1978.

[11] M. Golumbic, H. Kaplan, and R. Shamir. Graph sandwich problems. *Journal of Algorithms*, 19:449–473, 1995.

[12] M. Habib and C. Paul. A survey on algorithmic aspects of modular decomposition. *Computer Science Review*, 4(1):41–59, 2010.

[13] P. Heggernes, C. Paul, J. A. Telle, and Y. Villanger. Interval completion with few edges. In *STOC*, pages 374–381, 2007.

[14] C. Kenyon-Mathieu and W. Schudy. How to rank with few errors. In *Annual ACM Symposium on Theory of Computing (STOC)*, pages 95–103, 2007.

[15] S. Kratsch and M. Wahlström. Two edge modification problems without polynomial kernels. In *IWPEC*, volume 5917 of *LNCS*, pages 264–275, 2009.

[16] A. Natanzon, R. Shamir, and R. Sharan. Complexity classification of some edge modification problems. *Discrete Applied Mathematics*, 113(1):109 – 128, 2001.

[17] R. Niedermeier. *Invitation to fixed parameter algorithms*, volume 31 of *Oxford Lectures Series in Mathematics and its Applications.* Oxford University Press, 2006.

[18] R. Niedermeier and P. Rossmanith. A general method to speed up fixed-parameter-tractable algorithms. *Information Processing Letters*, 73(3-4):125–129, 2000.

[19] D. Rose. A graph-theoretic study of the numerical solution of sparse positive systems of linear equations. *Graph Theory and Computing*, pages 183–217, 1972.

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

[21] R. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM journal of Computing*, 13:566–579, 1984.

[22] A. van Zuylen and D. Williamson. Deterministic algorithms for rank aggragation and other ranking and clustering problems. In *WAOA*, volume 4927 of *LNCS*, pages 260–273, 2008.

# Conflict Packing yields linear vertex-kernels for Rooted Triplet Inconsistency and other problems *

Christophe Paul, Anthony Perez, Stéphan Thomassé
LIRMM - Université Montpellier 2, CNRS - FRANCE

**Abstract**

We develop a technique that we call *Conflict Packing* in the context of kernelization. We illustrate this technique on several well-studied problems: Feedback Arc Set in Tournaments, Dense Rooted Triplet Inconsistency and Betweenness in Tournaments. For the former, one is given a tournament $T = (V, A)$ and seeks a set of at most $k$ arcs whose reversal in $T$ leads to an acyclic tournament. While a linear vertex-kernel is already known for this problem [6], using the Conflict Packing allows us to find a so-called *safe partition*, the central tool of the kernelization algorithm in [6], with simpler arguments. Regarding the Dense Rooted Triplet Inconsistency problem, one is given a set of vertices $V$ and a dense collection $\mathcal{R}$ of rooted binary trees over three vertices of $V$ and seeks a rooted tree over $V$ containing all but at most $k$ triplets from $\mathcal{R}$. Using again the Conflict Packing, we prove that the Dense Rooted Triplet Inconsistency problem admits a linear vertex-kernel. This result improves the best known bound of $O(k^2)$ vertices for this problem [19]. Finally, we use this technique to obtain a linear vertex-kernel for Betweenness in Tournaments, where one is given a set of vertices $V$ and a dense collection $\mathcal{R}$ of *betweenness triplets* and seeks an ordering containing all but at most $k$ triplets from $\mathcal{R}$. To the best of our knowledge this result constitutes the first polynomial kernel for the problem.

## 1 Introduction

The concept of *fixed parameter algorithms* [13] has been introduced to cope with NP-Hard problems. For a given (parameterized) problem, the goal is to identify a parameter $k$, independent from the data-size $n$, which captures the exponential growth of the complexity cost to solve the problem in hand. That is the complexity of such an (FPT) algorithm is $f(k) \cdot n^{0(1)}$, where $f$ is an arbitrary computable function. As one of the most powerful techniques to design efficient fixed parameter algorithms, *kernelization algorithms* [7] have attracted a lot of attention during the last few years. A kernelization algorithm transforms, in polynomial time (via *reduction rules*), an arbitrary instance $(I, k)$ of a parameterized problem into an equivalent instance $(I', k')$ with the property that the parameter $k'$ and the size $|I'|$ of the reduced instance only depend on $k$. The smaller the size of the reduced instance (called *kernel*) is, the faster the problem can be solved. Indeed, once reduced the instance can be efficiently tackle with any exact algorithms (e.g. bounded search tree or exponential time algorithms).

*Research supported by the AGAPE project (ANR-09-BLAN-0159) and the Phylariane project (ANR-08-EMER-011-01).

In the design of polynomial kernels, a few type of reduction rules are oftenly used. *Sunflower* rules or *domination* rules are classical reduction rules which locally affect the instance. These type of rules may be enough to obtain polynomial size kernels but linear kernels often require a more global "attack". More recently, reduction rules based on matching theory [28] or on bidimensionality theory [15] have been proposed. In this paper, we develop and push further a kernelization technique used for a few parameterized problems [9, 30], called *Conflict Packing*, which also uses matching arguments. Combined with a polynomial time algorithm that computes an accurate vertex partition (called *safe partition* in [6]), Conflict Packing yields linear vertex-kernels.

In this extended abstract, we illustrate the Conflict Packing technique on three parameterized problems. We first obtain a linear vertex-kernel for Feedback Arc Set in Tournaments: while such a kernel was already known to exist [6], our proofs are simpler and shorter. Then we obtain the main result of this paper, namely the first linear vertex-kernel for Dense Rooted Triplet Inconsistency. Finally we apply the technique on Betweenness in Tournaments and obtain a linear vertex-kernel. No polynomial kernel was known before.

**Feedback Arc Set in Tournaments (**$k$-FAST) Let $T = (V, A)$ be a tournament on $n$ vertices, *i.e.* a directed graph obtained from an arbitrary orientation of the complete (undirected) graph, and $k$ be an integer. The task is to check whether there exists a subset of at most $k$ arcs of $A$ whose reversal transform $T$ into an acyclic (i.e. transtive) tournament. In other words, $k$-FAST consists of computing a linear vertex ordering $v_1 \dots v_n$ having at most $k$ *backward arcs* ($v_i \to v_j$ with $i > j$). It is well known that a tournament is transitive if and only if it does not contain a directed triangle (circuit on 3 vertices). $k$-FAST is a well studied problem from the combinatorial [14, 25] as well as from the algorithmic point of view [3, 23, 29]. It is known to be NP-complete [3, 11], but fixed parameter tractable [4, 22, 24]. The first kernelization algorithms for $k$-FAST [4, 12] yield $O(k^2)$ vertex-kernels. Recently, a linear vertex-kernel has been proposed [6]. More precisely, using a PTAS which computes a linear vertex ordering with at most $(1 + \epsilon)k$ backward arcs, the authors of [6] show how to find in polynomial time an ordered vertex partition, called *safe partition* $\mathcal{P}$, of $T$. Roughly speaking, a vertex partition is *safe* if the backward arcs whose extremities lie in different part can be reversed independently from the others (inside the parts). We prove that the Conflict Packing technique (a maximal collection of arc-disjoint triangles) can be used to compute such a partition.

**Dense Rooted Triplet Inconsistency (**$k$-dense RTI) The use of fixed parameter algorithms in computational biology and more specifically in phylogenetics has lead to efficient solutions to handle practical data set, see [17] for a survey. This evolutionary history of a set $V$ of species is often represented by a tree (rooted or not), whose leaves represent the species of $V$. In this context, if we are given a set of phylogenetic trees on overlapping set of species, one would like to check whether this partial information can be combined in a common supertree [16]. The tree to be reconstructed could be unrooted or rooted. In this paper we consider the rooted setting. Then, the simplest case consists in testing whether a collection $\mathcal{R}$ of rooted binary trees of three leaves in $V$, called *rooted triplets*, are *consistent*: does there exist a binary tree $T$ on leaves $V$ such that every triplet $\{a, b, c\}$ of $\mathcal{R}$ is homeomorphic to the subtree of $T$ spanning $\{a, b, c\}$? This problem can be solved in polynomial time [1]. When $\mathcal{R}$ is not consistent, different optimization problems can be considered: removing a minimum number of leaves or removing a minimum number of triplets (see e.g. [10, 27]). We consider the parameterized version of the latter problem, called $k$-dense RTI,

where one is given a dense collection of rooted triplets $\mathcal{R}$ and an integer $k$ (the parameter) and seeks a rooted tree over $V$ containing all but at most $k$ triplets from $\mathcal{R}$. It is known that when $\mathcal{R}$ is dense, i.e. contains exactly one rooted triplet for every triple of leaves (or vertices), then it is consistent with a binary tree if and only if it does not contain any conflict on four leaves [5, 19]. The $k$-DENSE RTI problem is known to be NP-complete [10] but fixed parameter tractable [18, 19], the fastest algorithm running in time $O(n^4 + 2^{O(k^{1/3}logk)})$ [19]. Moreover, [19] provided a quadratic vertex-kernel for $k$-DENSE RTI. However, unlike $k$-FAST, no PTAS nor constant approximation algorithm is known [19]. Using *Conflict Packing* enables us to obtain a linear vertex-kernel for this problem. This result improves the best known bound of $O(k^2)$ vertices for this problem [19].

**Betweenness in Tournaments ($k$-BETWEENNESSTOUR)** In this problem one is given a set of vertices $V$ and a dense collection $\mathcal{R}$ of *betweenness triplets* and seeks an ordering containing all but at most $k$ triplets from $\mathcal{R}$. The $k$-BETWEENNESSTOUR problem is NP-Complete [2] but fixed-parameter tractable [22, 26]. Using *Conflict Packing* we obtain a linear vertex-kernel for this problem, which is to the best of our knowledge the first polynomial kernel for this problem.

**Outline** We first illustrate the Conflict Packing technique on the $k$-FAST problem, proving how to compute a so-called safe partition in polynomial time (Section 2). Next, we generalize the results to the $k$-DENSE RTI problem (Section 3). Finally, we give another example where this technique can be applied, namely $k$-BETWEENNESSTOUR (Section 4).

## 2 Linear vertex-kernel for $k$-FAST

**Preliminaries.** Let $T = (V, A)$ be a tournament. We write $uv$ whenever the arc of $A$ between vertices $u$ and $v$ is oriented from $u$ to $v$. If $V' \subseteq V$, then $T[V'] = (V', A')$ is the subtournament induced by $V'$, that is $A' = \{uv \in A \mid u \in V', v \in V'\}$. If $A' \subseteq A$, then $T[A'] = (V', A')$ denotes the digraph where $V' \subseteq V$ contains the vertices incident to some arc of $A'$. A tournament $T = (V, A)$ is *transitive* if for every triple of vertices $u$, $v$, $w$ such that $uv \in A$ and $vw \in A$, then $uw \in A$. A *directed triangle* is a circuit of size three, i.e. a set of vertices $\{u, v, w\}$ such that $\{uv, vw, wu\} \subseteq A$.

**Lemma 2.1 (Folklore)** *Let $T = (V, A)$ be a tournament. Then the following properties are equivalent: (i) $T$ is acyclic; (ii) $T$ is transitive; (iii) $T$ does not contain any directed triangle.*

Clearly $T$ is transitive if and only if there exists an ordering $\sigma$ on $V$ such that for every $u \in V$ and $v \in V$ with $\sigma(u) < \sigma(v)$ (also denoted $u <_\sigma v$) then $uv \in A$. Such an ordering is called *transitive*. We use $T_\sigma = (V, A, \sigma)$ to denote a tournament whose vertices are ordered with respect to some ordering $\sigma$. An arc $vu \in A$ such that $u <_\sigma v$ is called *backward* in $T_\sigma$. In other words, $T$ is transitive if and only if there exists a total ordering $\sigma$ of its vertices such that $T_\sigma$ does not contains any backward arc. Hereafter, a directed triangle will be called a *conflict*. Our kernel uses the following rule from [4].

**Rule 1 (Irrelevant vertex)** *Remove any vertex $v$ that does not belong to any conflict.*

**Certificate and safe partition** The following definitions are adapted from notions introduced in [6]. Let $e = vu$ be a backward arc of an ordered tournament $T_\sigma$. The *span* of $e$ is the set of vertices $span(e) = \{w \in V \mid u <_\sigma w <_\sigma v\}$. If $w \in span(e)$ is a vertex not incident to any backward arc, then $c(e) = \{u, v, w\}$ is a *certificate* of $e$. Observe that $c(e)$ induces a directed triangle. By convention, when speaking of an arc $e$ of a certificate $c$ we mean that $e$ belongs to $T[c]$. If $F \subseteq A$ is a set of backward arcs of $T_\sigma$, we can *certify* $F$ whenever there exists a set $c(F) = \{c(f) : f \in F\}$ of arc-disjoint certificates (i.e. for every distinct $e$ and $f$ of $F$, $|c(e) \cap c(f)| \leqslant 1$).

If $T_\sigma = (V, A, \sigma)$, then $\mathcal{P} = \{V_1, \ldots, V_l\}$ is an *ordered partition* of $T_\sigma$ if it is a partition of $V$ and for every $i \in [l]$, $V_i$ is a set of consecutive vertices in $\sigma$. By convention, if $i < j$, then for every $u \in V_i$ and $v \in V_j$ we have $u <_\sigma v$. We denote by $A_O = \{uv \in A \mid u \in V_i, v \in V_j, i \neq j\}$ the subset of *outer-arcs*. We say that an ordered partition $\mathcal{P} = \{V_1, \ldots, V_l\}$ is a *safe partition* of an ordered tournament $T_\sigma = (V, A, \sigma)$ if $A_O$ contains at least one backward arc and if it is possible to certify the backward arcs of $T_\sigma[A_O]$ only with outer-arcs of $A_O$. The following rule was central in the linear kernel of [6]:

**Rule 2 (Safe partition)** *Let $T_\sigma$ be an ordered tournament, and $\mathcal{P} = \{V_1, \ldots, V_l\}$ be a safe partition of $T_\sigma$ with $F$ the set of backward arcs of $T_\sigma[A_O]$. Then reverse all the arcs of $F$ and decrease $k$ by $|F|$.*

**Conflict packing.** Our kernelization algorithm applies the two rules above. The basic idea to identify a safe partition in polynomial time is to compute a maximal packing of arc-disjoint conflicts $\mathcal{C}$, called *conflict packing*. We first give a bound on the number of vertices $V(\mathcal{C})$ covered by a conflict packing $\mathcal{C}$. An instance of $k$-FAST is *positive* if there exists a set of at most $k$ arcs whose reversal lead to a transitive tournament.

**Lemma 2.2** *Let $T = (V, A)$ be a positive instance of $k$-FAST and $\mathcal{C}$ be a conflict packing of $T$. Then $|V(\mathcal{C})| \leq 3k$.*

**Proof**. By definition, the conflicts of $\mathcal{C}$ are arc-disjoint. Hence at least one arc per triangle of $\mathcal{C}$ has to be reversed. As $|\mathcal{C}| \leq k$, $\mathcal{C}$ covers at most $3k$ vertices. □

We now use the maximality of a conflict packing $\mathcal{C}$ to compute a particular ordering $\sigma$ of $T$. Providing that $V \setminus V(\mathcal{C})$ is large enough (with respect to parameter $k$) we will next prove that a safe partition $\mathcal{P}$ of $T_\sigma$ can be identified.

**Lemma 2.3 (Conflict Packing)** *Let $T = (V, A)$ be an instance of $k$-FAST and $\mathcal{C}$ a conflict packing of $T$. There exists an ordering of $T$ whose backward arcs $uv$ are such that $u, v \in V(\mathcal{C})$.*

**Proof**. We denote by $G$ the set of *good* vertices, i.e. $G = V \setminus V(\mathcal{C})$. Clearly by maximality of $\mathcal{C}$, $T' = T[G]$ is acyclic. Observe also that for every $v \in V(\mathcal{C})$, $T[G \cup \{v\}]$ is acyclic: otherwise it would contain a conflict which is by construction arc-disjoint from those of $\mathcal{C}$. Let $\sigma'$ be the transitive ordering of $T'$. For every vertex $v \in V(\mathcal{C})$, there is a unique pair of consecutive vertices $u$ and $w$ in $\sigma'$ such that the insertion of $v$ between $u$ and $w$ yields the transitive ordering of $T[G \cup \{v\}]$. The pair $u, w$ is the *locus* of $v$. Consider any ordering $\sigma$ on $V$ such that: for $u, w \in G$, if $u <_{\sigma'} w$, then $u <_\sigma w$; and if $v \in V(\mathcal{C})$, then $u <_\sigma v <_\sigma w$ where the pair $u, w$ is the locus of $v$. By the previous arguments, every backward arc $uv$ of $\sigma$ is such that $u, v \in V(\mathcal{C})$. □

An ordering obtained through the above statement is a *nice* ordering. We now prove the claimed result.

**Theorem 2.4** *The $k$-FAST problem admits a kernel with at most $4k$ vertices that can be computed in polynomial time.*

**Proof**. Let $T = (V, A)$ be a positive instance of $k$-FAST reduced under Rule 1. We greedily (hence in polynomial time) compute a conflict packing $\mathcal{C}$ of $T$ and let $\sigma$ be a nice ordering of $V$. Consider the bipartite graph $B = (I \cup G, E)$ where $(i)$ $G = V \setminus V(\mathcal{C})$, $(ii)$ there is a vertex $i_{vu}$ in $I$ for every backward arc $vu$ of $T_\sigma$ and $(iii)$ $i_{vu}w \in E$ if $w \in G$ and $\{u, v, w\}$ is a certificate of $vu$. Observe that any matching in $B$ of size at least $k+1$ would correspond to a conflict packing (*i.e.* a collection of arc-disjoint conflicts) of size at least $k+1$, which cannot be. Hence a minimum vertex cover $D$ of $B$ has size at most $k$ [8]. We denote $D_1 = D \cap I$ and $D_2 = D \cap G$. Assume that $|V| > 4k$. Then since $|D_2| \leqslant k$ and $|V(\mathcal{C})| \leqslant 3k$ (by Lemma 2.2), $G \setminus D_2 \neq \emptyset$. Let $\mathcal{P} = \{V_1, \ldots, V_l\}$ be the ordered partition of $T_\sigma$ such that every part $V_i$ consists of either a vertex of $G \setminus D_2$ or a maximal subset of consecutive vertices (in $\sigma$) of $V \setminus (G \setminus D_2)$.

**Claim 1** *$\mathcal{P}$ is a safe partition of $T_\sigma$.*

*Proof.* Let $w$ be a vertex of $G \setminus D_2$. By Lemma 2.3, $w$ is is not incident to any backward arc. As $T$ is reduced under Rule 1, there must exist a backward arc $e = vu$ such that $w \in span(e)$. It follows that $A_O$ contains at least one backward arc. Let $e = vu \in A_O$ be a backward arc of $\sigma$. By construction of $\mathcal{P}$, there exists a vertex $w \in (G \setminus D_2) \cap span(e)$. Then $\{u, v, w\}$ is a certificate of $e$ and $i_{vu}w$ is an edge of $B$. Observe that as $D$ is a vertex cover and $w \notin D_2$, the vertex $i_{vu}$ has to belong to $D_1$ to cover the edge $i_{vu}w$. Thereby the subset $I' \subseteq I$ corresponding to the backward arcs of $A_O$ is included in $D_1$.

Finally, we argue that $I'$ can be matched into $G \setminus D_2$ in $B$. Assume there exists $I'' \subseteq I'$ such that $|I''| > |N(I'') \cap (G \setminus D_2)|$. As there is no edge in $B$ between $I \setminus D_1$ and $G \setminus D_2$ ($D$ is a vertex cover of $B$), the set $D' = (D \setminus I'') \cup (N(I'') \cap (G \setminus D_2))$ is a vertex cover of $B$ and $|D'| < |D|$: contradicting the minimality of $D$ (see Figure 1). Thereby for every subset $I'' \subseteq I'$, we have $|I''| \leqslant |N(I'') \cap (G \setminus D_2)|$. By Hall's theorem [20], $I'$ can be matched into $G \setminus D_2$.

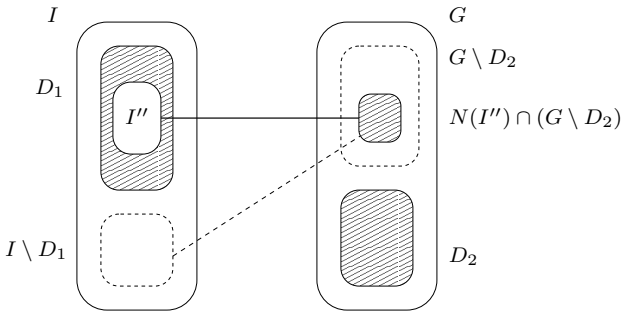


Figure 1: Illustration of the case $|I''| > |N(I'') \cap (G \setminus D_2)|$. The dashed sets represent the vertex cover $D'$.

As every vertex of $G \setminus D_2$ is a singleton in $\mathcal{P}$, the existence of the matching shows that the backward arcs of $A_O$ can be certified using arcs of $A_O$ only, and hence $\mathcal{P}$ is safe. ◇

Hence if $|V| > 4k$, there exists a safe partition that can be computed in polynomial time, and we can reduce the tournament using Rule 2. We then apply Rule 1 and repeat the previous steps until we either do not find a safe partition or $k < 0$. In the former case we know that $|V| \leqslant 4k$; in the latter case, we return a small trivial No-instance. This concludes the proof. □

## 3 Linear vertex-kernel for $k$-DENSE RTI

The kernelization algorithm for $k$-DENSE RTI follows the same lines than the kernelization algorithm for $k$-FAST. It involves two rules: the first removes *irrelevant* leaves and the second deals with a *safe partition* of the instance. The first rule was already used to obtain a quadratic kernel in [19]. As an instance of $k$-DENSE RTI is constituted of triplets that choose one vertex (observe that an instance of $k$-FAST can be seen as couples that choose one vertex), we have to adapt the notions of *conflict* and *certificate*. In $k$-FAST, the goal was to find an adequate ordering on the vertices of the input tournament, and a safe partition was thus defined as an ordered partition. Here, as we seek for a tree, the notion of *safe partition* needs to be adjusted. To prove our safe partition rule, we use again the technique of conflict packing presented in the previous section. We first give some definitions and notations.

**Preliminaries.** A *rooted triplet* $t$ is a rooted binary tree on a set of three leaves $V(t) = \{a, b, c\}$. We write $t = ab|c$ if $a$ and $b$ are siblings of a child of the root of $t$, the other child of the root being $c$. We also say that $t$ *chooses* $c$. An instance of $k$-DENSE RTI is a pair $R = (V, \mathcal{R})$, where $\mathcal{R}$ is a set of rooted triplets on $V$. We only consider *dense* instances, that is $\mathcal{R}$ contains exactly one rooted triplet for every triple of $V$. For a subset $S \subseteq V$, we define $\mathcal{R}[S] = \{t \in \mathcal{R} \mid V(t) \subseteq S\}$ and $R[S] = (S, \mathcal{R}[S])$. A rooted binary tree is defined *over* a set $V$ if the elements of $V$ are in one-to-one correspondence with the leaves of $T$. Hereafter the elements of $V$ are called *leaves* and the term *nodes* stands for internal nodes of $T$. By $T_{|S}$, with $S \subseteq V$, we denote the rooted binary tree over $S$ which is homeomorphic to the subtree of $T$ spanning the leaves of $S$. Let $t \in \mathcal{R}$ be a rooted triplet and $T$ be a tree over $V$. Then $t$ is *consistent with* $T$ if $T_{|V(t)} = t$, and *inconsistent* otherwise. A set of rooted triplets $\mathcal{R}$ is *consistent* if there exists a rooted binary tree $T$ over $V$ such that every $t \in \mathcal{R}$ is consistent with respect to $T$. If such a tree does not exist, then $\mathcal{R}$ is *inconsistent*. A *conflict* $C$ is a subset of $V$ such that $\mathcal{R}[C]$ is inconsistent. If a dense set of rooted triplets $\mathcal{R}$ is consistent, then there exists a unique binary tree $T$ in which every rooted triplet of $\mathcal{R}$ is consistent. We know from [19] that:

**Lemma 3.1** *Let $R = (V, \mathcal{R})$ be an instance of $k$-DENSE RTI. The following properties are equivalent: (i) $\mathcal{R}$ is consistent; (ii) $\mathcal{R}$ contains no conflict on four leaves; (iii) $\mathcal{R}$ contains no conflict of the form $\{ab|c, cd|b, bd|a\}$ or $\{ab|c, cd|b, ad|b\}$.*

It follows that, as in $k$-FAST where it was enough to consider directed triangles as conflict, we can restrict our attention to conflicts on sets of four leaves. Hereafter, the term of *conflict* is only used on set of four leaves. Our kernelization algorithm uses the following rule from [19].

**Rule 3 (Irrelevant leaf)** *Remove any leaf $v \in V$ that does not belong to any conflict.*

**Certificate.** An *embedded* instance of $k$-DENSE RTI is a triple $R_T = (V, \mathcal{R}, T)$ such that $\mathcal{R}$ is a dense set of rooted triplets on $V$ and $T$ is a rooted binary tree over $V$. When dealing with an embedded instance $R_T$, the inconsistency of a rooted triplet always refers to the tree $T$. If $x$ is a node of $T$, then $T_x$ denotes the subtree of $T$ rooted in $x$. Given three leaves $\{a, b, c\}$, we define $span(t)$ as the set of leaves of $V$ contained in $T_{lca(\{a,b,c\})}$, where $lca$ stands for *least common ancestor*. Moreover, given $S \subseteq V$ we define $R_T[S] = (S, \mathcal{R}[S], T_{|S})$. Finally, *editing* an inconsistent rooted triplet $t = ab|c$ *w.r.t.* $T$ means replacing $t$ with the rooted triplet on $\{a, b, c\}$ consistent w.r.t. $T$. As mentioned earlier, our kernelization algorithm only uses conflicts on sets of four leaves. The following lemma describes more precisely the topology of such conflicts.

**Lemma 3.2** *Let $R_T = (V, \mathcal{R}, T)$ be an embedded instance of $k$-DENSE RTI. Let $\{a, b, c, d\}$ be a set of leaves such that $t = bc|a$ is the only inconsistent rooted triplet of $R_T[\{a, b, c, d\}]$. Then $\{a, b, c, d\}$ is a conflict if and only if $d \in span(t)$.*

**Proof**. Since $t = bc|a$ is inconsistent, $T_{|\{a,b,c\}}$ is homeomorphic to $ab|c$ or $ac|b$. The two cases are symmetric, so assume the former holds. Suppose that $d \in span(t)$. By assumption the rooted triplet $t' \in \mathcal{R}$ on $\{b, c, d\}$ is consistent with $T$. Consider the two possible cases for $t'$ (observe that $t' = bc|d$ is not possible since it implies $d \notin span(t)$):

- $t' = bd|c$: observe that if $t'$ and $bc|a$ are consistent with a tree $T'$, then $T'_{|\{a,c,d\}}$ is homeomorphic to $cd|a$. Whereas if $t'$ and $ab|c$ are consistent with another tree $T''$, then $T''_{|\{a,c,d\}}$ is homeomorphic to $ad|c$.
- $t' = cd|b$: observe that if $t'$ and $bc|a$ are consistent with a tree $T'$, then $T'_{|\{a,b,d\}}$ is homeomorphic to $bd|a$. Whereas if $t'$ and $ab|c$ are consistent with another tree $T''$, then $T''_{|\{a,b,d\}}$ is homeomorphic to $ab|d$.

So if $d \in span(t)$, whichever choice on $\{b, c, d\}$ leads to an inconsistency. It follows that $\{a, b, c, d\}$ is a conflict. Assume now that $d \notin span(t)$. Again, as $bc|a$ is the unique rooted triplet of $R_T[\{a, b, c, d\}]$ inconsistent with $T$, every rooted triplet containing $d$ chooses $d$. It follows that $\{a, b, c, d\}$ is not a conflict. □

In the following, given an embedded instance $R_T = (V, \mathcal{R}, T)$ of $k$-DENSE RTI, a conflict containing exactly one rooted triplet inconsistent with $T$ is called a *simple conflict*. We now formally define the notion of *certificate* for an embedded instance $R_T$. Let $t$ be a rooted triplet inconsistent with $T$. If $d \in span(t)$ does not belong to any inconsistent rooted triplet then $c(t) = V(t) \cup \{d\}$ is a *certificate* of $t$. Observe that $c(t)$ induces a simple conflict. By convention, when speaking of a rooted triplet $t$ of a certificate $c$ we mean that $t$ belongs to $R[c]$. If $\mathcal{F} \subseteq \mathcal{R}$ is a set of rooted triplets inconsistent with $T$, we can *certify* $\mathcal{F}$ whenever it is possible to find a set $c(\mathcal{F}) = \{c(t) : t \in \mathcal{F}\}$ of *triplet-disjoint* certificates (i.e. for every distinct $t$ and $t'$, $|c(t) \cap c(t')| \leqslant 2$).

**Safe partition reduction rule.** Let $R_T = (V, \mathcal{R}, T)$ be an embedded instance of $k$-DENSE RTI. We say that $\mathcal{P} = \{T_1, \ldots, T_l\}$ is a *tree partition* of $V$ if there exist $l$ nodes and leaves $x_1, \ldots, x_l$ of $T$ such that: $(i)$ for every $i \in [l]$ $T_i = T_{x_i}$ and $(ii)$ the set of leaves in $\cup_{i=1}^{l} T_{x_i}$ partition $V$. A tree partition of $R_T$ naturally distinguishes two sets of rooted triplets: $\mathcal{R}_I = \{t \in \mathcal{R} \mid \exists i \in [l]\ V(t) \subseteq V(T_i)\}$ and $\mathcal{R}_O = \mathcal{R} \setminus \mathcal{R}_I$. Let us call a rooted triplet of $\mathcal{R}_O$, an *outer triplet*.

**Definition 3.3** *Let $R_T = (V, \mathcal{R}, T)$ be an embedded instance of $k$-*DENSE RTI *and $\mathcal{P} = \{T_1, \ldots, T_l\}$ a tree partition of $R_T$ such that $\mathcal{R}_O$ contains at least one triplet inconsistent with $T$. Then $\mathcal{P}$ is a* safe partition *if it is possible to certify the rooted triplets of $\mathcal{R}_O$ inconsistent with $T$ only with rooted triplets of $\mathcal{R}_O$.*

We show that it is possible to reduce any embedded instance which has a safe partition.

**Rule 4 (safe partition)** *Let $R_T = (V, \mathcal{R}, T)$ be an embedded instance of $k$-*DENSE RTI *and $\mathcal{P}$ be a safe partition of $T$ with $\mathcal{F}$ the set of rooted triplets of $\mathcal{R}_O$ inconsistent with $T$. Edit every rooted triplet $t \in \mathcal{F}$ w.r.t. $T$ and decrease $k$ by $|\mathcal{F}|$.*

**Lemma 3.4** *The safe partition rule (Rule 4) is sound.*

**Proof**. We use the following observation, which follows from the definition of a tree partition:

**Observation 3.5** *Let $\mathcal{P} = \{T_1, \ldots, T_l\}$ be a tree partition of an embedded instance $R_T = (V, \mathcal{R}, T)$. Let $t$ be a rooted triplet such that $V(t) \subseteq V(T_i)$ for some $1 \leqslant i \leqslant l$, and $l \in V \setminus V(T_i)$. Then $l \notin span(t)$.*

As $\mathcal{P}$ is a safe partition, there exists a set $c(\mathcal{F})$ of triplet-disjoint certificates for $\mathcal{F}$. By construction of $c(\mathcal{F})$, at least one edition has to be done for every such certificate. We prove that $R_T$ can be made consistent by editing $k$ of its triplets iff the instance $R'$ obtained by editing every triplet of $\mathcal{F}$ w.r.t. $T$ can be made consistent by editing $k - |\mathcal{F}|$ of its triplets. Assume that the rooted triplets of $\mathcal{F}$ have been edited as described in Rule 4. Let $C = \{a, b, c, d\}$ be a conflict of the resulting instance and let $t$ be a rooted triplet of $C$ inconsistent with $T$ (suppose that $V(t) = \{a, b, c\}$). Clearly, as $t \notin \mathcal{F}$, we have $t \in \mathcal{R}_I$ and thus $\{a, b, c\}$ is a subset of leaves of some subtree $T_i$ (with $i \in [l]$). Moreover, by Lemma 3.2, we have $d \in span(t)$, since otherwise $d$ would not belong to any inconsistent rooted triplet in $R_T[C]$. It follows that $d$ is also a leaf of $T_i$, implying that every conflict is a subset of leaves of some tree $T_i$ of $\mathcal{P}$.

To conclude it suffices to observe that the edition of a rooted triplet of $\mathcal{R}_I$ cannot create a conflict involving a rooted triplet of $\mathcal{R}_O$. Let $t$ be a rooted triplet such that $V(t) = \{a, b, c\}$ is a subset of leaves of $T_i$. Let $d$ be a leaf not in $T_i$. By Observation 3.5, $d \notin span(t)$ and the three rooted triplets of $\mathcal{R}[\{a, b, c, d\}]$ involving $d$ are consistent with $T$. By Lemma 3.2, $\{a, b, c, d\}$ is not a conflict for any choice of $t$. This means that there exists an edition of $R_T$ that contains the triplets of $\mathcal{F}$. Hence $R_T$ has can be made consistent by editing $k$ of its triplets iff $R'$ can be made consistent by editing $k - |\mathcal{F}|$ of its triplets. □

**Conflict packing.** The remaining problem is now to either compute in polynomial time a safe partition if one exists or bound the size of the instance with respect to $k$. To that end, we use the conflict packing technique as for $k$-FAST. Observe that the aim of a conflict packing is to provide a lower bound on the number of editions required to obtain a consistent instance. In the context of $k$-DENSE RTI, two conflicts may share a rooted triplet $t$ but still require two distinct editions. To see this, let $\{a, b, c, d, e\}$ be a set of leaves and consider the following conflicts: $C = \{ab|c, ac|d, ad|b, cd|b\}$ and $C' = \{ed|c, ed|b, bc|e, bd|c\}$. Observe first that $C$ remains a conflict for any choice of $\{b, c, d\}$. Since $C$ and $C'$ only have this rooted triplet in common, no edition on $C'$ can solve $C$. Hence (at least) two distinct editions are require to solve both $C$ and $C'$. Due to

this remark, we refine our definition of conflict packing as follows. A leaf $a$ belonging to a conflict $\{a, b, c, d\}$ is a *seed* if $\{a, b, c, d\}$ is a conflict for any choice of $\{b, c, d\}$. A *conflict packing* is a maximal sequence of conflicts $\mathcal{C} = \{C_1, C_2, \ldots, C_l\}$ such that for every $2 \leq i \leq l$:

- Either $C_i$ intersects $\cup_{1 \leqslant j < i} C_j$ on at most two leaves,
- Or $C_i$ has a unique leaf not belonging to $\cup_{1 \leqslant j < i} C_j$ and that leaf is a seed of $C_i$.

As in Section 2 we will use a conflict packing $\mathcal{C}$ to compute a safe partition $\mathcal{P}$ (providing that $V \setminus V(\mathcal{C})$ is large enough w.r.t. parameter $k$).

**Lemma 3.6** *Let $R = (V, \mathcal{R})$ be a positive instance of $k$-*DENSE RTI *and $\mathcal{C}$ be a conflict packing of $R$. Then $l \leq k$ and $|V(\mathcal{C})| \leq 4k$.*

**Proof**. Assume $\mathcal{C} = \{C_1, \ldots, C_l\}$. We prove by induction on the number of conflicts $l$ contained in $\mathcal{C}$ that at least $l$ editions are necessary to solve all conflicts of $\mathcal{C}$. If $l = 1$ then the result trivially holds. Otherwise, we know that $l - 1$ editions are necessary to solve all conflicts of $\{C_1, \ldots, C_{l-1}\}$. If $C_l$ intersects the union of $C_j$, $1 \leqslant j \leqslant l-1$ in at most two leaves, then one clearly needs to do an extra edition for $C_l$. Otherwise, we know by definition that the unique leaf of $C_l$ not belonging to $\cup_{1 \leqslant j < i} C_j$ is a seed of $C_l$: hence none of the editions made to solve the conflicts of $\{C_1, \ldots, C_{l-1}\}$ can solve $C_l$. Hence $l \leqslant k$; since the conflicts involve four leaves, we also have $|V(\mathcal{C})| \leqslant 4k$. □

**Lemma 3.7 (Conflict packing)** *Let $R = (V, \mathcal{R})$ be an instance of $k$-*DENSE RTI *and $\mathcal{C}$ a conflict packing of $R$. There exists an embedded tree $T$ of $R$ such that every rooted triplet $t$ inconsistent with $T$ is such that $V(t) \subseteq V(\mathcal{C})$.*

**Proof**. The leaves of $G = V \setminus V(\mathcal{C})$ are called *good* leaves. As already observed $R[G]$ is consistent with a unique tree $T'$. Notice that for every leaf $a \in V(\mathcal{C})$, $R_a = R[G \cup \{a\}]$ is also consistent (otherwise $\mathcal{C}$ would not be maximal). Thereby there exists a unique binary tree $T_a$ such that every rooted triplet $t$ of $R_a$ is consistent with $T_a$. In other words $T'$ contains a unique tree edge $e = xz$ which can be subdivided into $xyz$ to attach the leaf $a$ to node $y$. Hereafter the edge $e$ will be called the *locus* of $a$. The maximality argument on $\mathcal{C}$ also implies that for any pair of leaves $a$ and $b$ in $V(\mathcal{C})$, $R_{ab} = R[G \cup \{a, b\}]$ is consistent. If $a$ and $b$ have different loci, then $R_{ab}$ is clearly consistent with the tree $T$ obtained from $T'$ by inserting $a$ and $b$ in their respective loci. It remains to consider the case where $a$ and $b$ have the same locus.

Let $e = xy$ be a tree edge of $T'$ such that $x$ is the child of $y$ and let $B_e \subseteq V(\mathcal{C})$ be the subset of leaves whose locus is $e$. Given $a, b \in B_e$, we define the following binary relations $<_e$ and $\sim_e$ on $B_e$ as follows:

$$a <_e b \text{ if there exists } c \in G \text{ such that } ac|b \in \mathcal{R}$$

$$a \sim_e b \text{ if neither } a <_e b \text{ nor } b <_e a$$

Using the maximality of $\mathcal{C}$ we will prove that $<_e$ is a strict weak ordering (i.e. $<_e$ is a transitive and asymmetric relation and $\sim_e$ is transitive). This implies that the equivalence classes of $\sim_e$ partition the leaves of $B_e$ and are totally ordered by $<_e$.

**Claim 2** *The relation $<_e$ is a strict weak ordering.*

*Proof.* Observe first that if $a <_e b$ then the vertex $c$ belongs to $T'_x$ (otherwise since $R_{ab}$ is consistent we would have $ab|c \in \mathcal{R}$). Assume $<_e$ is not asymmetric. Then there exist two leaves $c \in G \cap T'_x$ and $d \in G \cap T'_x$ such that $\{ca|b, db|a, cd|a, cd|b\} = \mathcal{R}[\{a, b, c, d\}]$. Thereby $\{a, b, c, d\}$ is a conflict: contradicting the fact that $R_{ab}$ is consistent for every $a, b \in B_e$. So $<_e$ is asymmetric.

Suppose we have $a, b, c \in B_e$ such that $a <_e b$ and $b <_e c$. So there exists $d$ such that $ad|b \in \mathcal{R}$. Since $d$ is a leaf of $T_x$ and $<_e$ is asymmetric, we also have $bd|c \in \mathcal{R}$. Now assume that $dc|a \in \mathcal{R}$ (the case $ac|d$ is similar). Then whatever the rooted triplet on $\{a, b, c\}$ is, $\{a, b, c, d\}$ is a conflict. Hence $d$ is a seed of the conflict $\{a, b, c, d\}$. This means that the conflict packing $\mathcal{C}$ is not maximal: contradiction. If follows that $da|c \in \mathcal{R}$ and thereby $a <_e c$. So $<_e$ is transitive.

Suppose we have $a, b, c \in B_e$ such that $a \sim_e b$ and $b \sim_e c$. Then for every $d \in G$, $ab|d$ and $bc|d$ are rooted triplets of $\mathcal{R}$. Now assume that $ad|c \in \mathcal{R}$ (the case $dc|a$ is similar). Then whatever the rooted triplet on $\{a, b, c\}$ is, $\{a, b, c, d\}$ is a conflict. Hence $d$ is a seed of the conflict $\{a, b, c, d\}$. This means that the conflict packing $\mathcal{C}$ is not maximal: contradiction. If follows that neither $ad|c$ nor $cd|a$ belong to $\mathcal{R}$: thereby $a \sim_e c$ and $\sim_e$ is transitive. ⋄

We can now describe how the tree $T$ is build from $T'$. For every tree edge $e = xy$ with $x$ a child of $y$ such that $B_e \neq \emptyset$ we proceed as follows. Let $B_1 \dots B_q$ be the equivalence classes of $\sim_e$ such that $B_i <_e B_j$ for $1 \leqslant i < j \leqslant q$. The tree edge $e$ is subdivided into the path $x, z_1 \dots, z_q, y$. For every $i \in [q]$, if $B_i$ contains a unique leaf $a$, then $a$ is attached to node $z_i$. Otherwise, a new node $w_i$ is attached to $z_i$ and we add an arbitrary binary tree (rooted in $w_i$) over the leaves of $B_i$. We now prove that $T$ has the desired property. Let $t = \{a, b, c\}$ be any triplet of $\mathcal{R}$, and assume first that $V(t) \subseteq G$. Then $t$ is consistent by construction. Next, assume w.l.o.g. that $V(t) \cap V(\mathcal{C}) = \{a\}$: then $t$ is consistent with $T$ since $R_a$ is consistent and $a$ has been inserted to its locus. Finally, assume $V(t) \cap V(\mathcal{C}) = \{a, b\}$. If $a$ and $b$ have different loci then $t$ is clearly consistent with $T$. Now, if $a$ and $b$ have the same locus $e$ then $t$ is consistent since $a$ and $b$ have been added to $e$ according to the strict weak ordering $<_e$. It follows that any triplet of $T$ such that $V(t) \cap G \neq \emptyset$ is consistent with $T$. □

We now state the main result of this section.

**Theorem 3.8** *The $k$-*DENSE RTI *problem admits a kernel with at most $5k$ vertices.*

**Proof**. Let $R = (V, \mathcal{R})$ be a positive instance of $k$-DENSE RTI reduced under Rule 3. We say that a tree $T$ obtained through Lemma 3.7 is a *nice* tree. We greedily (hence in polynomial time) compute a conflict packing $\mathcal{C}$ and let $T$ be a nice tree. Consider the bipartite graph $B = (I \cup G, E)$ where:

- $G = V \setminus V(\mathcal{C})$,
- there is a vertex $i_t$ in $I$ for every rooted triplet $t$ inconsistent with $T$ and,
- $i_t g \in E$ if $a \in G$ and $\{a\} \cup V(t)$ is a certificate of $t$.

Observe that any matching in $B$ of size at least $k + 1$ would correspond to a conflict packing of size at least $k + 1$, which cannot be. Hence a minimum vertex cover $D$ of $B$ has size at most $k$ [8]. We denote $D_1 = D \cap I$ and $D_2 = D \cap G$.

Assume that $|V| > 5k$. Then since $|D_2| \leqslant k$ and $|V(\mathcal{C})| \leqslant 4k$ (by Lemma 3.6), $G \setminus D_2 \neq \emptyset$. Let $\mathcal{P} = \{T_1, \dots, T_l\}$ be a tree partition of $R_T$ such that every tree $T_i$, $i \in [l]$, consists of either

a leaf of $G \setminus D_2$ or a connected component of $T \setminus S$ where $S$ is the smallest spanning subtree of $(G \setminus D_2) \cup \{r\}$ ($r$ being the root of $T$, see Figure 2).

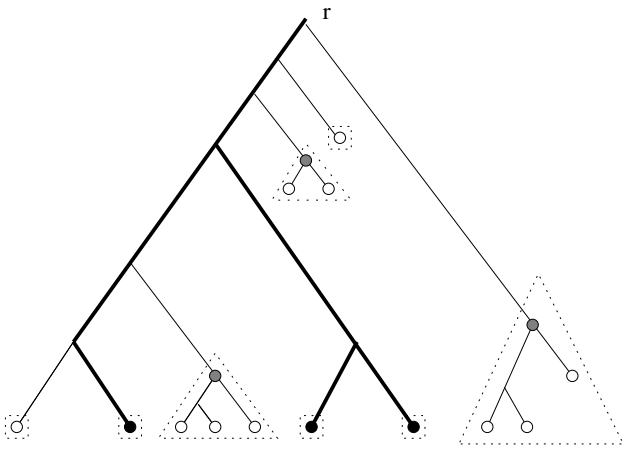


Figure 2: Illustration of the construction of $\mathcal{P}$. The black vertices belong to $G \setminus D_2$ and the bold edges represent $S$. The partition $\mathcal{P}$ is pictured by the dotted sets, while its associated nodes $u$ are in grey.

**Claim 3** *The partition $\mathcal{P} = \{T_1, \ldots, T_l\}$ is safe.*

*Proof.* Let $a$ be any leaf of $G \setminus D_2$. By Lemma 3.7, $a$ is not contained in any rooted triplet inconsistent with $T$. As $R$ is reduced under Rule 3, there exists an inconsistent rooted triplet $t$ such that $a \in span(t)$. It follows that $\mathcal{R}_O$ contains at least one inconsistent rooted triplet.

Let $t \in \mathcal{R}_O$ be a rooted triplet inconsistent with $T$. By construction of $\mathcal{P}$, there exists a leaf $a \in (G \setminus D_2) \cap span(t)$. Then $\{a\} \cup V(t)$ is a certificate of $t$ and $i_t a$ is an edge of $B$. Observe that as $D$ is a vertex cover and $a \notin D_2$, the vertex $i_t$ has to belong to $D_1$ to cover the edge $i_t a$. Thereby the subset $I' \subseteq I$ corresponding to the rooted triplets of $\mathcal{R}_O$ inconsistent with $T$ is included in $D_1$.

Finally, we argue that $I'$ can be matched into $G \setminus D_2$ in $B$. Assume there exists $I'' \subseteq I'$ such that $|I''| > |N(I'') \cap (G \setminus D_2)|$. As there is no edge in $B$ between $I \setminus D_1$ and $G \setminus D_2$ ($D$ is a vertex cover of $B$), the set $D' = (D \setminus I'') \cup (N(I'') \cap (G \setminus D_2))$ is a vertex cover of $B$ and $|D'| < |D|$: contradicting the minimality of $D$.

Thereby for every subset $I'' \subseteq I'$, we have $|I''| \leqslant |N(I'') \cap (G \setminus D_2)|$. By Hall's theorem [20], $I'$ can be matched into $G \setminus D_2$. As every leaf of $G \setminus D_2$ is a singleton in $\mathcal{P}$, the existence of the matching shows that the set of rooted triplets of $\mathcal{R}_O$ inconsistent with $T$ can be certified using rooted triplet of $\mathcal{R}_O$ only, and hence $\mathcal{P}$ is safe. ◇

Hence if $|V| > 5k$, there exists a safe partition that can be computed in polynomial time, and we can reduce the instance using Rule 4. We then apply Rule 3 and repeat the previous steps until we either do not find a safe partition or $k < 0$. In the former case we know that $|V| \leqslant 5k$; in the latter case, we return a small trivial No-instance. This concludes the proof. □

## 4 Linear vertex-kernel for $k$-BetweennessTour

**Preliminaries.** A *betweenness triplet* $t$ defined over a set of three vertices $\{a, b, c\}$ *chooses* one of its vertices. We write $t = abc$ to illustrate that $t$ chooses $b$. An instance of $k$-BetweennessTour is a pair $B = (V, \mathcal{R})$ where $\mathcal{R}$ is a set of betweenness triplets defined over $V$. We only consider *dense* instances, that is $\mathcal{R}$ contains exactly one triplet for every triple of $V$. Let $t = abc \in \mathcal{R}$ be a betweenness triplet (or triplet for short) and $\sigma$ be an ordering on $V$. Then $t$ is *consistent* with $\sigma$ if $a <_\sigma b <_\sigma c$ or $c <_\sigma b <_\sigma a$. A set of triplets $\mathcal{R}$ is *consistent* if there exists an ordering $\sigma$ on $V$ such that every $t \in \mathcal{R}$ is consistent with $\sigma$. If such an ordering does not exist, then $\mathcal{R}$ is *inconsistent*. A conflict $C$ is a subset of $V$ such that $\mathcal{R}[C]$ is inconsistent. Given an instance $B = (V, \mathcal{R})$, an *edition* for a triplet $t \in \mathcal{R}$ is a modification of its choosen vertex. A set $\mathcal{F}$ of edited triplets of $\mathcal{R}$ is an *edition* for $B$ if the edition of every $t \in \mathcal{R}$ leads to a consistent instance. We use $B_\sigma = (V, \mathcal{R}, \sigma)$ to denote an instance of $k$-BetweennessTour fixed under some ordering $\sigma$. For a subset $S \subseteq V$, we define $\mathcal{R}[S] = \{t \in \mathcal{R} \mid V(t) \subseteq S\}$ and $B_\sigma[S] = (S, \mathcal{R}[S], \sigma_{|S})$ (*i.e.* the ordering $\sigma$ restricted to elements of $S$). Finally, given an ordered instance $B_\sigma = (V, \mathcal{R}, \sigma)$ and a triplet $t = \{a, b, c\}$ such that $a <_\sigma b <_\sigma c$, *editing* $t$ w.r.t. $\sigma$ means that the choice of $t$ is set to $b$. When dealing with an ordered instance the inconsistency of a triplet is always considered w.r.t. $\sigma$.

**Lemma 4.1** *Let $B = (V, \mathcal{R})$ be an instance of $k$-BetweennessTour. Then $B$ is consistent if and only if $B$ does not contain any conflict on 4 vertices.*

**Proof**. ($\Rightarrow$) This direction follows by definition of consistency.
($\Leftarrow$) The proof is by induction on the number of vertices. Observe that for $n = 4$ the statement clearly holds. Assume that the statement holds for $n-1$ vertices and let $B = (V, \mathcal{R})$ be an instance of $k$-BetweennessTour on $n$ vertices that does not contain any conflict on 4 vertices. Let $d \in V$ be any vertex; by induction hypothesis, we know that $B_\sigma = B[V \setminus \{d\}]$ admits a (unique) consistent ordering $\sigma$. We will prove how to *insert* $d$ in $\sigma$ to obtain a consistent ordering $\sigma_d$ for $B$. Let $a, b, c$ be respectively the first, second and last vertex of $\sigma$. Since $B_\sigma$ is consistent, $abc \in \mathcal{R}$. Moreover, since there are no conflicts on 4 vertices there is a unique way to insert $d$ in $\sigma$ to obtain an ordering $\sigma_d$ such that $\mathcal{R}[\{a, b, c, d\}]$ is consistent with $\sigma_d$. There are several cases to consider: either $d <_{\sigma_d} a$ (or $c <_{\sigma_d} d$) or $a <_{\sigma_d} d <_{\sigma_d} b$ or $b <_{\sigma_d} d <_{\sigma_d} c$. In the latter case we repeat the previous steps on $\sigma_{|V\setminus\{a\}}$ (*i.e.* the ordering $\sigma$ restraint to $V \setminus \{a\}$).

Let $a'b'c'$ be the last triplet considered in the process and assume that $d$ is between $a'$ and $b'$ in $\sigma_d$ (the other cases are similar). We will prove that $\sigma_d$ is a consistent ordering of $B$. Let $t$ be a triplet containing $d$ (observe that the other triplets are consistent with $\sigma_d$ by construction). Assume first w.l.o.g. that $V(t) = \{a', d, e\}$ with $e \notin \{a', b', c'\}$. Observe that the triplets $\{a', b', d\}$ and $\{a', b', e\}$ are consistent with $\sigma_d$ by construction. This means that $a'db', \{ea'b' \vee a'b'e\} \in \mathcal{R}$: it follows that $t$ is consistent with $\sigma_d$, since otherwise $\{a', b', d, e\}$ would be a conflict on 4 vertices. Now, assume w.l.o.g. $V(t) = \{d, e, f\}$ with $\{e, f\} \notin \{a', b', c'\}$. By the previous arguments we know that the triplets $\{a', d, e\}$, $\{a', d, f\}$ and $\{a', e, f\}$ are consistent with $\sigma_d$. It follows that $t$ is consistent with $\sigma_d$ since otherwise $\{a', d, e, f\}$ would be a conflict on 4 vertices. In all cases we have shown that $\sigma_d$ does not contain any inconsistent triplet, implying that $B$ is consistent. □

**Sunflower reduction rule.** The main difference with the previous section lies in the definition of *simple conflict* for an ordered instance $B_\sigma = (V, \mathcal{R}, \sigma)$: given an inconsistent triplet $t$ and a vertex $d$

that does not belong to any inconsistent triplet, we do not need to require that $d$ belongs to $span(t)$ to obtain a conflict. As indicated by Lemma 4.3, *any* such vertex can be used to form a conflict with $V(t)$. In particular, this result allows us to replace the *Safe Partition* rule with a *Sunflower*-based reduction rule. A *sunflower* $\mathcal{S}$ is a set of conflicts $\{C_1, \ldots, C_m\}$ pairwise intersecting in exactly one triplet $t$. We say that $t$ is the *centre* of $\mathcal{S}$.

**Lemma 4.2 (Sunflower Lemma)** *Let $B = (V, \mathcal{R})$ be an instance of $k$-BetweennessTour. Let $\mathcal{S} = \{C_1, \ldots, C_m\}$, $m > k$ be a sunflower of centre $t$. Any edition of size at most $k$ has to edit $t$.*

**Proof**. Let $\mathcal{F}$ be any edition of size at most $k$, and assume that $\mathcal{F}$ does not contain $t$. This means that $\mathcal{F}$ must contain one triplet for every conflict $C_i$, $1 \leq i \leq m$. Since $m > k$, we conclude that $\mathcal{F}$ contains more than $k$ triplets, a contradiction. □

Observe that there exist two ways to edit the centre $t$. By setting $m > 2k$ we can fix this and obtain a quadratic vertex-kernel for $k$-BetweennessTour (by adapting techniques from [19]). Nevertheless, this is not enough to obtain a *linear* vertex kernel. To that aim, we combine Conflict Packing and a sunflower rule on *simple conflicts*.

**Lemma 4.3** *Let $B_\sigma = (V, \mathcal{R}, \sigma)$ be an ordered instance of $k$-BetweennessTour. Let $\{a, b, c, d\}$ be a set of vertices such that $bca \in \mathcal{R}$ is the only inconsistent triplet of $B_\sigma[\{a, b, c, d\}]$. Then $\{a, b, c, d\}$ is a conflict.*

**Proof**. Since $bca \in \mathcal{R}$ is inconsistent with $\sigma$, $c$ is not between $a$ and $b$ in $\sigma$. Hence we either have $b$ between $a$ and $c$ or $a$ between $b$ and $c$. Assume w.l.og. that $a <_\sigma b <_\sigma c$ or $b <_\sigma a <_\sigma c$. The two cases are similar, so assume the former holds. By assumption, the triplet $t \in \mathcal{R}$ on $\{b, c, d\}$ is consistent with $\sigma$. Consider the three possible cases for $t$:

- $t = bcd$: observe that if $t$ and $bca$ are consistent with an ordering $\rho$, then w.l.o.g. $c <_\rho d <_\rho a$ or $c <_\rho a <_\rho d$ holds (the cases $a <_\rho d <_\rho c$ and $d <_\rho a <_\rho c$ are symmetric). On the other hand, if $t$ and $abc$ are consistent with another ordering $\gamma$, then $a <_\gamma c <_\gamma d$ holds.
- $t = cbd$: observe that if $t$ and $bca$ are consistent with an ordering $\rho$, then $d <_\rho c <_\rho a$ holds. On the other hand, if $t$ and $abc$ are consistent with another ordering $\gamma$, then $c <_\gamma d <_\gamma a$ or $c <_\gamma a <_\gamma d$ holds.
- $t = bdc$: observe that if $t$ and $bca$ are consistent with an ordering $\rho$, then $d <_\rho c <_\rho a$ holds. On the other hand, if $t$ and $abc$ are consistent with another ordering $\gamma$, then $a <_\gamma d <_\gamma c$ holds.

Since $t$ and $abc$ are consistent with $\sigma$, it follows that $\{a, b, c, d\}$ is a conflict for any choice of $\{b, c, d\}$. This concludes the proof. □

Given an ordered instance $B_\sigma = (V, \mathcal{R}, \sigma)$, a triplet $t = \{a, b, c\}$ inconsistent with $\sigma$ and a vertex $d$ that does not belong to any inconsistent triplet, the set $V(t) \cup \{d\}$ is called a *simple conflict*. A sunflower $\mathcal{S} = \{C_1, \ldots, C_m\}$ of $B_\sigma$ is *simple* if the $C_i$'s are simple conflicts and if the centre of $\mathcal{S}$ is the only triplet inconsistent with $\sigma$ for every $C_i$, $1 \leqslant i \leqslant m$. The soundness of the Simple sunflower rule follows from Lemmas 4.2 and 4.3.

**Rule 5 (Simple sunflower)** *Let $B_\sigma = (V, \mathcal{R}, \sigma)$ be an ordered instance of $k$-BetweennessTour. Let $\mathcal{S} = \{C_1, \ldots, C_m\}$, $m > k$ be a simple sunflower of centre $t$. Edit $t$ w.r.t. $\sigma$ and decrease $k$ by 1.*

**Proof**. Let $\mathcal{F}$ be any edition of size at most $k$: by Lemma 4.2, $\mathcal{F}$ must contain $t$. Since $|\mathcal{F}| \leqslant k$, there exists a conflict $C_i$ whose only edited triplet is $t$. Assume that $t$ was not edited w.r.t. $\sigma$: since no other triplet has been edited in $C_i$ it still contains only one inconsistent triplet in $B_\sigma[C_i]$. By Lemma 4.3, it follows that $C_i$ induces a conflict, contradicting the fact that $\mathcal{F}$ is an edition. □

**Conflict Packing.** The definition of a conflict packing $\mathcal{C}$, involving the notion of *seed*, is the same than the one given in Section 3. Given a conflict packing $\mathcal{C}$, we can compute in polynomial time an ordering $\sigma$ such that any triplet $t$ inconsistent with $\sigma$ verifies $V(t) \subseteq V(\mathcal{C})$. The following result is similar to Lemma 3.6.

**Lemma 4.4** *Let $B = (V, \mathcal{R})$ be a positive instance of $k$-BetweennessTour and $\mathcal{C}$ be a conflict packing of $T$. Then $|V(\mathcal{C})| \leq 4k$.*

**Lemma 4.5 (Conflict Packing)** *Let $B = (V, \mathcal{R})$ be an instance of $k$-BetweennessTour and $\mathcal{C}$ a conflict packing of $B$. There exists an ordering whose inconsistent triplets $\{a, b, c\}$ are such that $a, b, c \in V(\mathcal{C})$.*

**Proof**. Let $G = V \setminus V(\mathcal{C})$. Observe that $B' = B[G]$ is consistent with a (unique) ordering $\sigma$. Moreover, notice that for every vertex $a \in V(\mathcal{C})$ the instance $B_a = B[G \cup \{a\}]$ is consistent (otherwise $\mathcal{C}$ would not be maximal). Thereby there exists a unique ordering $\sigma_a$ such that every triplet of $B_a$ is consistent with $\sigma_a$. In other words, $\sigma$ contains a unique pair of consecutive vertices $(u, w)$ such that we have $u <_{\sigma_a} < a <_{\sigma_a} w$. We wall such a pair $(u, w)$ the *locus* of $a$. Using again the maximality argument on $\mathcal{C}$, we know that for any pair of vertices $a, b \in V(\mathcal{C})$ the instance $B_{ab} = B[G \cup \{a, b\}]$ is consistent. Hence if $a$ and $b$ have different loci then $B_{ab}$ is clearly consistent with the ordering obtained from $\sigma$ by inserting $a$ and $b$ in their respective loci. It remains to consider the case where $a$ and $b$ have the same locus. Given two consecutive vertices $u$ and $w$ of $\sigma$, let $B_{uw} \subseteq V(\mathcal{C})$ be the subset of vertices of $V(\mathcal{C})$ whose locus is $uw$. Given $a, b \in B_{uw}$ we define the binary relation $<_{uw}$ on $B_{uw}$ as follows (observe that there are exactly two ways to add $a$ and $b$ w.r.t. their locus):

$$a <_{uw} b \text{ if } abw \in \mathcal{R}$$

**Claim 4** *The relation $<_{uw}$ is a strict total order (*i.e. *asymmetric and transitive).*

*Proof.* Observe that $<_{uw}$ is asymmetric by definition. Hence it remains to prove that $<_{uw}$ is transitive. Suppose that $a, b, c \in B_{uw}$ are such that $a <_{uw} b$ and $b <_{uw} c$. This means that $abw \in \mathcal{R}$ and $bcw \in \mathcal{R}$. Now assume that $acw \notin \mathcal{R}$, *i.e.* w.l.o.g. that $wac \in \mathcal{R}$. Then $\{a, b, c, w\}$ is a conflict for any choice of $\{a, b, c\}$. Moreover, observe that $w$ is a seed of the conflict $\{a, b, c, w\}$, meaning that the conflict packing $\mathcal{C}$ is not maximal: contradiction. It follows that $acw \in \mathcal{R}$ and thereby $a <_{uw} c$. Hence $<_{uw}$ is transitive. ◇

We now describe how the ordering $\sigma'$ is built from $\sigma$. For any two consecutive vertices $u$ and $w$ such that $B_{uw} \neq \emptyset$ we introduce the vertices of $B_{uw}$ according to the (strict) total order $<_{uw}$.

We now prove that $\sigma'$ satisfies the desired property. Let $t = \{a, b, c\}$ be any triplet of $\mathcal{R}$ such that $V(t) \cap G \neq \emptyset$. Observe that if $t$ is such that $V(t) \subseteq G$ then $t$ is consistent by construction. Next, if $|V(t) \cap G| = 2$ then $t$ is consistent by construction. Finally, if $|V(t) \cap G| = 1$ with $a, b \in V(\mathcal{C})$, then $t$ is consistent since $B_{ab}$ is consistent. It follows that $\sigma'$ is the sought ordering. □

Using Lemma 4.5 and the Simple sunflower rule, we prove that the $k$-BetweennessTour problem can be solved in polynomial time on instances whose parameter is such that $k < (|V|/5)$.

**Theorem 4.6** *Let $B = (V, \mathcal{R})$ be an instance of $k$-BetweennessTour such that $k < (|V|/5)$. There exists an algorithm that either computes an edition of size at most $k$ or answers* No *in polynomial time.*

**Proof**. Let $B = (V, \mathcal{R})$ be an instance of $k$-BetweennessTour. We say that an ordering obtained through Lemma 4.5 is a *nice* ordering. We greedily compute a conflict packing $\mathcal{C}$ of $B$ and let $\sigma$ be a nice ordering of $B$. Observe that $|V| > 5k$ (since we assume $k < (|V|/5)$) and let $G = V \setminus V(\mathcal{C})$. We know that $|V(\mathcal{C})| \leqslant 4k$. Hence $G$ contains at least $k + 1$ vertices that do not belong to any triplet inconsistent with $\sigma$ by construction.

**Claim 5** *There exists a simple sunflower $\mathcal{S} = \{C_1, \ldots, C_m\}$, $m > k$ that can be computed in polynomial time.*

**Proof**. Let $t$ be any triplet inconsistent with $\sigma$ and $G' \subseteq G$ be a set of $k + 1$ vertices of $G$. By Lemma 4.3 we know that $C_d = V(t) \cup \{d\}$ is a simple conflict for any vertex $d \in G'$. Assuming $G' = \{d_1, \ldots, d_{k+1}\}$, it follows that $C_i = V(t) \cup \{d_i\}$ is a simple conflict whose only inconsistent triplet is $t$ for every $1 \leqslant i \leqslant k + 1$. Hence $\mathcal{S} = \{C_1, \ldots, C_{k+1}\}$ is a simple sunflower of centre $t$. □

Using Rule 5 we have to edit the centre $t \in \mathcal{R}$ of $\mathcal{S}$ w.r.t. to $\sigma$. Since $\sigma$ still contains at least $k+1$ vertices that do not belong to any inconsistent triplet, every inconsistent triplet of $\sigma$ must be edited. Hence if $\sigma$ contains at most $k$ inconsistent triplets then editing such triplets is an edition of $B$ and we answer Yes; otherwise we answer No. □

As a particular consequence of Theorem 4.6 we obtain the following result.

**Corollary 4.7** *The $k$-BetweennessTour problem admits a kernel with at most $5k$ vertices.*

**Proof**. Let $B = (V, \mathcal{R})$ be an instance of $k$-BetweennessTour. By Theorem 4.6, the problem can be solved in polynomial time if $k < (|V|/5)$. Hence we can assume $k \geqslant (|V|/5)$, which implies that $|V| \leqslant 5k$. □

## Conclusion

In this paper we develop a technique to design kernelization algorithms, namely *Conflict Packing*. In particular, we applied this technique to the $k$-FAST and $k$-dense RTI problems. Although a linear vertex-kernel was already known for $k$-FAST [6], our analysis gives more insights on the structure of this problem. Regarding the $k$-dense RTI problem, the Conflict Packing allows us to obtain a linear vertex-kernel, improving the previous bound of $O(k^2)$ vertices [19]. Moreover, we provide a linear

vertex-kernel for the $k$-BetweennessTour problem, answering a question left open in [22]. Such a kernel may improve the parameterized complexity of the $k$-BetweennessTour problem [22]. We defer this analysis to a full version of the paper. We conclude by addressing some open problems. First, observe that the simple sunflower rule together with a PTAS for $k$-BetweennessTour [21] implies the existence of a linear vertex-kernel for the problem (but using a more complicated algorithm). One important remaining question is thus whether the $k$-dense RTI problem admits a constant-factor approximation algorithm? We would like to mention that such an algorithm together with the safe partition reduction rule would also imply a linear vertex-kernel for the problem. Finally, there exist a large number of problems on dense instances (see e.g. [2, 21]): we believe that our technique will yield linear vertex-kernels for a number of these problems as well.

## References


[1] A. Aho, Y. Sagiv, T. Szymansk, and J. Ullman. Inferring a tree from lowest common ancestor with an application to the optimization of relational expressions. *SIAM Journal on Computing*, 10(3):405–421, 1981.

[2] Nir Ailon and Noga Alon. Hardness of fully dense problems. *Inf. Comput*, 205(8):1117–1129, 2007.

[3] N. Alon. Ranking tournaments. *SIAM J. Discrete Math.*, 20(1):137–142, 2006.

[4] N. Alon, D. Lokshtanov, and S. Saurabh. Fast FAST. In *ICALP*, volume 5555 of *LNCS*, pages 49–58. Springer, 2009.

[5] H.-J. Bandelt and A. Dress. Reconstructing the shape of a tree from observed dissimilarity data. *Advances in Applied Mathematics*, 7:309–343, 1986.

[6] S. Bessy, F. V. Fomin, S. Gaspers, C. Paul, A. Perez, S. Saurabh, and S. Thomassé. Kernels for feedback arc set in tournaments. In *FSTTCS*, volume 4 of *LIPIcs*, pages 37–47, 2009.

[7] H. L. Bodlaender. Kernelization: New upper and lower bound techniques. In *IWPEC*, pages 17–37, 2009.

[8] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications*. North Holland, 1976.

[9] D. Brügmann, C. Komusiewicz, and H. Moser. On generating triangle-free graphs. *Electronic Notes in Discrete Mathematics*, 32:51–58, 2009.

[10] J. Byrka, S. Guillemot, and J. Jansson. New results on optimizing rooted triplets consistency. *Discrete Applied Mathematics*, 158(11):1136–1147, 2010.

[11] P. Charbit, S. Thomassé, and A. Yeo. The minimum feedback arc set problem is NP-hard for tournaments. *Combin. Probab. Comput.*, 16(1):1–4, 2007.

[12] M. Dom, J. Guo, F. Hüffner, R. Niedermeier, and A. Truß. Fixed-parameter tractability results for feedback set problems in tournaments. In *CIAC*, volume 3998 of *LNCS*, pages 320–331, 2006.

[13] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.

[14] P. Erdös and J. W. Moon. On sets on consistent arcs in tournaments. *Canad. Math. Bull.*, 8:269–271, 1965.

[15] F. V. Fomin, D. Lokshtanov, S. Saurabh, and D. M. Thilikos. Bidimensionality and kernels. In *SODA*, pages 503–510, 2010.

[16] A.D. Gordon. Consensus super tree containing overlapping sets of labeled leaves. *Journal of Classification*, 3:31–39, 1986.

[17] J. Gramm, A. Nickelsen, and T. Tantau. Fixed-parameter algorithms in phylogenetics. *The Computer Journal*, 51(1):79–100, 2008.

[18] S. Guillemot and V. Berry. Fixed-parameter tractability of the maximum agreement supertree problem. *IEEE/ACM Trans. Comput. Biology Bioinform*, 7(2), 2010.

[19] S. Guillemot and M. Mnich. Kernel and fast algorithm for dense triplet inconsistency. In *TAMC*, volume 6108 of *Lecture Notes in Computer Science*, pages 247–257. Springer, 2010.

[20] P. Hall. On representatives of subsets. *J. London Math. Soc.*, 10(37):26–30, 1935.

[21] M. Karpinski and W. Schudy. Approximation schemes for the betweenness problem in tournaments and related ranking problems. *CoRR*, abs/0911.2214, 2009.

[22] M. Karpinski and W. Schudy. Faster algorithms for feedback arc set tournament, kemeny rank aggregation and betweenness tournament. In *ISAAC*, pages 3–14, 2010.

[23] C. Kenyon-Mathieu and W. Schudy. How to rank with few errors. In *STOC*, pages 95–103, 2007.

[24] V. Raman and S. Saurabh. Parameterized algorithms for feedback set problems and their duals in tournaments. *Theor. Comput. Sci*, 351(3):446–458, 2006.

[25] K. D. Reid and E. T. Parker. Disproof of a conjecture of Erdös and Moser on tournaments. *J. Combin. Theory*, 9:225–238, 1970.

[26] S. Saurabh. Chromatic coding and universal (hyper-)graph coloring families. *Parameterized Complexity News*, pages 49–58, June 2009.

[27] C. Semple and M. Steel. *Phylogenetics*, volume 24 of *Oxford Lecture Series in Mathematics and Its Applications*. Oxford University Press, 2003.

[28] S. Thomassé. A $4k^2$ kernel for feedback vertex set. *ACM Transactions on Algorithms*, 6(2), 2010.

[29] A. van Zuylen, R. Hegde, K. Jain, and D. P. Williamson. Deterministic pivoting algorithms for constrained ranking and clustering problems. In *SODA*, pages 405–414, 2007.

[30] J. Wang, D. Ning, Q. Feng, and J. Chen. An improved kernelization for $P_2$-packing. *Inf. Process. Lett*, 110(5):188–192, 2010.

# Polynomial kernels for Proper Interval Completion and related problems *

Stéphane Bessy Anthony Perez
bessy@lirmm.fr perez@lirmm.fr
LIRMM – Université Montpellier II - FRANCE

April 18, 2011

**Abstract**

Given a graph $G = (V, E)$ and a positive integer $k$, the Proper Interval Completion problem asks whether there exists a set $F$ of at most $k$ pairs of $(V \times V) \setminus E$ such that the graph $H = (V, E \cup F)$ is a proper interval graph. The Proper Interval Completion problem finds applications in molecular biology and genomic research [14, 22]. First announced by Kaplan, Tarjan and Shamir in FOCS '94, this problem is known to be FPT [14], but no polynomial kernel was known to exist. We settle this question by proving that Proper Interval Completion admits a kernel with at most $O(k^5)$ vertices. Moreover, we prove that a related problem, the so-called Bipartite Chain Deletion problem, admits a kernel with at most $O(k^2)$ vertices, completing a previous result of Guo [12].

## Introduction

The aim of a graph modification problem is to transform a given graph in order to get a certain property $\Pi$ satisfied. Several types of transformations can be considered: for instance, in *vertex deletion* problems, we are only allowed to delete vertices from the input graph, while in *edge modification problems* the only allowed operation is to modify the edge set of the input graph. The optimization version of such problems consists in finding a *minimum* set of edges (or vertices) whose modification makes the graph satisfy the given property $\Pi$. Graph modification problems cover a broad range of NP-Complete problems and have been extensively studied in the literature [18, 21, 22]. Well-known examples include the Vertex Cover [8], Feedback Vertex Set [24], or Cluster Editing [5] problems. These problems find applications in various domains, such as computational biology [14, 22], image processing [21] or relational databases [23].

Due to these applications, one may be interested in computing an *exact* solution for such problems. *Parameterized complexity* provides a useful theoretical framework to that aim [9, 19]. A problem *parameterized* by some integer $k$ is said to be *fixed-parameter tractable* (FPT for short) whenever it can be solved in time $f(k) \cdot n^c$ for any constant $c > 0$. A natural parameterization for graph modification problems thereby consists in the number of allowed transformations. As one of the most powerful technique to design fixed-parameter algorithms, *kernelization algorithms* have been extensively studied in the last decade (see [2] for a survey). A *kernelization algorithm* is a

*Research supported by the AGAPE project (ANR-09-BLAN-0159).

polynomial-time algorithm (called *reduction rules*) that given an instance $(I, k)$ of a parameterized problem $P$ computes an instance $(I', k')$ of $P$ such that $(i)$ $(I, k)$ is a YES-instance if and only if $(I', k')$ is a YES-instance and $(ii)$ $|I'| \leq h(k)$ for some computable function $h()$ and $k' \leq k$. The instance $(I', k')$ is called the *kernel* of $P$. We say that $(I', k')$ is a polynomial kernel if the function $h()$ is a polynomial. It is well-known that a parameterized problem is FPT if and only if it has a kernelization algorithm [19]. But this equivalence only yields kernels of super-polynomial size. To design efficient fixed-parameter algorithms, a kernel of small size - polynomial (or even linear) in $k$ - is highly desirable [20]. However, recent results give evidence that not every parameterized problem admits a polynomial kernel, unless $NP \subseteq coNP/poly$ [3]. On the positive side, notable kernelization results include a less-than-$2k$ kernel for VERTEX COVER [8], a $4k^2$ kernel for FEEDBACK VERTEX SET [24] and a $2k$ kernel for CLUSTER EDITING [5].

We follow this line of research with respect to graph modification problems. It has been shown that a graph modification problem is FPT whenever $\Pi$ is hereditary and can be characterized by a finite set of forbidden induced subgraphs [4]. However, recent results proved that several graph modification problems do not admit a polynomial kernel even for such properties $\Pi$ [11, 16]. In this paper, we are in particular interested in *completion* problems, where the only allowed operation is to add edges to the input graph. We consider the property $\Pi$ as being the class of *proper interval graphs*. This class is a well-studied class of graphs, and several characterizations are known to exist [17, 28]. In particular, there exists an *infinite* set of forbidden induced subgraphs that characterizes proper interval graphs [28] (see Figure 1). More formally, we consider the following problem:

PROPER INTERVAL COMPLETION:
**Input**: A graph $G = (V, E)$ and a positive integer $k$.
**Parameter**: $k$.
**Output**: A set $F$ of at most $k$ pairs of $(V \times V) \setminus E$ such that the graph $H = (V, E \cup F)$ is a proper interval graph.

Interval completion problems find applications in molecular biology and genomic research [13, 14], and in particular in *physical mapping* of DNA. In this case, one is given a set of long contiguous intervals (called *clones*) together with experimental information on their pairwise overlaps, and the goal is to reconstruct the relative position of the clones along the target DNA molecule. We focus here on the particular case where all intervals have equal length, which is a biologically important case (e.g. for cosmid clones [13]). In the presence of (a small number of) unidentified overlaps, the problem becomes equivalent to the PROPER INTERVAL COMPLETION problem. It is known to be NP-Complete for a long time [10], but fixed-parameter tractable due to a result of Kaplan, Tarjan and Shamir in FOCS '94 [14, 15]. [1] The fixed-parameter tractability of the PROPER INTERVAL COMPLETION can also be seen as a corollary of a characterization of Wegner [28] combined with Cai's result [4]. Nevertheless, it was not known whether this problem admit a polynomial kernel or not.

**Our results** We prove that the PROPER INTERVAL COMPLETION problem admits a kernel with at most $O(k^5)$ vertices. To that aim, we identify *nice* parts of the graph that induce proper interval graphs and can hence be safely reduced. Moreover, we apply our techniques to the so-called BIPARTITE CHAIN DELETION problem, closely related to the PROPER INTERVAL COMPLETION

[1] Notice also that the *vertex deletion* of the problem is fixed-parameter tractable [26].

problem where one is given a graph $G = (V, E)$ and seeks a set of at most $k$ edges whose deletion from $E$ result in a bipartite chain graph (a graph that can be partitioned into two independent sets connected by a join). We obtain a kernel with $O(k^2)$ vertices for this problem. This result completes a previous result of Guo [12] who proved that the Bipartite Chain Deletion With Fixed Bipartition problem admits a kernel with $O(k^2)$ vertices.

**Outline** We begin with some definitions and notations regarding proper interval graphs. Next, we give the reduction rules the application of which leads to a kernelization algorithm for the Proper Interval Completion problem. These reduction rules allow us to obtain a kernel with at most $O(k^5)$ vertices. Finally, we prove that our techniques can be applied to Bipartite Chain Deletion to obtain a quadratic-vertex kernel, completing a previous result of Guo [12].

# 1 Preliminaries

## 1.1 Proper interval graphs

We consider simple, loopless, undirected graphs $G = (V, E)$ where $V(G)$ denotes the vertex set of $G$ and $E(G)$ its edge set[2]. Given a vertex $v \in V$, we use $N_G(v)$ to denote the *open neighborhood* of $v$ and $N_G[v] = N_G(v) \cup \{v\}$ for its *closed neighborhood*. Two vertices $u$ and $v$ are *true twins* if $N[u] = N[v]$. If $u$ and $v$ are not true twins but $uv \in E$, we say that a vertex of $N[u] \triangle N[v]$ *distinguishes* $u$ and $v$. Given a subset of vertices $S \subseteq V$, $N_S(v)$ denotes the set $N_G(v) \cap S$ and $N_G(S)$ denotes the set $\{N_G(s) \setminus S : s \in S\}$. Moreover, $G[S]$ denotes the subgraph *induced* by $S$, i.e. $G[S] = (S, E_S)$ where $E_S = \{uv \in E \ : \ u, v \in S\}$. A *join* in a graph $G = (V, E)$ is a bipartition $(X, Y)$ of $G$ and an order $x_1, \ldots, x_{|X|}$ on $X$ such that for all $i = 1, \ldots, |X| - 1$, $N_Y(x_i) \subseteq N_Y(x_{i+1})$. The edges between $X$ and $Y$ are called the *edges of the join*, and a subset $F \subseteq E$ is said to *form a join* if $F$ corresponds to the edges of a join of $G$. Finally, a graph is an *interval graph* if it admits a representation on the real line such that: $(i)$ the vertices of $G$ are in bijection with intervals of the real line and $(ii)$ $uv \in E$ if and only if $I_u \cap I_v \neq \emptyset$, where $I_u$ and $I_v$ denote the intervals associated to $u$ and $v$, respectively. Such a graph is said to admit an *interval representation*. A graph is a *proper interval graph* if it admits an interval representation such that $I_u \not\subset I_v$ for every $u, v \in V$. In other words, no interval strictly contains another interval.
We will make use of the two following characterizations of proper interval graphs to design our kernelization algorithm.

**Theorem 1.1** (Forbidden subgraphs [28])**.** *A graph is a proper interval graph if and only if it does not contain any* $\{hole, claw, net, 3\text{-}sun\}$ *as an induced subgraph (see Figure 1).*

The claw graph is the bipartite graph $K_{1,3}$. Denoting the bipartition by $(\{c\}, \{l_1, l_2, l_3\})$, we call $c$ the *center* and $\{l_1, l_2, l_3\}$ the *leaves* of the claw.

**Theorem 1.2** (Umbrella property [17])**.** *A graph is a proper interval graph if and only if its vertices admit an ordering* $\sigma$ *(called* umbrella ordering*) satisfying the following property: given* $v_iv_j \in E$ *with* $i < j$ *then* $v_iv_l, v_lv_j \in E$ *for every* $i < l < j$ *(see Figure 2).*

In the following, we associate an umbrella ordering $\sigma_G$ to any proper interval graph $G = (V, E)$. There are several things to remark. First, note that in an umbrella ordering $\sigma_G$ of a graph $G$, every

[2]In all our notations, we forget the mention to the graph $G$ whenever the context is clear.

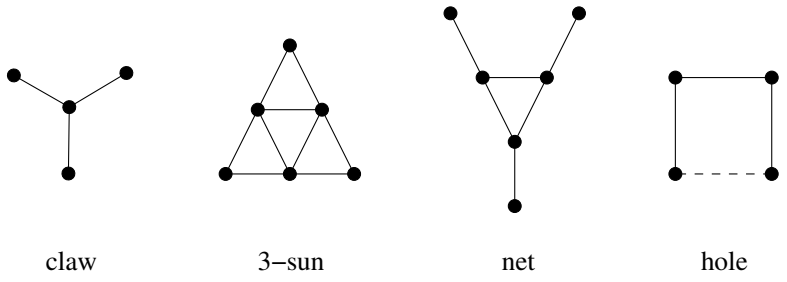


Figure 1: The forbidden induced subgraphs of proper interval graphs. A *hole* is an induced cycle of length at least 4.

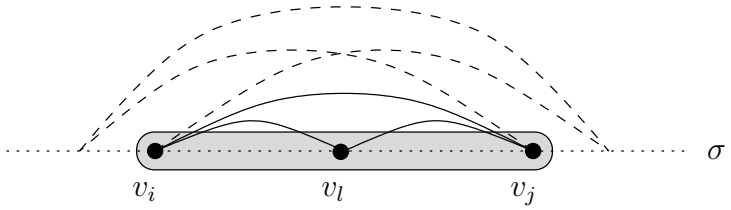


Figure 2: Illustration of the umbrella property. The edge $v_iv_j$ is extremal. [3]

maximal set of true twins of $G$ is consecutive, and that $\sigma_G$ is unique up to permutation of true twins of $G$. Remark also that for any edge $uv$ with $u <_{\sigma_G} v$, the set $\{w \in V \ : \ u \leq_{\sigma_G} w \leq_{\sigma_G} v\}$ is a clique of $G$, and for every $i$ with $1 \leq i < l$, $(\{v_1, \ldots, v_i\}, \{v_{i+1}, \ldots, v_n\})$ is a join of $G$.
According to this ordering, we say that an edge $uv$ is *extremal* if there does not exist any edge $u'v'$ different of $uv$ such that $u' \leq_{\sigma_G} u$ and $v \leq_{\sigma_G} v'$ (see Figure 2).

Let $G = (V, E)$ be an instance of Proper Interval Completion. A *completion* of $G$ is a set $F \subseteq (V \times V) \setminus E$ such that the graph $H = (V, E \cup F)$ is a proper interval graph. In a slight abuse of notation, we use $G + F$ to denote the graph $H$. A *$k$-completion* of $G$ is a completion such that $|F| \leq k$, and an *optimal completion* $F$ is such that $|F|$ is minimum. We say that $G = (V, E)$ is a *positive* instance of Proper Interval Completion whenever it admits a $k$-completion. We state a simple observation that will be very useful for our kernelization algorithm.

**Observation 1.3.** *Let $G = (V, E)$ be a graph and $F$ be an optimal completion of $G$. Given an umbrella ordering $\sigma$ of $G + F$, any extremal edge of $\sigma$ is an edge of $G$.*

*Proof.* Assume that there exists an extremal edge $e$ in $\sigma$ that belongs to $F$. By definition, $\sigma$ is still an umbrella ordering if we remove the edge $e$ from $F$, contradicting the optimality of $F$. □

## 1.2 Branches

We now give the main definitions of this Section. The branches that we will define correspond to some parts of the graph that already behave like proper interval graphs. They are the parts of the graph that we will reduce in order to obtain a kernelization algorithm.

[3] In all the figures, (non-)edges between blocks stand for all the possible (non-)edges between the vertices that lie in these blocks, and the vertices within a gray box form a clique of the graph.

**Definition 1.4** (1-branch)**.** *Let $B \subseteq V$. We say that $B$ is a 1-branch if the following properties hold (see Figure 3):*

*(i) The graph $G[B]$ is a connected proper interval graph admitting an umbrella ordering $\sigma_B = b_1, \ldots, b_{|B|}$ and,*

*(ii) The vertex set $V \setminus B$ can be partitioned into two sets $R$ and $C$ with: no edges between $B$ and $C$, every vertex in $R$ has a neighbor in $B$, no edges between $\{b_1, \ldots, b_{l-1}\}$ and $R$ where $b_l$ is the neighbor of $b_{|B|}$ with minimal index in $\sigma_B$, and for every $l \leq i < |B|$, we have $N_R(b_i) \subseteq N_R(b_{i+1})$.*

We denote by $B_1$ the set of vertices $\{v \in V : \ b_l \leq_{\sigma_B} v \leq_{\sigma_B} b_{|B|}\}$, which is a clique (because $b_l$ is a neighbor of $b_{|B|}$). We call $B_1$ the *attachment clique* of $B$, and use $B^R$ to denote $B \setminus B_1$.

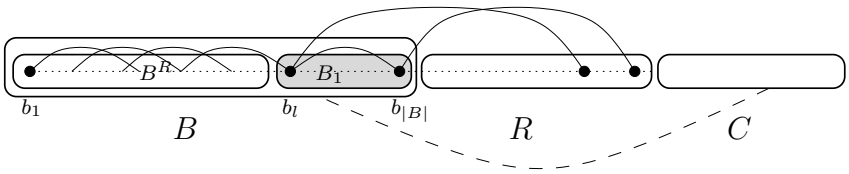


Figure 3: A 1-branch of a graph $G = (V, E)$. The vertices of $B$ are ordered according to the umbrella ordering $\sigma_B$.

**Definition 1.5** (2-branch)**.** *Let $B \subseteq V$. We say that $B$ is a 2-branch if the following properties hold (see Figure 4):*

*(i) The graph $G[B]$ is a connected proper interval graph admitting an umbrella ordering $\sigma_B = b_1, \ldots, b_{|B|}$ and,*

*(ii) The vertex set $V \setminus B$ can be partitioned into sets $L, R$ and $C$ with:*

- *no edges between $B$ and $C$,*
- *every vertex in $L$ (resp. $R$) has a neighbor in $B$,*
- *no edges between $\{b_1, \ldots, b_{l-1}\}$ and $R$ where $b_l$ is the neighbor of $b_{|B|}$ with minimal index in $\sigma_B$,*
- *no edges between $\{b_{l'+1}, \ldots, b_{|B|}\}$ and $L$ where $b_{l'}$ is the neighbor of $b_1$ with maximal index in $\sigma_B$ and,*
- *$N_R(b_i) \subseteq N_R(b_{i+1})$ for every $l \leq i < |B|$ and $N_L(b_{i+1}) \subseteq N_L(b_i)$ for every $1 \leq i < l'$.*

Again, we denote by $B_1$ (resp. $B_2$) the set of vertices $\{v \in V : \ b_1 \leq_{\sigma_B} v \leq_{\sigma_B} b_{l'}\}$ (resp. $\{v \in V : \ b_l \leq_{\sigma_B} v \leq_{\sigma_B} b_{|B|}\}$). We call $B_1$ and $B_2$ the *attachment cliques* of $B$, and use $B^R$ to denote $B \setminus (B_1 \cup B_2)$. Observe that the cases where $L = \emptyset$ or $R = \emptyset$ are possible, and correspond to the definition of a 1-branch. Finally, when $B^R = \emptyset$, it is possible that a vertex of $L$ or $R$ is adjacent to all the vertices of $B$. In this case, we will denote by $N$ the set of vertices that are adjacent to every vertex of $B$, remove them from $R$ and $L$ and abusively still denote by $L$ (resp. $R$) the set $L \setminus N$ (resp. $R \setminus N$). We will precise when we need to use the set $N$.

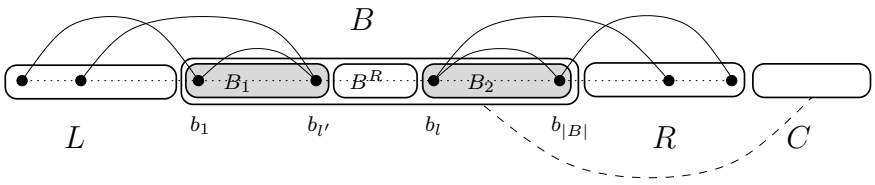


Figure 4: A 2-branch of a graph $G = (V, E)$. The vertices of $B$ are ordered according to the umbrella ordering $\sigma_B$.

In both cases, in a 1- or 2-branch, whenever the proper interval graph $G[B]$ is a *clique*, we say that $B$ is a *$K$-join*. Observe that, in a 1- or 2-branch $B$, for any extremal edge $uv$ in $\sigma_B$, the set of vertices $\{w \in V : u \leq_{\sigma_B} w \leq_{\sigma_B} v\}$ defines a $K$-join. In particular, this means that a branch can be decomposed into a sequence of $K$-joins. Observe however that the decomposition is not unique: for instance, the $K$-joins corresponding to all the extremal edges of $\sigma_B$ are not disjoint. We will precise in Section 2.1.5, when we will reduce the size of 2-branches, how to fix a decomposition. Finally, we say that a $K$-join is *clean* whenever its vertices are not contained in any claw or 4-cycle. Remark that a subset of a $K$-join (resp. clean $K$-join) is also a $K$-join (resp. clean $K$-join).

# 2 Kernel for Proper Interval Completion

The basic idea of our kernelization algorithm is to detect the large enough branches and then to reduce them. This section details the rules we use for that.

## 2.1 Reduction rules

### 2.1.1 Basic rules

We say that a rule is *safe* if when it is applied to an instance $(G, k)$ of the problem, $(G, k)$ admits a $k$-completion iff the instance $(G', k')$ reduced by the rule admits a $k'$-completion.

The first reduction rule gets rid of connected components that are already proper interval graphs. This rule is trivially safe and can be applied in $O(n + m)$ time using any recognition algorithm for proper interval graphs [6].

**Rule 2.1** (Connected components)**.** *Remove any connected component of $G$ that is a proper interval graph.*

The following reduction rule can be applied since proper interval graphs are closed under true twin addition and induced subgraphs. For a class of graphs satisfying these two properties, we know that this rule is safe [1] (roughly speaking, we edit all the large set of true twins in the same way).

**Rule 2.2** (True twins [1])**.** *Let $T$ be a set of true twins in $G$ such that $|T| > k$. Remove $|T| - (k+1)$ arbitrary vertices from $T$.*

We also use the classical *sunflower* rule, allowing to identify a set of edges that must be added in any optimal completion.

**Rule 2.3** (Sunflower)**.** *Let $\mathcal{S} = \{C_1, \ldots, C_m\}$, $m > k$ be a set of claws having two leaves $u, v$ in common but distinct third leaves. Add $uv$ to $F$ and decrease $k$ by* 1*.*
*Let $\mathcal{S} = \{C_1, \ldots, C_m\}$, $m > k$ be a set of distinct* 4*-cycles having a non-edge $uv$ in common. Add $uv$ to $F$ and decrease $k$ by* 1*.*

**Lemma 2.1.** *Rule 2.3 is safe and can be carried out in polynomial time.*

*Proof.* We only prove the first rule. The second rule can be proved similarly. Let $F$ be a $k$-completion of $G$ and assume that $F$ does not contain $(u, v)$. Since any two claws in $\mathcal{S}$ only share $(u, v)$ as a common non-edge, $F$ must contain one edge for every $C_i$, $1 \leq i \leq m$. Since $m > k$, we have $|F| > k$, which cannot be. Observe that a sunflower can be found in polynomial time once we have enumerated all the claws and 4-cycles of a graph, which can clearly be done in $O(n^4)$. □

### 2.1.2 Extracting a clean $K$-join from a $K$-join

Now, we want to reduce the size of the 'simplest' branches, namely the $K$-joins. More precisely, in the next subsection we will bound the number of vertices in a clean $K$-join (whose vertices are not contain in any claw or 4-cycle), and so, we first indicate how to extract a clean $K$-join from a $K$-join.

**Lemma 2.2.** *Let $G = (V, E)$ be a positive instance of* Proper Interval Completion *on which Rule 2.3 has been applied. There are at most $k^2$ claws with distinct sets of leaves, and at most $k^2 + 2k$ vertices of $G$ are leaves of claw. Furthermore, there are at most $2k^2 + 2k$ vertices of $G$ that are vertices of a* 4*-cycle.*

*Proof.* As $G$ is a positive instance of Proper Interval Completion, every claw or 4-cycle of $G$ has a non-edge that will be completed and then is an edge of $F$. Let $xy$ be an edge of $F$. As we have applied Rule 2.3 on $G$, there are at most $k$ vertices in $G$ that form the three leaves of a claw with $x$ and $y$. So, at most $(k + 2)k$ vertices of $G$ are leaves of claws. Similarly, there are at most $k$ non-edges of $G$, implying at most $2k$ vertices, that form a 4-cycle with $x$ and $y$. So, at most $(2k + 2)k$ vertices of $G$ are in a 4-cycle. □

**Lemma 2.3.** *Let $G = (V, E)$ be a positive instance of* Proper Interval Completion *on which Rule 2.2 and Rule 2.3 have been applied and $B$ be a $K$-join of $G$. There are at most $k^3+4k^2+5k+1$ vertices of $B$ that belong to a claw or a* 4*-cycle.*

*Proof.* By Lemma 2.2, there are at most $3k^2 + 4k$ vertices of $B$ that are leaves of a claw or in 4-cycles. We remove these vertices from $B$ and denote $B'$ the set of remaining vertices, which forms a $K$-join. Now, we remove from $B'$ all the vertices that do not belong to any claw and contract all the true twins in the remaining vertices. As Rule 2.2 has been applied on $B$, every contracted set has size at most $k + 1$. We denote by $B''$ the obtained set which can be seen as a subset of $B$ and then, $B''$ is also a $K$-join of $G$. Remark that every vertex of $B''$ is the center of a claw. We consider an umbrella ordering $b_1, \ldots, b_l$ of $B''$. We will find a set of $l - 1$ claws with distinct sets of leaves, which will bound $l$ by $k^2 + 1$, by Lemma 2.2. As, for all $i = 1, \ldots, l - 1$, $b_i$ and $b_{i+1}$ are not true twins, there exists $c_i$ such that $b_i c_i \in E$ and $b_{i+1} c_i \notin E$ or $b_i c_i \notin E$ and $b_{i+1} c_i \in E$. As $B''$ is $K$-join, by definition, all the $c_i$'s are distinct. Now, for every $i = 1, \ldots, l - 1$, we will find a claw containing $c_i$ as leaf. Assume that $b_i c_i \notin E$ and $b_{i+1} c_i \in E$. As $b_{i+1}$ is the center of a claw, there exists a set $\{x, y, z\}$ which is an independent set and is fully adjacent to $b_{i+1}$. If $c_i \in \{x, y, z\}$, we

are done. Assume this is not the case. This means that $b_i$ is adjacent to any vertex of $\{x, y, z\}$ (otherwise one of this vertex would be adjacent to $b_{i+1}$ and not to $b_i$, and we choose it to be $c_i$). Now, if two elements of this set, say $x$ and $y$, are adjacent to $c_i$, then $\{x, c_i, y, b_i\}$ forms a 4-cycle that contains $b_i$, which is not possible. So, at least two elements among $\{x, y, z\}$, say $x$ and $y$, are not adjacent to $c_i$ and then, we find the claw $\{b_{i+1}, x, y, c_i\}$ of center $b_{i+1}$ that contains $c_i$. In the case where $b_ic_i \in E$ and $b_{i+1}c_i \notin E$, we proceed similarly by exchanging the role of $b_i$ and $b_{i+1}$ and find also a claw containing $c_i$. Finally, all the considered claws have distinct sets of leaves and there are at most $k^2$ such claws by Lemma 2.2. What means that $B''$ has size at most $k^2 + 1$ and $B'$ at most $(k + 1)(k^2 + 1)$. As we removed at most $3k^2 + 4k$ vertices of $B$ that could be leaves of claws or contain in 4-cycles, we obtain $k^3 + 4k^2 + 5k + 1$ vertices of $B$ that are possibly in claws or 4-cycles. □

Since any subset of a $K$-join forms a $K$-join, Lemma 2.3 implies that it is possible to remove a set of at most $k^3 + 4k^2 + 5k + 1$ vertices from any $K$-join to obtain a clean $K$-join.

### 2.1.3 Bounding the size of the $K$-joins

Now, we set a rule that will bound the number of vertices in a clean $K$-join, once applied. Although quite technical to prove, this rule is the core tool of our process of kernelization.

**Rule 2.4** ($K$-join)**.** *Let $B$ be a clean $K$-join of size at least $2k+2$. Let $B_L$ be the $k+1$ first vertices of $B$, $B_R$ be its $k+1$ last vertices and $M = B \setminus (B_R \cup B_L)$. Remove the set of vertices $M$ from $G$.*

**Lemma 2.4.** *Rule 2.4 is safe.*

*Proof.* Let $G' = G \setminus M$. Observe that the restriction to $G'$ of any $k$-completion of $G$ is a $k$-completion of $G'$, since proper interval graphs are closed under induced subgraphs. So, let $F$ be a $k$-completion for $G'$. We denote by $H = G' + F$ the resulting proper interval graph and by $\sigma_H = b_1, \ldots, b_{|H|}$ an umbrella ordering of $H$. We prove that we can insert the vertices of $M$ into $\sigma_H$ and modify it if necessary, to obtain an umbrella ordering for $G$ without adding any edge (in fact, some edges of $F$ might even be deleted during the process). This will imply that $G$ admits a $k$-completion as well. To see this, we need the following structural description of $G$. As explained before, we denote by $N$ the set $\cap_{b \in B} N_G(b) \setminus B$, and abusively still denote by $L$ (resp. $R$) the set $L \setminus N$ (resp. $R \setminus N$) (see Figure 5).

**Claim 2.5.** *The sets $L$ and $R$ are cliques of $G$.*

*Proof.* We prove that $R$ is a clique in $G$. The proof for $L$ uses similar arguments. No vertex of $R$ is a neighbor of $b_1$, otherwise such a vertex must be adjacent to every vertex of $B$ and then stand in $N$. So, if $R$ contains two vertices $u, v$ such that $uv \notin E$, we form the claw $\{b_{|B|}, b_1, u, v\}$ of center $b_{|B|}$, contradicting the fact that $B$ is clean. ◇

The following observation comes from the definition of a $K$-join.

**Observation 2.6.** *Given any vertex $r \in R$, if $N_B(r) \cap B_L \neq \emptyset$ holds then $M \subseteq N_B(r)$. Similarly, given any vertex $l \in L$, if $N_B(l) \cap B_R \neq \emptyset$ holds then $M \subseteq N_B(l)$.*

We use these facts to prove that an umbrella ordering can be obtained for $G$ by inserting the vertices of $M$ into $\sigma_H$. Let $b_f$ and $b_l$ be respectively the first and last vertex of $B \setminus M$ appearing in $\sigma_H$. We let $B_H$ denote the set $\{u \in V(H) \ : \ b_f \leq_{\sigma_H} u \leq_{\sigma_H} b_l\}$. Observe that $B_H$ is a clique in

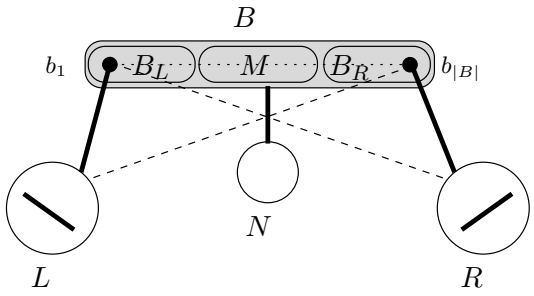


Figure 5: The structure of the $K$-join $B$.

$H$ since $b_f b_l \in E(G)$ and that $B \setminus M \subseteq B_H$. Now, we modify $\sigma_H$ by ordering the true twins in $H$ according to their neighborhood in $M$: if $x$ and $y$ are true twins in $H$, are consecutive in $\sigma_H$, verify $x <_{\sigma_H} y <_{\sigma_H} b_f$ and $N_M(y) \subset N_M(x)$, then we exchange $x$ and $y$ in $\sigma_H$. This process stops when the considered true twins are ordered following the join between $\{u \in V(H) \ : \ u <_{\sigma_H} b_f\}$ and $M$. We proceed similarly on the right of $B_H$, i.e. for $x$ and $y$ consecutive twins with $b_l <_{\sigma_H} x <_{\sigma_H} y$ and $N_M(x) \subset N_M(y)$. The obtained order is clearly an umbrella ordering too (in fact, we just re-labeled some vertices in $\sigma_H$), and we abusively still denote it by $\sigma_H$.

**Claim 2.7.** *The set $B_H \cup \{m\}$ is a clique of $G$ for any $m \in M$, and consequently $B_H \cup M$ is a clique of $G$.*

*Proof.* Let $u$ be any vertex of $B_H$. We claim that $um \in E(G)$. Observe that if $u \in B$ then the claim trivially holds. So assume $u \notin B$. Recall that $B_H$ is a clique in $H$. It follows that $u$ is adjacent to every vertex of $B \setminus M$ in $H$. Since $B_L$ and $B_R$ both contain $k+1$ vertices, we have $N_G(u) \cap B_L \neq \emptyset$ and $N_G(u) \cap B_R \neq \emptyset$. Hence, $u$ belongs to $L \cup N \cup R$ and $um \in E(G)$ by Observation 2.6. ◇

**Claim 2.8.** *Let $m$ be any vertex of $M$ and $\sigma'_H$ be the ordering obtained from $\sigma_H$ by removing $B_H$ and inserting $m$ to the position of $B_H$. The ordering $\sigma'_H$ respects the umbrella property.*

*Proof.* Assume that $\sigma'_H$ does not respect the umbrella property, i.e. that there exist (w.l.o.g.) two vertices $u$ and $v$ of $H \setminus B_H$ such that either (1) $u <_{\sigma'_H} v <_{\sigma'_H} m$, $um \in E(H)$ and $uv \notin E(H)$ or (2) $u <_{\sigma'_H} m <_{\sigma'_H} v$, $um \notin E(H)$ and $uv \in E(H)$ or (3) $u <_{\sigma'_H} v <_{\sigma'_H} m$, $um \in E(H)$ and $vm \notin E(H)$. First, assume that (1) holds. Since $uv \notin E(H)$ and $\sigma_H$ is an umbrella ordering, $uw \notin E(H)$ for any $w \in B_H$, and hence $uw \notin E(G)$. This means that $B_L \cap N_G(u) = \emptyset$ and $B_R \cap N_G(u) = \emptyset$, which is impossible since $um \in E(G)$. Then, assume that (2) holds. Since $uv \in E(H)$ and $\sigma_H$ is an umbrella ordering, $B_H \subseteq N_H(u)$, and in particular $B_L$ and $B_R$ are included in $N_H(u)$. As $|B_L| = |B_R| = k+1$, we know that $N_G(u) \cap B_L \neq \emptyset$ and $N_G(u) \cap B_R \neq \emptyset$, but then, Observation 2.6 implies that $um \in E(G)$. So, (3) holds, and we choose the first $u$ satisfying this property according to the order given by $\sigma'_H$. So we have $wm \notin E(G)$ for any $w <_{\sigma'_H} u$. Similarly, we choose $v$ to be the first vertex after $u$ satisfying $vm \notin E(G)$. Since $um \in E(G)$, we know that $u$ belongs to $L \cup N \cup R$. Moreover, since $vm \notin E(G)$, $v \in C \cup L \cup R$. There are several cases to consider:

(i) $u \in N$: in this case we know that $B \subseteq N_G(u)$, and in particular that $ub_l \in E(G)$. Since $\sigma_H$ is an umbrella ordering for $H$, it follows that $vb_l \in E(H)$ and $B_H \subseteq N_H(v)$. Since

$|B_L| = |B_R| = k+1$, we know that $N_G(v) \cap B_L \neq \emptyset$ and $N_G(v) \cap B_R \neq \emptyset$. But, then Observation 2.6 implies that $vm \in E(G)$.

(ii) $u \in R$, $v \notin R$: since $um \in E(G)$, $B_R \subseteq N_G(u)$. Let $b \in B_R$ be the vertex such that $B_R \subseteq \{w \in V \ : \ u <_{\sigma_H} w \leq_{\sigma_H} b\}$. Since $ub \in E(G)$, this means that $B_R \subseteq N_H(v)$. Now, since $|B_R| = k+1$, it follows that $N_G(v) \cap B_R \neq \emptyset$. Observation 2.6 allows us to conclude that $vm \in E(G)$.

(iii) $u, v \in R$: in this case, $uv \in E(G)$ by Claim 2.7 but $u$ and $v$ are not true twins in $H$ (otherwise $v$ would be placed before $u$ in $\sigma_H$ due to the modification we have applied to $\sigma_H$). This means that there exists a vertex $w \in V(H)$ that distinguishes $u$ from $v$ in $H$.
Assume first that $w <_{\sigma_H} u$ and $uw \in E(H), vw \notin E(H)$. We choose the first $w$ satisfying this according to the order given by $\sigma_H$. There are two cases to consider. First, if $uw \in E(G)$, then since $wm \notin E(G)$ for any $w <_{\sigma_H} u$ by the choice of $u$, $\{u, v, w, m\}$ is a claw in $G$ containing a vertex of $B$ (see Figure 6 $(a)$ ignoring the vertex $u'$), which cannot be. So assume $uw \in F$. By Observation 1.3, $uw$ is not an extremal edge of $\sigma_H$. By the choice of $w$ and since $vw \notin E(H)$, there exists $u'$ with $u <_{\sigma_H} u' <_{\sigma_H} v$ such that $u'w$ is an extremal edge of $\sigma_H$ (and hence belongs to $E(G)$, see Figure 6 $(a)$). Now, by the choice of $v$ we have $u'm \in E(G)$ and hence $u' \in N \cup R \cup L$. Observe that $u'v \notin E(G)$: otherwise $\{u', v, w, m\}$ would form a claw in $G$. Since $R$ is a clique of $G$, it follows that $u' \in L \cup N$. Moreover, since $u'm \in E(G)$, $B_L \subseteq N_G(u')$. We conclude like in configuration $(ii)$ that $v$ should be adjacent to a vertex of $B_L$ and hence to $m$.
Hence we can assume that all the vertices that distinguish $u$ and $v$ are after $u$ in $\sigma_H$ and that $uw'' \in E(H)$ implies $vw'' \in E(H)$ for any $w'' <_{\sigma_H} u$. Now, suppose that there exists $w \in H$ such that $b_l <_{\sigma_H} w$ and $uw \notin E(H)$, $vw \in E(H)$. In particular, this means that $B_L \subseteq N_H(v)$. Since $|B_L| = k+1$ we have $N_G(v) \cap B_L \neq \emptyset$, implying $vm \in E(G)$ by Observation 2.6. Assume now that there exists a vertex $w$ which distinguishes $u$ and $v$ with $v <_{\sigma_H} w <_{\sigma_H} b_f$. In this case, since $uw \notin E(H)$, $B \cap N_H(u) = \emptyset$ holds and hence $B \cap N_G(u) = \emptyset$, which cannot be since $u \in R$. Finally, assume that there is $w \in B_H$ with $wu \notin E(H)$ and $wv \in E(H)$. Recall that $wm \in E(G)$ as $B_H \cup \{m\}$ is a clique by Claim 2.7. We choose $w$ in $B_H$ distinguishing $u$ and $v$ to be the last according to the order given by $\sigma_H$ (i.e. $vw' \notin E(H)$ for any $w <_{\sigma_H} w'$, see Figure 6 $(b)$, ignoring the vertex $u'$).

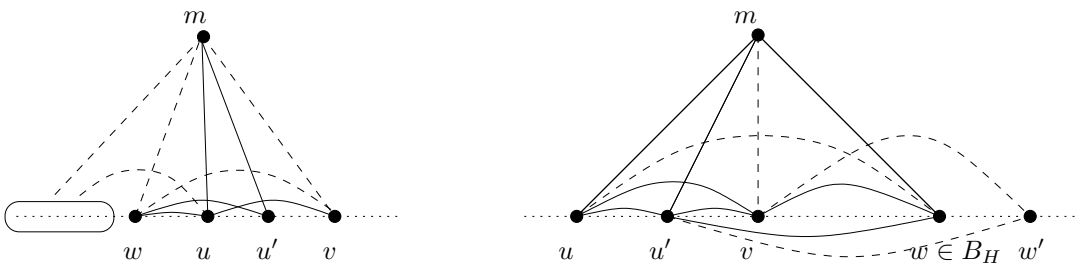


Figure 6: $(a)$ $u$ and $v$ are distinguished by some vertex $w <_{\sigma_H} u$; $(b)$ $u$ and $v$ are distinguished by a vertex $w \in B_H$.

If $vw \in E(G)$ then $\{u, m, w, v\}$ is a 4-cycle in $G$ containing a vertex of $B$, which cannot be.

Hence $vw \in F$ and by the choice of $w$, there exists $u' \in V(H)$ such that $u <_{\sigma_H} u' <_{\sigma_H} v$ and $u'w$ is an extremal edge (and then belongs to $E(G)$). By the choice of $v$ we know that $u'm \in E(G)$. Moreover, by the choice of $w$, observe that $u'$ and $v$ are true twins in $H$ (if a vertex $s$ distinguishes $u'$ and $v$ in $H$, $s$ cannot be before $u$, since otherwise $s$ would distinguishes $u$ and $v$, not between $u$ and $w$ because it would be adjacent to $u'$ and $v$, and not after $w$, by choice of $w$). This leads to a contradiction since we assumed that $N_M(x) \subseteq N_M(y)$ for any true twins $x$ and $y$ with $x <_{\sigma_H} y <_{\sigma_H} b_f$.

The cases where $u \in L$ are similar, what concludes the proof of Claim 2.8 $\diamond$

**Claim 2.9.** *Let $m \in M$. Then $m$ can be added to the graph $H$ while preserving an umbrella ordering.*

*Proof.* Let $m \in M$ and $v_i$ (resp. $v_j$) be the vertex with minimal (resp. maximal) index in $\sigma_H$ such that $v_i m \in E(G)$ (resp. $v_j m \in E(G)$). By definition, we have $v_{i-1}m, v_{j+1}m \notin E(G)$ and through Claim 2.8, we know that $N_H(m) = \{w \in V \ : \ v_i \leq_{\sigma_H} w \leq_{\sigma_H} v_j\}$. Moreover, since $B_H \cup M$ is a clique by Claim 2.7, it follows that $v_{i-1} <_{\sigma_H} b_f$ and $b_l <_{\sigma_H} v_{j+1}$. Hence, by Claim 2.8, we know that $v_{i-1}v_{j+1} \notin E(G)$, otherwise the ordering $\sigma'_H$ defined in Claim 2.8 would not be an umbrella ordering. The situation is depicted in Figure 7 $(a)$. For any vertex $v \in N_H(m)$, let $N^-(v)$ (resp. $N^+(v)$) denote the set of vertices $\{w \leq_{\sigma_H} v_{i-1} \ : \ wv \in E(H)\}$ (resp. $\{w \geq_{\sigma_H} v_{j+1} \ : \ wv \in E(H)\}$). Observe that for any vertex $v \in N_H(m)$, if there exist two vertices $x \in N^-(v)$ and $y \in N^+(v)$ such that $xv, yv \in E(G)$, then the set $\{v, x, y, m\}$ defines a claw containing $m$ in $G$, which cannot be. We now consider $b_{v_{i-1}}$ the neighbor of $v_{i-1}$ with maximal index in $\sigma_H$. Similarly we let $b_{v_{j+1}}$ be the neighbor of $v_{j+1}$ with minimal index in $\sigma_H$. Since $v_{i-1}v_{j+1} \notin E(G)$, we have $b_{v_{i-1}}, b_{v_{j+1}} \in N_H(m)$. We study the behavior of $b_{v_{i-1}}$ and $b_{v_{j+1}}$ in order to conclude.

Assume first that $b_{v_{j+1}} <_{\sigma_H} b_{v_{i-1}}$. Let $X$ be the set of vertices $\{w \in V \ : \ b_{v_{j+1}} \leq_{\sigma_H} w \leq_{\sigma_H} b_{v_{i-1}}\}$. Remark that we have $b_{v_{i-1}} \leq_{\sigma_H} b_l$ and $b_f \leq_{\sigma_H} b_{v_{j+1}}$, otherwise for instance, if we have $b_{v_{i-1}} >_{\sigma_H} b_l$, then $B_H \subseteq N_H(v_{i-1})$ implying, as usual, that $v_{i-1}m \in E(G)$ which is not. So, we know that $X \subseteq B_H$. Then, let $X_1 \subseteq X$ be the set of vertices $x \in X$ such that there exists $w \in N^+(x)$ with $xw \in E(G)$ and $X_2 = X \setminus X_1$. Let $x \in X_1$: observe that by construction $xw' \in F$ for any $w' \in N^-(x)$. Similarly, given $x \in X_2$, $xw'' \in F$ for any $w'' \in N^+(x)$. We now reorder the vertices of $X$ as follows: we first put the vertices from $X_2$ and then the vertices from $X_1$, preserving the order induced by $\sigma_H$ for both sets. Moreover, we remove from $E(H)$ all edges between $X_1$ and $N^-(X_1)$ and between $X_2$ and $N^+(X_2)$. Recall that such edges have to belong to $F$. We claim that inserting $m$ *between* $X_2$ and $X_1$ yields an umbrella ordering (see Figure 7 $b$). Indeed, by Claim 2.8, we know that the umbrella ordering is preserved between $m$ and the vertices of $H \setminus B_H$.

Now, remark that there is no edge between $X_1$ and $\{w \in V \ : \ w \leq_{\sigma_H} v_{i-1}\}$, that there is no edge between $X_2$ and $\{w \in V \ : \ w \geq_{\sigma_H} v_{j+1}\}$), that there are still all the edges between $N_H(m)$ and $X_1 \cup X_2$ and that the edges between $X_1$ and $\{w \in V \ : \ w \geq_{\sigma_H} v_{j+1}\}$ and the edges between $X_2$ and $\{w \in V \ : \ w \leq_{\sigma_H} v_{i-1}\}$ are unchanged. So, it follows that the new ordering respects the umbrella property, and we are done.

Next, assume that $b_{v_{i-1}} <_{\sigma_H} b_{v_{j+1}}$. We let $b_{v_i}$ (resp. $b_{v_j}$) be the neighbor of $v_i$ (resp. $v_j$) with maximal (resp. minimal) index in $N_H(m)$. Notice that $b_{v_{i-1}} \leq_{\sigma_H} b_{v_i}$ and $b_{v_j} \leq_{\sigma_H} b_{v_{j+1}}$ (see Figure 8). Two cases may occur:

(i) First, assume that $b_{v_i} <_{\sigma_H} b_{v_j}$, case depicted in Figure 8 $(a)$. In particular, this means that $v_iv_j \notin E(G)$. If $b_{v_i}$ and $b_{v_j}$ are consecutive in $\sigma_H$, then inserting $m$ between $b_{v_i}$ and $b_{v_j}$ yields

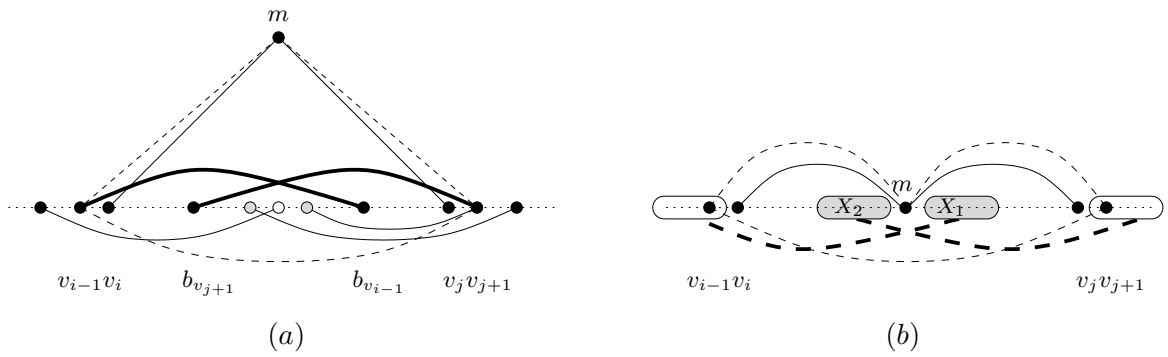


Figure 7: Illustration of the reordering applied to $\sigma_H$. The thin edges stand for edges of $G$. On the left, the gray vertices represent vertices of $X_1$ while the white vertex is a vertex of $X_2$.

an umbrella ordering (since $b_{v_j}$ (resp. $b_{v_i}$) does not have any neighbor before (resp. after) $v_i$ (resp. $v_j$) in $\sigma_H$). Now, if there exists $w \in V$ such that $b_{v_i} <_{\sigma_H} w <_{\sigma_H} b_{v_j}$, then one can see that the set $\{m, v_i, w, v_j\}$ forms a claw containing $m$ in $G$, which is impossible.

(ii) The second case to consider is when $b_{v_j} \leq_{\sigma_H} b_{v_i}$. In such a case, one can see that $m$ and the vertices of $\{w \in V \ : \ b_{v_j} \leq_{\sigma_H} w \leq_{\sigma_H} b_{v_i}\}$ are true twins in $H \cup \{m\}$, because their common neighborhood is exactly $\{w \in V \ : \ v_i \leq_{\sigma_H} w \leq_{\sigma_H} v_j\}$. Hence, inserting $m$ just before $b_{v_i}$ (or anywhere between $b_{v_i}$ and $b_{v_j}$ or just after $b_{v_j}$) yields an umbrella ordering.

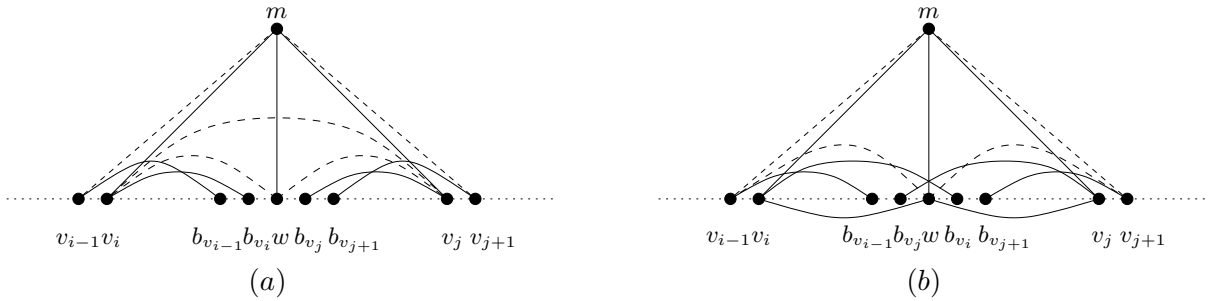


Figure 8: The possible cases for $b_{v_{i-1}} <_{\sigma_H} b_{v_{j+1}}$.

◇

Since the proof of Claim 2.9 does not use the fact that the vertices of $H$ do not belong to $M$, it follows that we can iteratively insert the vertices of $M$ into $\sigma_H$, preserving an umbrella ordering at each step. This concludes the proof of Lemma 2.4. □

The complexity needed to compute Rule 2.4 will be discussed in the next section. The following observation results from the application of Rule 2.4 and from Section 2.1.2.

**Observation 2.10.** *Let $G = (V, E)$ be a positive instance of* Proper Interval Completion *reduced under Rules 2.2 to 2.4. Any $K$-join of $G$ has size at most $k^3 + 4k^2 + 7k + 3$.*

*Proof.* Let $B$ be any $K$-join of $G$, and assume $|B| > k^3+4k^2+7k+3$. By Lemma 2.2 we know that it is possible to extract a clean $K$-join from $B$ of size at least $|B| - (k^3 + 4k^2 + 5k + 1) > 2(k + 1)$ what is impossible after having applied Rule 2.4. □

### 2.1.4 Cutting the 1-branches

We now turn our attention to branches of a graph $G = (V, E)$, proving how they can be reduced.

**Lemma 2.11.** *Let $G = (V, E)$ be a connected graph and $B$ be a 1-branch of $G$ associated with the umbrella ordering $\sigma_B$. Assume that $|B^R| \geq 2k + 1$ and let $B_f$ be the $2k + 1$ last vertices of $B^R$ according to $\sigma_B$. For any $k$-completion $F$ of $G$ into a proper interval graph, there exists a $k$-completion $F'$ of $G$ with $F' \subseteq F$ and a vertex $b \in B_f$ such that the umbrella ordering of $G + F'$ preserves the order of the set $B_b = \{v \in V \ : \ b_1 \leq_{\sigma_B} v \leq_{\sigma_B} b_{l'}\}$, where $l'$ is the maximal index such that $bb_{l'} \in E(G)$. Moreover, the vertices of $B_b$ are the first in an umbrella ordering of $G + F'$.*

*Proof.* Let $F$ be any $k$-completion of $G$, $H = G + F$ and $\sigma_H$ be the umbrella ordering of $H$. Since $|B_f| = 2k + 1$ and $|F| \leq k$, there exists a vertex $b \in B_f$ not incident to any added edge of $F$. We let $N_D$ be the set of neighbors of $b$ that are *after* $b$ in $\sigma_B$, $B'$ the set of vertices that are *before* $N_G[b]$ in $\sigma_B$, $B_b = B' \cup N_G[b]$ and $C = V \setminus B_b$ (see Figure 9).

**Claim 2.12.** *(i) $G[C]$ is a connected graph and*
*(ii) Either $\forall u \in C$ $b <_{\sigma_H} u$ holds or $\forall u \in C$ $u <_{\sigma_H} b$ holds.*

*Proof.* The first point follows from the fact that $G$ is connected and that, by construction, $B_1 \subseteq C$ and $B_1$ is connected. To see the second point, assume that there exist $u, v \in C$ such that w.l.o.g. $u <_{\sigma_H} b <_{\sigma_H} v$. Since $G[C]$ is a connected graph, there exists a path between $u$ and $v$ in $G$ that avoids $N_G[b]$, which is equal to $N_H[b]$ since $b$ is not incident to any edge of $F$. Hence there exist $u', v' \in C$ such that $u' <_{\sigma_H} b <_{\sigma_H} v'$ and $u'v' \in E(G)$. Then, we have $u'b, v'b \notin E(H)$, contradicting the fact that $\sigma_H$ is an umbrella ordering for $H$. ◇

In the following, we assume w.l.o.g. that $b <_{\sigma_H} u$ holds for any $u \in C$. We now consider the following ordering $\sigma$ of $H$: we first put the set $B_b$ according to the order of $B$ and then put the remaining vertices $C$ according to $\sigma_H$ (see Figure 9). We construct a completion $F'$ from $F$ as follows: we remove from $F$ the edges with both extremities in $B_b$, and remove all edges between $B_b \setminus N_D$ and $C$. In other words, we set:

$$F' = F \setminus (F[B] \cup F[(B_b \setminus N_D) \times C])$$

Finally, we inductively remove from $F'$ any extremal edge of $\sigma$ that belongs to $F$, and abusively still call $F'$ the obtained edge set.

**Claim 2.13.** *The set $F'$ is a $k$-completion of $G$.*

*Proof.* We prove that $\sigma$ is an umbrella ordering of $H' = G+F'$. Since $|F'| \leq |F|$ by construction, the result will follow. Assume this is not the case. By definition of $F'$, $H'[B_b]$ and $H'[C]$ induce proper interval graphs. This means that there exists a set of vertices $S = \{u, v, w\}$, $u <_\sigma v <_\sigma w$, intersecting both $B_b$ and $C$ and violating the umbrella property. We either have (1) $uw \in E, uv \notin E$ or (2) $uw \in E, vw \notin E$. Since neither $F'$ nor $G$ contain an edge between $B_b \setminus N_D$ and $C$, it follows that $S$ intersects $N_D$ and $C$. We study the different cases:

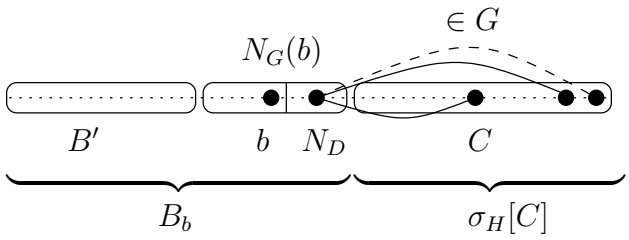


Figure 9: The construction of the ordering $\sigma$ according to $\sigma_H$.

(i) (1) holds and $u \in N_D,\ v, w \in C$: since the edge set between $N_D$ and $C$ is the same in $H$ and $H'$, it follows that $uv \notin E(H)$. Since $\sigma_H$ is an umbrella ordering of $H$, we either have $v <_{\sigma_H} u <_{\sigma_H} w$ or $v <_{\sigma_H} w <_{\sigma_H} u$ (recall that $C$ is in the same order in both $\sigma$ and $\sigma_H$). Now, recall that $b <_{\sigma_H} \{v, w\}$ by assumption. In particular, since $bu \in E(G)$, this implies in both cases that $\sigma_H$ is not an umbrella ordering, what leads to a contradiction.

(ii) (1) holds and $u, v \in N_D,\ w \in C$: this case cannot happen since $N_D$ is a clique of $H'$.

(iii) (2) holds and $u \in N_D,\ v, w \in C$: this case is similar to (i). Observe that we may assume $uv \in E(H)$ (otherwise (i) holds). By construction $vw \notin E(H)$ and hence $v <_{\sigma_H} w <_{\sigma_H} u$ or $v <_{\sigma_H} u <_{\sigma_H} w$. The former case contradicts the fact that $\sigma_H$ is an umbrella ordering since $bu \in E(H)$. In the latter case, since $\sigma_H$ is an umbrella ordering this means that $bv \in E(H)$. Since $b$ is non affected vertex and $bv \notin E(G)$, this leads to a contradiction.

(iv) (2) holds and $u, v \in N_D,\ w \in C$: first, if $uw \in E(G)$, then we have a contradiction since $N_C(u) \subseteq N_C(v)$. So, we have $uw \in F'$. By construction of $F'$, we know that $uw$ is not an extremal edge. Hence there exists an extremal edge (of $G$) containing $uw$, which is either $uw'$ with $w <_\sigma w'$ , $u'w$ with $u' <_\sigma u$ or $u'w'$ with $u' <_\sigma u <_\sigma w <_\sigma w'$. The three situation are depicted in Figure 10. In the first case, $vw' \in E(G)$ (since $N_C(u) \subseteq N_C(v)$ in $G$) and hence we are in configuration (i) with vertex set $\{v, w, w'\}$. In the second case, since $u'w \in E(G)$, we have a contradiction since $N_C(u') \subseteq N_C(v)$ in $G$ (observe that $u' \in B$ by construction). Finally, in the third case, $uw', vw' \in E(G)$ since $N_C(u') \subseteq N_C(u) \subseteq N_C(v)$ in $G$, and we are in configuration (i) with vertex set $\{v, w, w'\}$.

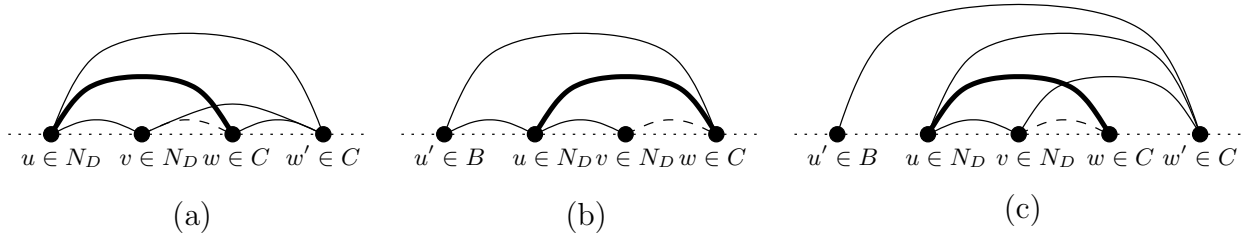


Figure 10: Illustration of the different cases of configuration (iv) (the bold edges belong to $F'$).

◇

Altogether, we proved that for any $k$-completion $F$, there exists an umbrella ordering where the vertices of $B_b$ are ordered in the same way than in the ordering of $B$ and stand at the beginning of this ordering, what concludes the proof. □

**Rule 2.5** (1-branches)**.** *Let $B$ be a 1-branch such that $|B^R| \geq 2k+1$. Remove $B^R \setminus B_f$ from $G$, where $B_f$ denotes the $2k+1$ last vertices of $B^R$.*

**Lemma 2.14.** *Rule 2.5 is safe.*

*Proof.* Let $G' = G \setminus (B^R \setminus B_f)$ denote the reduced graph. Observe that any $k$-completion of $G$ is a $k$-completion of $G'$ since proper interval graphs are closed under induced subgraphs. So let $F$ be a $k$-completion of $G'$. We denote by $H = G' + F$ the resulting proper interval graph and let $\sigma_H$ be the corresponding umbrella ordering. By Lemma 2.11 we know that there exists a vertex $b \in B_f$ such that the order of $B_b = N_G[b] \cup \{v \in B_f \ : \ v <_{\sigma_B} N_G[b]\}$ is the same than in $B$ and the vertices of $B_b$ are the first of $\sigma_H$. Since $N_G(B^R \setminus B_f) \subseteq N_G[b]$, it follows that the vertices of $B^R \setminus B_f$ can be inserted into $\sigma_H$ while respecting the umbrella property. Hence $F$ is a $k$-completion for $G$, implying the result. □

Here again, the time complexity needed to compute Rule 2.5 will be discussed in the next section. The following property of a reduced graph will be used to bound the size of our kernel.

**Observation 2.15.** *Let $G = (V, E)$ be a positive instance of* Proper Interval Completion *reduced under Rules 2.2 to 2.5. The 1-branches of $G$ contain at most $k^3 + 4k^2 + 9k + 4$ vertices.*

*Proof.* Let $B$ be a 1-branch of a graph $G = (V, E)$ reduced under Rules 2.2 to 2.5. Assume $|B| > k^3 + 4k^2 + 9k + 4$. Since $G$ is reduced under Rule 2.4, we know by Observation 2.10 that the attachment clique $B_1$ of $B$, which is a $K$-join, contains at most $k^3 + 4k^2 + 7k + 3$ vertices. This implies that $|B^R| > 2k + 1$, which cannot be since $G$ is reduced under Rule 2.5. □

### 2.1.5 Cutting the 2-branches

To obtain a rule reducing the 2-branches, we need to introduce a particular decomposition of 2-branches into $K$-joins. Let $B$ be a 2-branch with an umbrella ordering $\sigma_B = b_1, \ldots, b_{|B|}$. As usual, we denote by $B_1 = b_1, \ldots, b_{l'}$ its first attachment clique and by $B_2 = b_l, \ldots, b_{|B|}$ its second. The reversal of the permutation $\sigma_B$ gives a second possibility to fix $B_1$ and $B_2$. We fix one of these possibilities and define $\mathcal{B}$, the *$K$-join decomposition* of $B$. The $K$-joins of $\mathcal{B}$ are defined by $B'_i = b_{l_{i-1}+1}, \ldots, b_{l_i}$ where $b_{l_i}$ is the neighbor of $b_{l_{i-1}+1}$ with maximal index. The first $K$-join of $\mathcal{B}$ is $B_1$ (so, $l_0 = 0$ and $l_1 = l'$), and once $B'_{i-1}$ is defined, we set $B'_i$: if $b_{l_{i-1}+1} \in B_2$, then we set $B'_i = b_{l_{i-1}+1}, \ldots, b_{|B|}$, otherwise, we choose $B'_i = b_{l_{i-1}+1}, \ldots, b_{l_i}$ (see Figure 11).

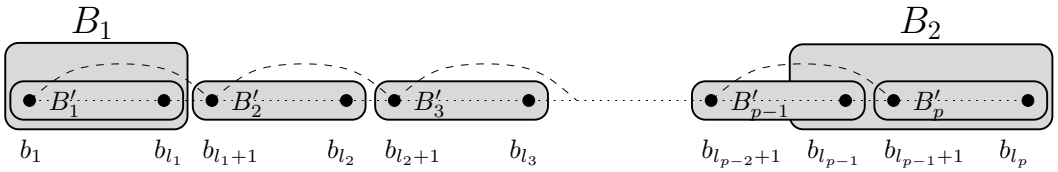


Figure 11: The $K$-join decomposition.

Now, we can prove the next lemma, which bounds the number of $K$-joins in the $K$-join decomposition of a 2-branch providing that some connectivity assumption holds.

**Lemma 2.16.** *Let $G = (V, E)$ be an instance of* Proper Interval Completion *and $B$ be a 2-branch containing $p \geq (k+4)$ $K$-joins in its $K$-join decomposition. Assume the attachment*

*cliques $B_1$ and $B_2$ of $B$ belong to the same connected component of $G[V \setminus B^R]$ . Then there is no $k$-completion for $G$.*

*Proof.* Let $B$ be a 2-branch of an instance $G = (V, E)$ of Proper Interval Completion respecting the conditions of Lemma 2.16. Since $B_1$ and $B_2$ belong to the same connected component in $G[V \setminus B^R]$, let $\pi$ be a shortest path between $B_1$ and $B_2$ in $G[V \setminus B^R]$. As $B$ has $p \geq k+4 \geq 3$ $K$-joins in its decomposition, no vertex of $B_1$ is adjacent to a vertex f $B_2$ and $\pi$ has length at least two. We denote by $u \in B_1$ and $v \in B_2$ the extremities of such a path. We now construct an induced path $P_{uv}$ of length at least $p-1$ between $u$ and $v$ within $B$. To do so, considering the $K$-join decomposition $\mathcal{B} = \{B'_1, \ldots, B'_p\}$ of $B$, we know that $u \in B'_1$ and that $v \in B'_{p-1} \cup B'_p$. We define $u_1 = u$ and while $v \notin N[u_i]$, we choose $u_{i+1}$ the neighbor of $u_i$ with maximum index in the umbrella ordering of $B$. In this case, we have $u_i \in \cup_{j=1}^{i} B'_j$ for every $1 \leq i \leq p$. Indeed, the neighbor of a vertex of $\cup_{j=1}^{i-1} B'_j$ with maximum index in the umbrella ordering of $B$ is in $\cup_{j=1}^{i} B'_j$. Finally, when $v \in N[u_i]$, we just choose $u_{i+1} = v$. So, the path $P_{uv} = u_1, \ldots, u_l$ is an induced path of length at least $p-1$, with $u_1 = u$, $u_l = v$ and the only vertices that could have neighbors in $G \setminus B$ are $u_1, u_{l-1}$ and $u_l$ ($u_1 \in B_1$, $u_l \in B_2$ and $u_{l-1}$ is possibly in $B_2$). Using $\pi$, we can form an induced cycle of length at least $p \geq k+4$ in $G$. Since at least $q-3$ completions are needed to triangulate any induced cycle of length $q$ [14], it follows that there is no $k$-completion for $G$. □

The following observation is a straightforward implication of Lemma 2.16.

**Observation 2.17.** *Let $G = (V, E)$ be a connected positive instance of* Proper Interval Completion*, reduced by Rule 2.4 and $B$ be a 2-branch such that $G[V \setminus B^R]$ is connected. Then $B$ contains at most $k+3$ $K$-joins in its $K$-join decomposition and hence at most $(k+3)(k^3+4k^2+5k+1)$ vertices.*

**Rule 2.6** (2-branches)**.** *Let $G$ be a connected graph and $B$ be a 2-branch such that $G[V \setminus B^R]$ is not connected. Assume that $|B^R| \geq 4(k+1)$ and let $B'_1$ be the $2k+1$ vertices after $B_1$ and $B'_2$ the $2k+1$ vertices before $B_2$. Remove $B \setminus (B_1 \cup B'_1 \cup B'_2 \cup B_2)$ from $G$.*

**Lemma 2.18.** *Rule 2.6 is safe.*

*Proof.* As usual, we denote by $\sigma_b = b_1, \ldots, b_{|B|}$ the umbrella ordering defined on $B$, with $B_1 = \{b_1, \ldots, b_{l'}\}$ and $B_2 = \{b_l, \ldots, b_{|B|}\}$. We partition $B^R$ into two sets $B' = \{b_{l'+1}, \ldots, b_i\}$ and $B'' = \{b_{i+1}, \ldots, b_{l-1}\}$ such that $|B'| \geq |B''| \geq 2k+1$. We now remove the edges $E(B', B'')$ between $B'$ and $B''$, obtaining two connected components of $G$, $G_1$ and $G_2$. Observe that $B'$ defines a 1-branch of $G_1$ with attachment clique $B_1$ such that $B' \setminus B_1$ contains at least $2k+1$ vertices. Similarly $B''$ defines a 1-branch of $G_2$ with attachment clique $B_2$ such that $B'' \setminus B_2$ contains at least $2k+1$ vertices. Hence Lemma 2.11 can be applied to both $G_1$ and $G_2$ and we continue as if Rule 2.5 has been applied to $G_1$ and $G_2$, preserving exactly $2k+1$ vertices $B'_f$ and $B''_f$, respectively. We denote by $G'$ the reduced graph. Let $F$ be a $k$-completion of $G$. Let $F_1$ and $F_2$ be the completions of $G_1$ and $G_2$ such that $|F_1| + |F_2| \leq k$. Moreover, let $H_1 = G_1 + F_1$ and $H_2 = G_2 + F_2$. By Lemma 2.11, we know that the vertices of $B' \setminus B'_f$ (resp. $B'' \setminus B''_f$) can be inserted into the umbrella ordering $\sigma_{H_1}$ of $H_1$ (resp. $\sigma_{H_2}$) in the same order than in $B'$ (resp. $B''$). We thus obtain two proper interval graphs $H'_1$ and $H'_2$ whose respective umbrella ordering preserve the order of $B'$ and $B''$. We now connect $H'_1$ and $H'_2$ by putting back the edges contained in $E(B', B'')$, obtaining a graph $H$ with ordering $\sigma_H$. Since $G[B]$ is a proper interval graph and $B'$ and $B''$ are ordered according to $B$ in $H'_1$ and $H'_2$ , it follows that $H$ is a proper interval graph, and hence $F = F_1 \cup F_2$ is a $k$-completion of $G$. □

**Observation 2.19.** *Let $G = (V, E)$ be a positive instance of* Proper Interval Completion *reduced under Rules 2.2 to 2.6. The 2-branches of $G$ contain at most $(k + 3)(k^3 + 4k^2 + 5k + 1)$ vertices.*

*Proof.* Let $B$ be a 2-branch of a graph $G = (V, E)$ and $C$ be the connected component containing $B$. If $G[C \setminus B^R]$ is connected, then Observation 2.17 implies the result. Otherwise, as $G$ has been reduced under Rules 2.2 to 2.6, we know that $|B^R| \leq 4k + 4$ and then that $|B| \leq 2(k^3 + 4k^2 + 5k + 1) + (4k + 4)$ which is less than $(k + 3)(k^3 + 4k^2 + 5k + 1)$, provided that $k \geq 1$. □

## 2.2 Detecting the branches

We now turn our attention to the complexity needed to compute reduction rules 2.4 to 2.6. Mainly, we indicate how to obtain the maximum branches in order to reduce them. The detection of a branch is straightforward except for the attachment cliques, where several choices are possible.
So, first, we detect the maximum 1-branches of $G$. Remark that for every vertex $x$ of $G$, the set $\{x\}$ is a 1-branch of $G$. The next lemma indicates how to compute a maximum 1-branch that contains a fixed vertex $x$ as first vertex.

**Lemma 2.20.** *Let $G = (V, E)$ be a graph and $x$ a vertex of $G$. In time $O(n^2)$, it is possible to detect a maximum 1-branch of $G$ containing $x$ as first vertex.*

*Proof.* To detect such a 1-branch, we design an algorithm which has two parts. Roughly speaking, we first try to detect the set $B^R$ of a 1-branch $B$ containing $x$. We set $B_0^R = \{x\}$ and $\sigma_0 = x$. Once $B_{i-1}^R$ has been defined, we construct the set $C_i$ of vertices of $G \setminus (\cup_{l=1}^{i-1} B_l^R)$ that are adjacent to at least one vertex of $B_{i-1}^R$. Two cases can appear. First, assume that $C_i$ is a clique and that it is possible to order the vertices of $C_i$ such that for every $1 \leqslant j < |C_i|$, we have $N_{B_{i-1}^R}(c_{j+1}) \subseteq N_{B_{i-1}^R}(c_j)$ and $(N_G(c_j) \setminus B_{i-1}^R) \subseteq (N_G(c_{j+1}) \setminus B_{i-1}^R)$. In this case, the vertices of $C_i$ correspond to a new $K$-join of the searched 1-branch (remark that, along this inductive construction, there is no edge between $C_i$ and $\cup_{l=1}^{i-2} B_l^R$). So, we let $B_i^R = C_i$ and $\sigma_i$ be the concatenation of $\sigma_{i-1}$ and the ordering defined on $C_i$. In the other case, such an ordering of $C_i$ can not be found, meaning that while detecting a 1-branch $B$, we have already detected the vertices of $B^R$ and at least one (possibly more) vertex of the attachment clique $B_1$ with neighbors in $B^R$. Assume that the process stops at step $p$ and let $C$ be the set of vertices of $G \setminus \cup_{l=1}^{p} B_l^R$ which have neighbors in $\cup_{l=1}^{p} B_l^R$ and $B_1' \subseteq B_p^R$ be the set of vertices that are adjacent to all the vertices of $C$. Remark that $B_1' \neq \emptyset$, as $B_1'$ contains at least the last vertex of $\sigma_p$. We denote by $B^R$ the set $(\cup_{l=1}^{p} B_l^R) \setminus B_1'$ and we will construct the largest $K$-join containing $B_1'$ in $G \setminus B^R$ which is compatible with $\sigma_p$, in order to define the attachment clique $B_1$ of the desired 1-branch. The vertices of $C$ are the candidates to complete the attachment clique. On $C$, we define the following oriented graph: there is an arc from $x$ to $y$ if: $xy$ is an edge of $G$, $N_{B^R}(y) \subseteq N_{B^R}(x)$ and $N_{G \setminus B^R}[x] \subseteq N_{G \setminus B^R}[y]$. This graph can be computed in time $O(n^2)$. Now, it is easy to check that the obtained oriented graph is a transitive graph, in which the equivalent classes are made of true twins in $G$. A path in this oriented graph corresponds, by definition, to a $K$-join containing $B_1'$ and compatible with $\sigma_p$. As it is possible to compute a longest path in linear time in this oriented graph, we obtain a maximum 1-branch of $G$ that contains $x$ as first vertex. □

Now, to detect the 2-branches, we first detect for all pairs of vertices a maximum $K$-join with these vertices as ends. More precisely, if $\{x, y\}$ are two vertices of $G$ linked by an edge, then $\{x, y\}$ is a $K$-join of $G$, with $N = N_G(x) \cap N_G(y)$, $L = N_G(x) \setminus N_G[y]$ and $R = N_G(y) \setminus N_G[x]$. So, there exist $K$-joins with $x$ and $y$ as ends, and we will compute such a $K$-join with maximum cardinality.

**Lemma 2.21.** *Let $G = (V, E)$ be a graph and $x$ and $y$ two adjacent vertices of $G$. It is possible to compute in cubic time a maximum (in cardinality) $K$-join that admits $x$ and $y$ as ends.*

*Proof.* We denote $N_G[x] \cap N_G[y]$ by $N$, $N_G(x) \setminus N_G[y]$ by $L$ and $N_G(y) \setminus N_G[x]$ by $R$. Let us denote by $N'$ the set of vertices of $N$ that contains $N$ in their closed neighborhood. The vertices of $N'$ are the candidates to belong to the desired $K$-join. Now, we construct on $N'$ an oriented graph, putting, for every vertices $u$ and $v$ of $N'$, an arc from $u$ to $v$ if: $N_G(v) \cap L \subseteq N_G(u) \cap L$ and $N_G(u) \cap R \subseteq N_G(v) \cap R$. Basically, it could take a $O(n)$ time to decide if there is an arc from $u$ to $v$ or not, and so the whole oriented graph could be computed in time $O(n^3)$. Now, it is easy to check that the obtained oriented graph is a transitive graph in which the equivalent classes are made of true twins in $G$. In this oriented graph, it is possible to compute a longest path from $x$ to $y$ in linear time. Such a path corresponds to a maximal $K$-join that admits $x$ and $y$ as ends. It follows that the desired $K$-join can be identified in $O(n^3)$ time. □

Now, for every edge $xy$ of $G$, we compute a maximum $K$-join that contains $x$ and $y$ as ends and a reference to all the vertices that this $K$-join contains. This computation takes a $O(n^3m)$ time and gives, for every vertex, some maximum $K$-joins that contain this vertex. These $K$-joins will be useful to compute the 2-branches of $G$, in particular through the next lemma.

**Lemma 2.22.** *Let $B$ be a 2-branch of $G$ with $B^R \neq \emptyset$, and $x$ a vertex of $B^R$. Then, for every maximal (by inclusion) $K$-join $B'$ that contains $x$ there exists an extremal edge $uv$ of $\sigma_B$ such that $B' = \{w \in B \ : \ u \leq_{\sigma_B} w \leq_{\sigma_B} v\}$.*

*Proof.* As usually, we denote by $L$, $R$ and $C$ the partition of $G \setminus B$ associated with $B$ and by $\sigma_B$ the umbrella ordering associated with $B$. Let $B'$ be a maximal $K$-join that contains $x$ and define by $b_f$ (resp. $b_l$) the first (resp. last) vertex of $B'$ according to $\sigma_B$. As there is no edge between $\{u \in B \ : \ u <_{\sigma_B} b_f\} \cup L \cup C$ and $b_l$ and no edge between $\{u \in B \ : \ b_l <_{\sigma_B} u\} \cup R \cup C$ and $b_f$, we have $B' \subseteq \{u \in B \ : \ b_f \leq_{\sigma_B} u \leq b_l\}$. Furthermore, as $\{u \in B \ : \ b_f \leq_{\sigma_B} u \leq b_l\}$ is a $K$-join and $B'$ is maximal, we have $B' = \{u \in B \ : \ b_f \leq_{\sigma_B} u \leq b_l\}$. Now, if $b_f b_l$ was not an extremal edge of $\sigma_B$, it would be possible to extend $B'$, contradicting the maximality of $B'$. □

Now, we can detect the 2-branches $B$ with a set $B^R$ non empty. Observe that this is enough for our purpose since we want to detect 2-branches of size at least $(k+3)(k^3+4k^2+5k+1)$ and the attachment cliques contain at most $2(k^3+4k^2+7k+3)$ vertices.

**Lemma 2.23.** *Let $G = (V, E)$ be a graph, $x$ a vertex of $G$ and $B'$ a given maximal $K$-join that contains $x$. There is a quadratic time algorithm to decide if there exists a 2-branch $B$ of $G$ which contains $x$ as a vertex of $B^R$, and if it exists, to find a maximum 2-branch with this property.*

*Proof.* By Lemma 2.22, if there exists a 2-branch $B$ of $G$ which contains $x$ as a vertex of $B^R$, then $B'$ corresponds to a set $\{u \in B \ : \ b_f \leq_{\sigma_B} u \leq_{\sigma_B} b_l\}$ where $b_f b_l$ is an extremal edge of $B$. We denote by $L'$, $R'$ and $C'$ the usual partition of $G \setminus B'$ associated with $B'$, and by $\sigma_{B'}$ the umbrella ordering of $B'$. In $G$, we remove the set of vertices $\{u \in B' \ : \ u <_{\sigma_{B'}} x\}$ and the edges between $L'$ and $\{u \in B' \ : \ x \leq_{\sigma_{B'}} u\}$ and denote by $H_1$ the resulting graph. From the definition of the 2-branch $B$, $\{u \in B \ : \ x \leq_{\sigma_B} u\}$ is a 1-branch of $H_1$ that contains $x$ as first vertex. So, using Lemma 2.20, we find a maximal 1-branch $B_1$ that contains $x$ as first vertex. Remark that $B_1$ has to contain $\{u \in B \ : \ x \leq_{\sigma_B} u\} \cap B^R$ at its beginning. Similarly, we define $H_2$ from $G$ by removing the vertex set $\{u \in B' \ : \ x <_{\sigma_{B'}} u\}$ and the edges between $R'$ and $\{u \in B' \ : \ u \leq_{\sigma_{B'}} x\}$. We detect

in $H_2$ a maximum 1-branch $B_2$ that contains $x$ as last vertex, and as previously, $B_2$ has to contain $\{u \in B \ : \ u \leq_{\sigma_B} x\} \cap B^R$ at its end. So, $B_1 \cup B_2$ forms a maximum 2-branch of $G$ containing $x$. □

We would like to mention that it could be possible to improve the execution time of our detecting branches algorithm, using possibly more involved techniques (as for instance, inspired from [7]). However, this is not our main objective here.
Anyway, using a $O(n^4)$ brute force detection to localize all the 4-cycles and the claws, we obtain the following result.

**Lemma 2.24.** *Given a graph $G = (V, E)$, the reduction rules 2.4 to 2.6 can be carried out in polynomial time, namely in time $O(n^3 m)$.*

## 2.3 Kernelization algorithm

We are now ready to the state the main result of this Section. The kernelization algorithm consists of an exhaustive application of Rules 2.1 to 2.6.

**Theorem 2.25.** *The* Proper Interval Completion *problem admits a kernel with $O(k^5)$ vertices.*

*Proof.* Let $G = (V, E)$ be a positive instance of Proper Interval Completion reduced under Rules 2.1 to 2.6. Let $F$ be a $k$-completion of $G$, $H = G + F$ and $\sigma_H$ be the umbrella ordering of $H$. Since $|F| \leq k$, $G$ contains at most $2k$ *affected* vertices (i.e. incident to an added edge). Let $A = \{a_1 <_{\sigma_H} \ldots <_{\sigma_H} a_i <_{\sigma_H} \ldots <_{\sigma_H} a_p\}$ be the set of such vertices, with $p \leq 2k$. The size of the kernel is due to the following observations (see Figure 12):

- Let $L_0 = \{l \in V \ : \ l <_{\sigma_H} a_1\}$ and $R_{p+1} = \{r \in V \ : \ a_p <_{\sigma_H} r\}$. Since the vertices of $L_0$ and $R_{p+1}$ are not affected, it follows that $G[L_0]$ and $G[R_{p+1}]$ induce a proper interval graph. As Rule 2.1 has been applied, $G[L_0]$ and $G[R_{p+1}]$ both contain one connected component, and $L_0$ and $R_{p+1}$ are 1-branches of $G$. So, by Observation 2.15, $L_0$ and $R_{p+1}$ both contain at most $k^3 + 4k^2 + 9k + 4$ vertices.

- Let $S_i = \{s \in V \ : \ a_i <_{\sigma_H} s <_{\sigma_H} a_{i+1}\}$ for every $1 \leq i < p$. Again, since the vertices of $S_i$ are not affected, it follows that $G[S_i]$ is a proper interval graph. As Rule 2.1 as been applied, there are at most two connected components in $G[S_i]$. If $G[S_i]$ is connected, then, $S_i$ is a 2-branch of $G$ and, by Observation 2.19, $S_i$ contains at most $(k+3)(k^3+4k^2+5k+1)$ vertices. Otherwise, if $G[S_i]$ contains two connected components, they correspond to two 1-branches of $G$, and by Observation 2.15, $S_i$ contain at most $2(k^3 + 4k^2 + 9k + 4)$ vertices. In both cases, we bound the number of vertices of $S_i$ by $(k + 3)(k^3 + 4k^2 + 5k + 1)$, provided that $k \geq 1$.

Altogether, the proper interval graph $H$ (and hence $G$) contains at most:

$$2(k^3 + 4k^2 + 9k + 4) + (2k - 1)((k + 3)(k^3 + 4k^2 + 5k + 1))$$

vertices, which implies the claimed $O(k^5)$ bound. The complexity directly follows from Lemma 2.24. □

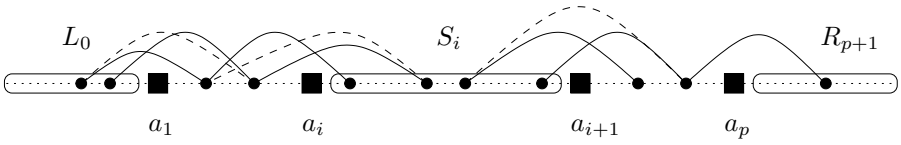


Figure 12: Illustration of the size of the kernel. The figure represents the graph $H = G + F$, the square vertices stand for the *affected* vertices, $L_0$ and $R_{p+1}$ are 1-branches of $G$, and, on the figure, $S_i$ defines a 2-branch.

# 3 A special case: Bi-clique Chain Completion

*Bipartite chain graphs* are defined as bipartite graphs whose parts are connected by a join. Equivalently, they are known to be the graphs that do not admit any $\{2K_2, C_5, K_3\}$ as an induced subgraph [29] (see Figure 13). In [12], Guo proved that the so-called Bipartite Chain Deletion With Fixed Bipartition problem, where one is given a *bipartite* graph $G = (V, E)$ and seeks a subset of $E$ of size at most $k$ whose deletion from $E$ leads to a bipartite chain graph, admits a kernel with $O(k^2)$ vertices. We define *bi-clique chain graph* to be the graphs formed by two disjoint cliques linked by a join. They correspond to interval graphs that can be covered by two cliques. Since the complement of a bipartite chain graph is a bi-clique chain graph, this result also holds for the Bi-clique Chain Completion With Fixed Bi-clique Partition problem. Using similar techniques than in Section 2, we prove that when the bipartition is not fixed, both problems admit a quadratic-vertex kernel. For the sake of simplicity, we consider the completion version of the problem, defined as follows.

Bi-clique Chain Completion:
**Input**: A graph $G = (V, E)$ and a positive integer $k$.
**Parameter**: $k$.
**Output**: A set $F \subseteq (V \times V) \setminus E$ of size at most $k$ such that the graph $H = (V, E \cup F)$ is a bi-clique chain graph.

It follows from definition that bi-clique chain graphs do not admit any $\{C_4, C_5, 3K_1\}$ as an induced subgraph, where a $3K_1$ is an independent set of size 3 (see Figure 13). Observe in particular that bi-clique chain graphs are proper interval graphs, and hence admit an umbrella ordering.

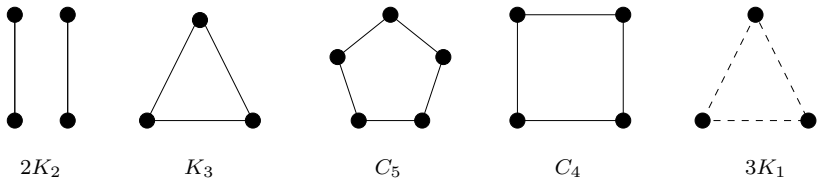


Figure 13: The forbidden induced subgraphs for bipartite and bi-clique chain graphs.

We provide a kernelization algorithm for the Bi-clique Chain Completion problem which follows the same lines that the one in Section 2.

**Rule 3.1** (Sunflower)**.** *Let $\mathcal{S} = \{C_1, \ldots, C_m\}$, $m > k$ be a set of $3K_1$ having two vertices $u, v$ in common but distinct third vertex. Add $uv$ to $F$ and decrease $k$ by* 1.

*Let $\mathcal{S} = \{C_1, \ldots, C_m\}$, $m > k$ be a set of distinct 4-cycles having a non-edge $uv$ in common. Add $uv$ to $F$ and decrease $k$ by 1.*

The following result is similar to Lemma 2.2.

**Lemma 3.1.** *Let $G = (V, E)$ be a positive instance of* Bi-clique Chain Completion *on which Rule 3.1 has been applied. There are at most $k^2+2k$ vertices of $G$ contained in $3K_1$'s. Furthermore, there at most $2k^2 + 2k$ vertices of $G$ that are vertices of a 4-cycle.*

We say that a $K$-join is *simple* whenever $L = \emptyset$ or $R = \emptyset$. In other words, a simple $K$-join consists in a clique connected to the rest of the graph by a join. We will see it as a 1-branch which is a clique and use for it the classical notation devoted to the 1-branch. Moreover, we (re)define a *clean $K$-join* as a $K$-join whose vertices do not belong to any $3K_1$ or 4-cycle. The following reduction rule is similar to Rule 2.4, the main ideas are identical, only some technical arguments change. Anyway, to be clear, we give the proof in all details.

**Rule 3.2** ($K$-join)**.** *Let $B$ be a simple clean $K$-join of size at least $2(k + 1)$ associated with an umbrella ordering $\sigma_B$. Let $B_L$ (resp. $B_R$) be the $k + 1$ first (resp. last) vertices of $B$ according to $\sigma_B$, and $M = B \setminus (B_L \cup B_R)$. Remove the set of vertices $M$ from $G$.*

**Lemma 3.2.** *Rule 3.2 is safe and can be computed in polynomial time.*

*Proof.* Let $G' = G \setminus M$. Observe that any $k$-completion of $G$ is a $k$-completion of $G'$ since bi-clique chain graphs are closed under induced subgraphs. So, let $F$ be a $k$-completion for $G'$. We denote by $H = G' + F$ the resulting bi-clique chain graph and by $\sigma_H$ an umbrella ordering of $H$. We prove that we can always insert the vertices of $M$ into $\sigma_H$ and modify it if necessary, to obtain an umbrella ordering of a bi-clique chain graph for $G$ without adding any edge. This will imply that $F$ is a $k$-completion for $G$. To see this, we need the following structural property of $G$. As usual, we denote by $R$ the neighbors in $G \setminus B$ of the vertices of $B$, and by $C$ the vertices of $G \setminus (R \cup B)$. For the sake of simplicity, we let $N = \cap_{b \in B} N_G(b) \setminus B$, and remove the vertices of $N$ from $R$. We abusively still denote by $R$ the set $R \setminus N$, see Figure 14.

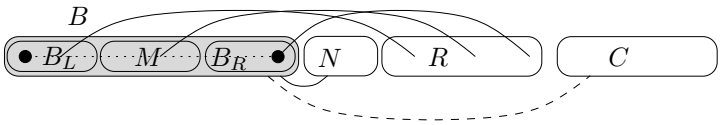


Figure 14: The $K$-join decomposition for the Bi-clique Chain Completion problem.

**Claim 3.3.** *The set $R \cup C$ is a clique of $G$.*

*Proof.* Observe that no vertex of $R$ is a neighbor of $b_1$, since otherwise such a vertex must be adjacent to all the vertices of $B$ and then must stand in $N$. So, if $R \cup C$ contains two vertices $u, v$ such that $uv \notin E$, we form the $3K_1$ $\{b_1, u, v\}$, contradicting the fact that $B$ is clean. ◇

The following observation comes from the definition of a simple $K$-join.

**Observation 3.4.** *Given any vertex $r \in R$, if $N_B(r) \cap B_L \neq \emptyset$ holds then $M \subseteq N_B(r)$.*

We use these facts to prove that an umbrella ordering of a bi-clique chain graph can be obtained for $G$ by inserting the vertices of $M$ into $\sigma_H$. Let $b_f, b_l$ be the first and last vertex of $B \setminus M$ appearing in $\sigma_H$, respectively. We let $B_H$ denote the set $\{u \in V(H) \ : \ b_f <_{\sigma_H} u <_{\sigma_H} b_l\}$. Now, we modify $\sigma_H$ by ordering the twins in $H$ according to their neighborhood in $M$: if $x$ and $y$ are twins in $H$, are consecutive in $\sigma_H$, verify $x <_{\sigma_H} y <_{\sigma_H} b_f$ and $N_M(y) \subset N_M(x)$, then we exchange $x$ and $y$ in $\sigma_H$. This process stops when the considered twins are ordered following the join between $\{u \in V(H) \ : \ u <_{\sigma_H} b_f\}$ and $M$. We proceed similarly on the right of $B_H$, i.e. for $x$ and $y$ consecutive twins with $b_l <_{\sigma_H} x <_{\sigma_H} y$ and $N_M(x) \subset N_M(y)$. The obtained order is clearly an umbrella ordering of a bi-clique chain graph too (in fact, we just re-labeled some vertices in $\sigma_H$, and we abusively still denote it by $\sigma_H$).

**Claim 3.5.** *The set $B_H \cup \{m\}$ is a clique of $G$ for any $m \in M$, and consequently $B_H \cup M$ is a clique of $G$.*

*Proof.* Let $u$ be any vertex of $B_H$. We claim that $um \in E(G)$. Observe that if $u \in B$ then the claim trivially holds. So, assume that $u \notin B$. By definition of $\sigma_H$, $B_H$ is a clique in $H$ since $b_f b_l \in E(G)$. It follows that $u$ is incident to every vertex of $B \setminus H$ in $H$. Since $B_L$ contains $k+1$ vertices, it follows that $N_G(u) \cap B_L \neq \emptyset$. Hence, $u$ belongs to $N \cup R$ and $um \in E$ by Observation 2.6. $\diamond$

**Claim 3.6.** *Let $m$ be any vertex of $M$ and $\sigma'_H$ be the ordering obtained from $\sigma_H$ by removing $B_H$ and inserting $m$ to the position of $B_H$. The ordering $\sigma'_H$ respects the umbrella property.*

*Proof.* Assume that $\sigma'_H$ does not respect the umbrella property, i.e. that there exist (w.l.o.g.) two vertices $u, v \in H \setminus B_H$ such that either (1) $u <_{\sigma'_H} v <_{\sigma'_H} m$, $um \in E(H)$ and $uv \notin E(H)$ or (2) $u <_{\sigma'_H} m <_{\sigma'_H} v$, $um \notin E(H)$ and $uv \in E(H)$ or (3) $u <_{\sigma'_H} v <_{\sigma'_H} m$, $um \in E(H)$ and $vm \notin E(H)$. First, assume that (1) holds. Since $uv \notin E$ and $\sigma_H$ is an umbrella ordering, $uw \notin E(H)$ for any $w \in B_H$, and hence $uw \notin E(G)$. This means that $B_R \cap N_G(u) = \emptyset$, which is impossible since $um \in E(G)$. If (2) holds, since $uv \in E(H)$ and $\sigma_H$ is an umbrella ordering of $H$, we have $B_H \subseteq N_H(u)$. In particular, $B_L \subseteq N_H(u)$ holds, and as $|B_L| = k+1$, we have $B_L \cap N_G(u) \neq \emptyset$ and $um$ should be an edge of $G$, what contradicts the assumption $um \notin E(H)$. So, (3) holds, and we choose the first $u$ satisfying this property according to the order given by $\sigma'_H$. So we have $wm \notin E(G)$ for any $w <_{\sigma'_H} u$. Similarly, we choose $v$ to be the first vertex satisfying $vm \notin E(G)$. Since $um \in E(G)$, we know that $u$ belongs to $N \cup R$. Moreover, since $vm \notin E(G)$, $v \in R \cup C$. There are several cases to consider:

(i) $u \in N$: in this case we know that $B \subseteq N_G(u)$, and in particular that $ub_l \in E(G)$. Since $\sigma_H$ is an umbrella ordering for $H$, it follows that $vb_l \in E(H)$ and that $B_L \subseteq N_H(v)$. Since $|B_L| = k+1$ we know that $N_G(v) \cap B_L \neq \emptyset$ and hence $v \in R$. It follows from Observation 2.6 that $vm \in E(G)$.

(ii) $u \in R, v \in R \cup C$: in this case $uv \in E(G)$, by Claim 3.3, but $u$ and $v$ are not true twins in $H$ (otherwise $v$ would be placed before $u$ in $\sigma_H$ due to the modification we have applied to $\sigma_H$). This means that there exists a vertex $w \in V(H)$ that *distinguishes* $u$ from $v$ in $H$.

Assume first that $w <_{\sigma_H} u$ and that $uw \in E(H)$ and $vw \notin E(H)$. We choose the first $w$ satisfying this according to the order given by $\sigma'_H$. Since $vm, wm, vw \notin E(H)$, it follows that $\{v, w, m\}$ defines a $3K_1$ of $G$, which cannot be since $B$ is clean. Hence we can assume that

for any $w'' <_{\sigma_H} u$, $uw'' \in E(H)$ implies that $vw'' \in E(H)$. Now, suppose that $b_l <_{\sigma_H} w$ and $uw \notin E(H)$, $vw \in E(H)$. In particular, this means that $B_L \subseteq N_H(v)$. Since $|B_L| = k+1$ we have $N_G(v) \cap B_L \neq \emptyset$, implying $vm \in E(G)$ (Observation 2.6). Assume now that $v <_{\sigma_H} w <_{\sigma_H} b_f$. In this case, since $uw \notin E(H)$, $B \cap N_H(u) = \emptyset$ holds and hence $B \cap N_G(u) = \emptyset$, which cannot be since $u \in R$. Finally, assume that $w \in B_H$ and choose the last vertex $w$ satisfying this according to the order given by $\sigma'_H$ (i.e. $vw' \notin E(H)$ for any $w <_{\sigma_H} w'$ and $w' \in B_H$). If $vw \in E(G)$ then $\{u, m, w, v\}$ is a 4-cycle in $G$ containing a vertex of $B$, which cannot be (recall that $B_H \cup \{m\}$ is a clique of $G$ by Claim 2.7). Hence $vw \in F$ and there exists an extremal edge above $vw$. The only possibility is that this edge is some edge $u'w$ for some $u'$ with $u' \in V(H)$, $u <_{\sigma_H} u' <_{\sigma_H} v$ and $u'w \in E(G)$. By the choice of $v$ we know that $u'm \in E(G)$. Moreover, by the choice of $w$, observe that $u'$ and $v$ are true twins in $H$ (if a vertex $s$ distinguishes $u'$ and $v$ in $H$, $s$ cannot be before $u$, since otherwise $s$ would distinguish $u$ and $v$, and not before $w$, by choice of $w$). This leads to a contradiction because $v$ should have been placed before $u$ through the modification we have applied to $\sigma_H$. ◇

**Claim 3.7.** *Every vertex $m \in M$ can be added to the graph $H$ while preserving an umbrella ordering.*

*Proof.* Let $m$ be any vertex of $M$. The graph $H$ is a bi-clique chain graph. So, we know that in its associated umbrella ordering $\sigma_H = b_1, \ldots, b_{|H|}$, there exists a vertex $b_i$ such that $H_1 = \{b_1, \ldots, b_i\}$ and $H_2 = \{b_{i+1}, \ldots, b_{|H|}\}$ are two cliques of $H$ linked by a join. We study the behavior of $B_H$ according to the partition $(H_1, H_2)$.

(i) Assume first that $B_H \subseteq H_1$ (the case $B_H \subseteq H_2$ is similar). We claim that the set $H_1 \cup \{m\}$ is a clique. Indeed, let $v \in H_1 \setminus B_H$: since $H_1$ is a clique, $B_H \subseteq N_H(v)$ and hence $N_G(v) \cap B_L \neq \emptyset$. In particular, this means that $vm \in E(G)$ by Observation 3.4. Since $B_H \cup \{m\}$ is a clique by Claim 3.5, the result follows. Now, let $u$ be the neighbor of $m$ with maximal index in $\sigma_H$, and $b_u$ the neighbor of $u$ with minimal index in $\sigma_H$. Observe that we may assume $u \in H_2$ since otherwise $N_H(m) \cap H_2 = \emptyset$ and hence we insert $m$ at the beginning of $\sigma_H$. First, if $b_u \in H_1$, we prove that the order $\sigma_m$ obtained by inserting $m$ directly before $b_u$ in $\sigma_H$ yields an umbrella ordering of a bi-clique chain graph. Since $H_1 \cup \{m\}$ is a clique, we only need to show that $N_{H_2}(v) \subseteq N_{H_2}(m)$ for any $v \leq_{\sigma_m} b_u$ and $N_{H_2}(m) \subseteq N_{H_2}(w)$ for any $w \in H_2$ with $w \geq_{\sigma_m} b_u$. Observe that by Claim 3.6 the set $\{w \in V \ : \ m \leq_{\sigma_m} w \leq_{\sigma_m} u\}$ is a clique. Hence the former case holds since $vu' \notin E(G)$ for any $v \leq_{\sigma_m} b_u$ and $u' \geq_{\sigma_m} u$. The latter case also holds since $N_H(m) \subseteq N_H(b_u)$ by construction. Finally, if $b_u \in H_2$, then $b_u = b_{|H_1|+1}$ since $H_2$ is a clique. Hence, using similar arguments one can see that inserting $m$ directly after $b_{|H_1|}$ in $\sigma_H$ yields an umbrella ordering of a bi-clique chain graph.

(ii) Assume now that $B_H \cap H_1 \neq \emptyset$ and $B_H \cap H_2 \neq \emptyset$. In this case, we claim that $H_1 \cup \{m\}$ or $H_2 \cup \{m\}$ is a clique in $H$. Let $u$ and $u'$ be the neighbors of $m$ with minimal and maximal index in $\sigma_H$, respectively. If $u = b_1$ or $u' = b_{|H|}$ then Claims 3.5 and 3.6 imply that $H_1 \cup \{m\}$ or $H_2 \cup \{m\}$ is a clique and we are done. So, none of these two conditions hold and $mb_1 \notin E(H)$ and $mb_{|H|} \notin E(H)$ Then, by Claim 3.6, we know that $b_1 b_{|H|}$ and the set $\{b_1, b_{|H|}, m\}$ defines a $3K_1$ containing $m$ in $G$, which cannot be. This means that we can assume w.l.o.g. that $H_1 \cup \{m\}$ is a clique, and we can conclude using similar arguments than in (i).

◇

Since the proof of Claim 3.7 does not use the fact that the vertices of $H$ do not belong to $M$, it follows that we can iteratively insert the vertices of $M$ into $\sigma_H$, preserving an umbrella ordering at each step. To conclude, observe that the reduction rule can be computed in polynomial time using Lemma 2.21. □

**Observation 3.8.** *Let $G = (V, E)$ be a positive instance of* Bi-clique Chain Completion *reduced under Rule 3.2. Any simple $K$-join $B$ of $G$ has size at most $3k^2 + 6k + 2$.*

*Proof.* Let $B$ be any simple $K$-join of $G$, and assume $|B| > 3k^2 + 6k + 2$. By Lemma 3.1 we know that at most $3k^2 + 2k$ vertices of $B$ are contained in a $3K_1$ or a 4-cycle. Hence $B$ contains a set $B'$ of at least $2k + 3$ vertices not contained in any $3K_1$ or a 4-cycle. Now, since any subset of a $K$-join is a $K$-join, it follows that $B'$ is a *clean* simple $K$-join. Since $G$ is reduced under rule 3.2, we know that $|B'| \leq 2(k + 1)$ what gives a contradiction. □

Finally, we can prove that Rules 3.1 and 3.2 form a kernelization algorithm.

**Theorem 3.9.** *The* Bi-clique Chain Completion *problem admits a kernel with $O(k^2)$ vertices.*

*Proof.* Let $G = (V, E)$ be a positive instance of Bi-clique Chain Completion reduced under Rules 3.1 and 3.2, and $F$ be a $k$-completion for $G$. We let $H = G + F$ and $H_1$, $H_2$ be the two cliques of $H$. Observe in particular that $H_1$ and $H_2$ both define simple $K$-joins. Let $A$ be the set of affected vertices of $G$. Since $|F| \leq k$, observe that $|A| \leq 2k$. Let $A_1 = A \cap H_1$, $A_2 = A \cap H_2$, $A'_1 = H_1 \setminus A_1$ and $A'_2 = H_2 \setminus A_2$ (see Figure 15). Observe that since $H_1$ is a simple $K$-join in $H$, $A'_1 \subseteq H_1$ is a simple $K$-join of $G$ (recall that the vertices of $A'_1$ are not affected). By Observation 3.8, it follows that $|A'_1| \leq 3k^2 + 6k + 2$. The same holds for $A'_2$ and $H$ contains at most $2(3k^2 + 6k + 2) + 2k$ vertices.

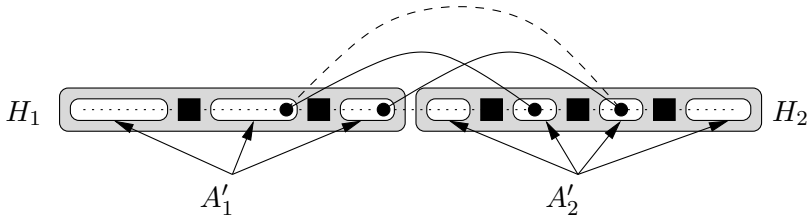


Figure 15: Illustration of the bi-clique chain graph $H$. The square vertices stand for affected vertices, and the sets $A'_1 = H_1 \setminus A_1$ and $A'_2 = H_2 \setminus A_2$ are simple $K$-joins of $G$, respectively.

□

**Corollary 3.10.** *The* Bipartite Chain Deletion *problem admits a kernel with $O(k^2)$ vertices.*

# 4 Conclusion

In this paper we prove that the Proper Interval Completion problem admits a kernel with $O(k^5)$ vertices. Two natural questions arise from our results: firstly, does the Interval Completion problem admit a polynomial kernel? Observe that this problem is known to be FPT not for long [27]. The techniques we developed here intensively use the fact that there are few claws in the graph, what help us to reconstruct parts of the umbrella ordering. Of course, these

considerations no more hold in general interval graphs. The second question is: does the Proper Interval Edge-Deletion problem admit a polynomial kernel? Again, this problem admits a fixed-parameter algorithm [25], and we believe that our techniques could be applied to this problem as well. Finally, we proved that the Bi-clique Chain Completion problem admits a kernel with $O(k^2)$ vertices, which completes a result of Guo [12]. In all cases, a natural question is thus whether these bounds can be improved?

# References


[1] S. Bessy, C. Paul, and A. Perez. Polynomial kernels for 3-leaf power graph modification problems. *Discrete Applied Mathematics*, 158(16):1732–1744, 2010.

[2] H. L. Bodlaender. Kernelization: New upper and lower bound techniques. In *IWPEC*, pages 17–37, 2009.

[3] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. On problems without polynomial kernels. *J. Comput. Syst. Sci*, 75(8):423–434, 2009.

[4] L. Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Inf. Process. Lett*, 58(4):171–176, 1996.

[5] J. Chen and J. Meng. A $2k$ kernel for the cluster editing problem. In *COCOON*, volume 6196 of *LNCS*, pages 459–468, 2010.

[6] Derek G. Corneil. A simple 3-sweep lbfs algorithm for the recognition of unit interval graphs. *Discrete Appl. Math.*, 138:371–379, April 2004.

[7] Derek G. Corneil, Hiryoung Kim, Sridhar Natarajan, Stephan Olariu, and Alan P. Sprague. Simple linear time recognition of unit interval graphs. *Information Processing Letters*, 55(2):99 – 104, 1995.

[8] F. K. H. A. Dehne, M. R. Fellows, F. A. Rosamond, and P. Shaw. Greedy localization, iterative compression, modeled crown reductions: New FPT techniques, an improved algorithm for set splitting, and a novel 2k kernelization for vertex cover. In *IWPEC*, volume 3162 of *LNCS*, pages 271–280, 2004.

[9] R.G. Downey and M.R. Fellows. *Parameterized complexity*. Springer, 1999.

[10] M. C. Golumbic, H. Kaplan, and R. Shamir. On the complexity of DNA physical mapping. *ADVAM: Advances in Applied Mathematics*, 15, 1994.

[11] S. Guillemot, C. Paul, and A. Perez. On the (non-)existence of polynomial kernels for $p_l$-free edge modification problems. In *IPEC*, volume 6478 of *LNCS*, pages 147–157, 2010.

[12] J. Guo. Problem kernels for NP-complete edge deletion problems: Split and related graphs. In *ISAAC*, volume 4835 of *LNCS*, pages 915–926, 2007.

[13] P. Hell, R. Shamir, and R. Sharan. A fully dynamic algorithm for recognizing and representing proper interval graphs. *SIAM Journal on Computing*, 31(1):289–305, 2001.

[14] H. Kaplan, R. Shamir, and R. E. Tarjan. Tractability of parameterized completion problems on chordal and interval graphs: Minimum fill-in and physical mapping. In *FOCS*, pages 780–791, 1994.

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

[16] S. Kratsch and M. Wahlström. Two edge modification problems without polynomial kernels. In *IWPEC*, volume 5917 of *LNCS*, pages 264–275. Springer, 2009.

[17] P. J. Looges and S. Olariu. Optimal greedy algorithms for indifference graphs. *Computers & Mathematics with Applications*, 25(7):15 – 25, 1993.

[18] F. Mancini. *Graph modification problems related to graph classes*. PhD thesis, University of Bergen, Norway, 2008.

[19] R. Niedermeier. *Invitation to fixed parameter algorithms*, volume 31 of *Oxford Lectures Series in Mathematics and its Applications*. Oxford University Press, 2006.

[20] R. Niedermeier and P. Rossmanith. A general method to speed up fixed-parameter-tractable algorithms. *Inf. Process. Lett*, 73(3-4):125–129, 2000.

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

[22] R. Sharan. *Graph modification problems and their applications to genomic research*. PhD thesis, Tel-Aviv University, 2002.

[23] R. E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comput*, 13(3):566–579, 1984.

[24] S. Thomassé. A $4k^2$ kernel for feedback vertex set. *ACM Transactions on Algorithms*, 6(2), 2010.

[25] Y. Villanger. `www.lirmm.fr/~paul/ANR/CIRM-TALKS-2010/Villanger-cirm-2010.pdf`, 2010.

[26] Y. Villanger. Proper interval vertex deletion. In *IPEC*, volume 6478 of *LNCS*, pages 228–238, 2010.

[27] Y. Villanger, P. Heggernes, C. Paul, and J. A. Telle. Interval completion is fixed parameter tractable. *SIAM J. Comput*, 38(5):2007–2020, 2009.

[28] G. Wegner. *Eigenschaften der nerven homologische-einfactor familien in $R^n$*. PhD thesis, Universität Gottigen, Gottingen, Germany, 1967.

[29] M. Yannakakis. Computing the minimum fill-in is NP-Complete. *SIAM J. Alg. and Discr. Meth.*, 2(1):77–79, 1981.