Problem Kernels for NP-Complete Edge Deletion Problems: Split and Related Graphs

Jiong Guo*

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

Abstract. In an edge deletion problem one is asked to delete at most k edges from a given graph such that the resulting graph satisfies a certain property. In this work, we study four NP-complete edge deletion problems where the goal graph has to be a chain, a split, a threshold, or a co-trivially perfect graph, respectively. All these four graph classes are characterized by a common forbidden induced subgraph $2K_2$, that is, an independent pair of edges. We present the seemingly first non-trivial algorithmic results for these four problems, namely, four polynomial-time data reduction algorithms that achieve problem kernels containing $O(k^2)$, $O(k^4)$, $O(k^3)$, and $O(k^3)$ vertices, respectively.

1 Introduction

Given a graph G, a graph property Π (for instance, to belong to a certain graph class), and an integer $k \geq 0$, the Π edge deletion (for short, Π deletion) problem asks for a set of at most k edges whose deletion transforms G into a graph satisfying Π. The solution set is called Π *deletion set*. Edge deletion problems have applications in several areas, such as molecular biology and numerical algebra (see, for example, [2,14,18]), and their computational complexity has been widely studied in the literature. Yannakakis [20] gave the first systematic study of the complexity of edge deletion problems. We refer to [2,14,18] for excellent overviews.

In contrast to the extensive study on the complexity of Π deletion problems, relatively few algorithmic results are known for these problems. A general, constant-factor approximation algorithm was given by Natazon et al. [14] for Π deletion problems on bounded-degree graphs with respect to properties Π that can be characterized by finite sets of forbidden induced subgraphs. A forbidden induced subgraph characterization of a graph property Π means that a graph satisfies Π iff it contains none of a given set $\mathcal{H}$ of graphs as induced subgraphs. Herein, the graphs satisfying Π are also called $\mathcal{H}$-free graphs. Concerning parameterized complexity, Cai [3] showed that, for a graph property Π characterized

* Supported by the Deutsche Forschungsgemeinschaft (DFG), Emmy Noether research group PIAF (fixed-parameter algorithms), NI 369/4, and research project DARE (data reduction and problem kernels), GU 1023/1-1.

T. Tokuyama (Ed.): ISAAC 2007, LNCS 4835, pp. 915–926, 2007.

by a finite set $\mathcal{H}$ of forbidden induced subgraphs, the corresponding Π deletion problem is *fixed-parameter tractable*. More precisely, there exists a search tree based algorithm solving the problem in $O(d^k \cdot p(|G|))$ time with d being the maximum size of the edge sets of the forbidden subgraphs in $\mathcal{H}$ and p a polynomial function of the size of the input graph G.

Problem kernelization has been recognized as one of the most important contributions of fixed-parameter algorithmics to practical computing [5,11,15]. A *kernelization* is a polynomial-time algorithm that transforms a given instance I with parameter k of a problem P into a new instance I' with parameter $k' \leq k$ of P such that the original instance I is a yes-instance with parameter k iff the new instance I' is a yes-instance with parameter k' and $|I'| \leq g(k)$ for a function g. The instance I' is called the *problem kernel*. Recently, kernelizations of Π deletion problems attracted attention of more and more researchers; for example, there is a series of papers improving the problem kernel for Cluster Editing, where the goal graph is required to be a set of disjoint cliques, from quadratic size to linear size [9,17,6,10].

In this work, we will provide kernelization results for several Π deletion problems whose corresponding properties have $2K_2$ as one of their forbidden induced subgraphs. A $2K_2$-graph is an independent pair of edges, that is, a graph with four vertices and two non-adjacent edges. The class of $2K_2$-free graphs contains many important graph classes such as *chain graphs*, *split graphs*, *threshold graphs*, and *co-trivially perfect graphs* [1]. Here, we will study the corresponding Π deletion problems for Π being these four graph classes, denoted as Chain Deletion, Split Deletion, Threshold Deletion, and Co-Trivially Perfect Deletion. The main results are four polynomial-time kernelization algorithms which achieve problem kernels with $O(k^2)$, $O(k^4)$, $O(k^3)$, and $O(k^3)$ vertices for the four problems, respectively. This seem to be the first non-trivial algorithmic results for these problems. Based on the general result by Cai [3] and the *interleaving* technique from [16], our kernelization results imply faster fixed-parameter algorithms for these problems with running times of $O(2^k + mnk)$, $O(5^k + m^4n)$, $O(4^k + kn^4)$, and $O(4^k + kn^4)$, respectively, where n denotes the number of vertices and m denotes the number of edges of a given graph.

2 Preliminaries

Parameterized complexity is a two-dimensional framework for studying the computational complexity of problems [5,7,15]. One dimension is the input size n (as in classical complexity theory) and the other one the *parameter* k (usually a positive integer). A problem is called *fixed-parameter tractable* (fpt) if it can be solved in $f(k) \cdot n^{O(1)}$ time, where f is a computable function only depending on k. A core tool in the development of fixed-parameter algorithms is polynomial-time preprocessing by *data reduction rules*, often yielding a *kernelization*. Herein, the goal is, given any problem instance I with parameter k, to transform it in polynomial time into a new instance I' with parameter k' such that the size of I' is bounded from above by some function only depending on k, $k' \leq k$, and (I, k)

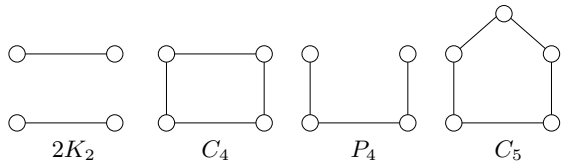

Fig. 1. The forbidden induced subgraphs $2K_2$, C_4, P_4, and C_5

is a yes-instance iff (I', k') is a yes-instance. A data reduction rule is *correct* if the new instance after an application of this rule is a yes-instance iff the original instance is a yes-instance. Throughout this paper, we call a problem instance *reduced* if the corresponding data reduction rules cannot be applied anymore.

We only consider *undirected* graphs $G = (V, E)$, where V is the set of vertices and E is the set of edges. The *complement graph* of G is denoted by $\bar{G} = (V, \bar{E})$ where an edge $\{u, v\} \in \bar{E}$ iff $\{u, v\} \notin E$. The *bipartite complement graph* $\bar{B}$ of a bipartite graph $B = (X, Y, E)$ contains an edge between two vertices $x \in X, y \in Y$ iff $\{x, y\} \notin E$. The *(open) neighborhood* $N_G(v)$ of a vertex $v \in V$ in graph G is the set of vertices that are adjacent to v in G. The *degree* of a vertex v, denoted by $\deg(v)$, is the size of $N_G(v)$. We use $N_G[v]$ to denote the *closed* neighborhood of v in G, that is, $N_G[v] := N_G(v) \cup \{v\}$. For a set of vertices $V' \subseteq V$, the *induced subgraph* $G[V']$ is the subgraph of G over the vertex set V' with the edge set $\{\{v, w\} \in E \mid v, w \in V'\}$. A subset I of vertices is called an *independent set* if $G[I]$ has no edge, whereas a subset K of vertices is called a *clique* if $G[K]$ has all possible edges. For an edge e and an edge set E', we use $G - e$ and $G - E'$ to denote the subgraph of G without e and the edges in E', respectively. For a vertex v and a vertex set V', the notions $G - v$ and $G - V'$ denote the subgraphs of G induced by $V \setminus \{v\}$ and $V \setminus V'$, respectively.

In the following, we will study several graph classes with forbidden induced subgraph characterization. We say that a vertex v *occurs* in a forbidden induced subgraph if v is contained in such an induced subgraph in G. See Fig. 1 for the forbidden induced subgraphs occurring in this work. We call the edge in a P_4 whose both endpoints have degree two the *middle edge*, and call the other two edges the *side edges*.

Compared to the Π vertex deletion problems where one deletes vertices instead of edges, the most difficult point in reducing a given instance of a Π edge deletion problem is how to deal with the vertices that do not occur in any forbidden subgraph. In the vertex version, we can simply remove these vertices, since such vertices can never be included in any optimal solution. However, since deleting an edge could create a new occurrence of a forbidden induced subgraph, it is possible that all optimal solutions of an edge deletion problem have to include an edge not involved in any forbidden subgraph in the original graph. Thus, in this case, we cannot simply remove the vertices that do not occur in any forbidden subgraph. In the following, as one of our main technical contributions, we show that, for the $2K_2$-free graph classes, chain, split, threshold, and co-trivially perfect, such vertices can be removed without affecting the solvability of the corresponding edge deletion problems. We begin with the most simple case, Chain Deletion.

3 Chain Deletion

A bipartite graph $B = (X, Y, E)$ with X and Y being two disjoint vertex subsets is called a *chain graph* if the neighborhoods of the vertices in X form a chain, that is, if there is an ordering of the vertices in X, say $x_1, x_2, \ldots, x_{|X|}$, such that $N_B(x_1) \subseteq N_B(x_2) \subseteq \ldots \subseteq N_B(x_{|X|})$. It is easy to see that the neighborhoods of the vertices in Y also form a chain. Yannakakis [19] introduced this graph class and proved that the Chain Completion problem, where one is asked whether there is a set of at most k edges whose addition transforms a given bipartite graph into a chain graph, is NP-complete. Since the bipartite complement graph of a chain graph is a chain graph as well, Chain Deletion is NP-complete as well.

The main result of this section consists of two data reduction rules for Chain Deletion that lead to a quadratic-size problem kernel. To this end, we need the following forbidden subgraph characterization of chain graphs given by Yannakakis [19]: A bipartite graph is a chain graph if and only if it does not contain a $2K_2$ as an induced subgraph. Without loss of generality, we assume that the input graph is connected: For a disconnected graph, only some edges of exactly one connected component can be kept. This means that we have to consider each single component individually. We apply the following two data reduction rules to a given bipartite graph $B = (X, Y, E)$:

Rule 1: If there is an edge e involved in more than k $2K_2$'s, then delete e from B, add e to the chain deletion set, and decrease the parameter k by one. If $k < 0$, then report "No".

Rule 2: Delete the vertices from B that do not occur in any $2K_2$.

The correctness of the first rule is easy to verify.

Lemma 1. *Rule 2 is correct.*

Proof. To show the lemma, it suffices to show that, for a vertex v not in any $2K_2$, graph $B = (X, Y, E)$ has a chain deletion set E' with $|E'| \leq k$ if and only if graph $B' := B - v$ has a chain deletion set E'' with $|E''| \leq k$.

"$\Rightarrow$": Since every induced subgraph of a chain graph is a chain graph, this direction is correct.

"$\Leftarrow$": Suppose that E'' is a chain deletion set for B'. Let B'' denote the chain graph resulting by removing the edges in E'' from B'. Then, add v to B'' and connect v to the vertices in $N_B(v)$. By contradiction we show that the resulting graph H is a chain graph and, thus, E'' is a chain deletion set for B. Suppose that H is not a chain graph; this means that v occurs in a $2K_2$ in H.

We associate with the edges in E'' an ordering as follows: Based on the forbidden subgraph $2K_2$, one can easily enumerate all size-at-most-k chain deletion sets for B' by a search tree of height k: At each node α of the search tree, find an induced $2K_2$ and branch into two cases, deleting one edge or the other. Then, recursively treat the two cases. Each of the two search tree edges between node α and its two children is labeled by the graph edge deleted in the corresponding

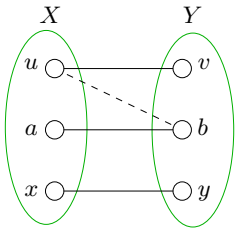

Fig. 2. Illustration of the proof of Lemma 1. The dashed line represents a deleted edge.

case. All solutions with size at most k are stored at the leaves of the search tree. Solution E'' corresponds to a path from the root to a leaf in the search tree. Assume that the edges in E'' are numbered according to their occurrences on this path, $e_1, e_2, \ldots, e_l$ with $l \leq k$, from the root to the leaf. Moreover, let e_i, e_i' with $1 \leq i \leq l$ be the edge pair of the induced $2K_2$ for which e_i has been deleted from B' and added to E''. Let $B_0 := B'$ and $B_i := B_{i-1} - e_i$ for $1 \leq i \leq l$. Obviously, $B_l = B''$. Moreover, let B_i' for $0 \leq i \leq l$ be the graph resulting by adding v to B_i and connecting v to $N_B(v)$. Clearly, $B_0' = B$ and $B_l' = H$.

Since v is not in any $2K_2$ in B but in one in H there is a $2K_2$ containing v, we can assume that B_j' is the graph with the minimum index among $B_1', \ldots, B_l'$ where v occurs in a $2K_2$ induced by edges $\{u, v\}$ and $\{a, b\}$ with $a, u \in X$ and $b, v \in Y$. Since v is not in any $2K_2$ in B_{j-1}' and E'' contains no edges incident to v, we have $\{a, v\} \notin E$ and $e_j = \{b, u\}$. According to the branching strategy stated above, $\{b, u\}$ has to form a $2K_2$ with an edge $\{x, y\}$ in B_{j-1}' with $x, y \notin \{a, v\}$. Assume that $x \in X$ and $y \in Y$. See Fig. 2 for an illustration of the subgraph of B_i' containing vertices u, v, x, y, a, b. Since v does not occur in any $2K_2$ in B, at least one of the edges $\{u, y\}$ and $\{v, x\}$ exists in B. If $\{v, x\} \in E$, then $\{b, x\} \in E$, since, otherwise, $\{v, x\}$ and $\{a, b\}$ would form a $2K_2$ containing v in B. Since v is not in any $2K_2$ in B_{j-1}', the edge $\{b, x\}$ would exist in B_{j-1}'. This is a contradiction to the fact that $\{b, u\}$ and $\{x, y\}$ form a $2K_2$ in B_{j-1}. Thus, $\{v, x\} \notin E$. This implies that $\{u, y\} \in E$ and $\{u, y\}$ exists in B_{j-1}'. Again, we have a contradiction to the fact that $\{b, u\}$ and $\{x, y\}$ form a $2K_2$ in B_{j-1}. Therefore, H is a chain graph and E'' is a chain deletion set of B. □

Based on these two rules, we prove the size bound of the reduced graphs.

Theorem 1. *If a reduced instance (B, k) of* Chain Deletion *is a yes-instance, then B has at most $2k^2$ vertices. The kernelization runs in $O(mnk)$ time.*

Proof. Let $B = (X, Y, E)$ be a Chain Deletion instance that is reduced with respect to Rules 1 and 2 and has a chain deletion set E' with $|E'| \leq k$. We analyze in the following the size of X; the size bound of Y follows analogously. Since B is reduced with respect to Rule 2, every vertex from X is involved in at least one $2K_2$ in B. Moreover, due to Rule 1, there can be at most k^2 many $2K_2$'s in B with $|E'| \leq k$. Therefore, $|X| \leq k^2$.

We prove the running time by showing that Rules 1 and 2 can be exhaustively executed in $O(mnk)$ time: We first apply Rule 1 exhaustively and then apply

once Rule 2. For one application of Rule 1, iterate over all edges and, for each edge e, remove all neighbors of its endpoints. If there remain more than k edges, then e occurs in more than k $2K_2$'s and Rule 1 is applicable. Obviously, Rule 1 can be applied at most k times and needs $O(mnk)$ time.

To apply Rule 2, we compute all $2K_2$'s in the graph after exhaustive application of Rule 1. If a vertex v does not occur in $2K_2$, then apply Rule 2 to v. Thus, Rule 2 needs $O(mn)$ time. □

4 Split Deletion

A graph $G = (V, E)$ is called a *split graph* if there exists a partition (K, I) of V such that K is a clique and I is an independent set. Földes and Hammer [8] introduced this graph class in 1977 and gave the following forbidden subgraph characterization:

Lemma 2 ([8]). *A graph is a split graph if and only if it contains no induced $2K_2$, C_4, and C_5.*

Split Deletion is NP-complete [14]. We prove that Split Deletion admits a problem kernel with $O(k^4)$ vertices. Because split graphs are closed under the complement operation, the result holds for Split Completion as well.

We follow almost the same approach as in Sect. 3, namely, first get rid of the vertices that do not occur in any forbidden induced subgraph and then delete the edges which have to be in any size-$\leq k$ split deletion set. However, since split graphs have, besides $2K_2$, also C_4 and C_5 as forbidden subgraphs, we need additional data reduction rules and their correctness proofs are more complicated than in the case of Chain Deletion. In particular, how to deal with two edges that occur together in more than k forbidden subgraphs that, with the exception of these two edges, are pairwisely edge-disjoint, is a new task here. In general, given such two edges, we only know that at least one of them has to be deleted, but we cannot decide in polynomial time which one of them has to be deleted. Here, for $2K_2$-free graphs, we solve this problem by showing that such two edges can be replaced by a $2K_2$-similar gadget (see Rules 2 and 3).

We apply seven data reduction rules to a Split Deletion instance (G, k) and prove their correctness and running times, respectively. During the reduction process, whenever $k < 0$, we know that the given instance has no solution and report "No".

Rule 1. Delete the vertices from G that are not in any $2K_2$, C_4, and C_5.

Lemma 3. *Rule 1 is correct and one application of Rule 1 needs $O(mn^3)$ time.*

Proof. We prove this lemma by showing that the input graph $G = (V, E)$ has a size-$\leq k$ split deletion set iff the graph $G' = (V', E')$ after one application of Rule 1 has a size-$\leq k$ split deletion set.

"⇒": This direction is clearly correct since every induced subgraph of a split graph is a split graph.

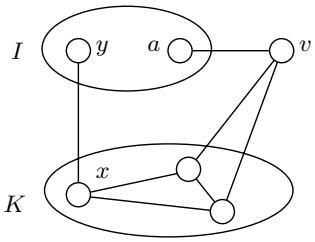

Fig. 3. Illustration of the proof of Lemma 3

"⇐": Suppose that G' has a split deletion set S with $|S| \leq k$ and let $G'' = (V'', E'')$ denote the graph after deleting the edges in S from G', where K and I are the clique and the independent set of G''. If the graph resulting by adding v to G'' and connecting v to $N_G(v)$ remains a split graph, then we are done; otherwise, there exist some vertex $x \in K$ with $\{x, v\} \notin E$ and some vertex $a \in I$ with $\{a, v\} \in E$. See Fig. 3 for an illustration. W.l.o.g., we can assume that, in G'', the set $K \cap N_G(v)$ is maximal in the sense that, for each vertex u from $I \cap N_G(v)$, there is a vertex in K that is not adjacent to u. Moreover, we can also assume that every vertex in K has a neighbor in $I \cup \{v\}$ and let y be a neighbor of x in I. Next, we prove the following claim.

Claim 1. The set $N_G(v)$ is a clique in G.

Proof of Claim 1: We prove the claim by contradiction. Suppose that $a_1, a_2 \in N_G(v)$ with $\{a_1, a_2\} \notin E$. Observe that $|K \setminus N_G(v)| \leq 1$. To see this, suppose that there exists, besides x, another vertex x' in $K \setminus N_G(v)$. Since v does not occur in any $2K_2$ in G, at least two of the edges $\{a_1, x\}$, $\{a_1, x'\}$, $\{a_2, x\}$, and $\{a_2, x'\}$ have to exist in G. These edges and the edges $\{x, x'\}$, $\{a_1, v\}$, and $\{a_2, v\}$ induce at least one C_4 or C_5 containing v, a contradiction to the fact that v does not occur in any C_4 and C_5 in G. Thus, $|K \setminus N_G(v)| \leq 1$.

Next, we show that $y \notin N_G(v)$. Suppose that $y \in N_G(v)$. Then there exists a vertex $z \in K$ with $\{y, z\} \notin E$; otherwise, $K \cap N_G(v)$ would not be maximal. However, due to the fact $|K \setminus N_G(v)| \leq 1$, we have $z \in N_G(v)$ and, thus, a C_4 with v, z, x, y in G, a contradiction to the fact that v does not occur in any C_4. Since $x, y \notin N_G(v)$ and v is not contained in any $2K_2$ in G, at least two of the edges $\{x, a_1\}$, $\{x, a_2\}$, $\{y, a_1\}$, and $\{y, a_2\}$ have to exist in E and, thus, we have a C_4 or C_5 containing v in G, a contradiction to the fact that v does not occur in any C_4 and C_5. This concludes the proof of Claim 1.

Now we show that, in the case that $N_G(v)$ is a clique, we can construct a split deletion set S' from S for G with $|S'| \leq |S| \leq k$. Consider the split graph $H = (V_H, E_H)$ where the clique K' of H consists of the vertices in $N_G(v) \cup X$ with $X := \{v \in K \mid v \in (\bigcap_{u \in N_G(v)} N_G(u))\}$ and the independent set I' of H consists of the vertices in $\{v\} \cup (I \setminus N_G(v)) \cup (K \setminus X)$. The set E_H contains all possible edges between the vertices in K' and all edges in E between K' and I'. Since $N_G(v)$ is a clique, there exists a split deletion set S' transforming G into H.

To show $|S'| \leq |S|$, it suffices to show $|E^1| + |E^2| \geq |E^3| + |E^4|$, where E^1 is the set of the edges in E between the vertices in $N'(v) := N_G(v) \cap I$, E^2 the set of the edges in E between $N'(v)$ and $I \setminus N_G(v)$, E^3 the set of the edges in E between $K \setminus (X \cup N_G(v))$ and $I \setminus N'(v)$, and E^4 the set of the edges in E between the vertices in $K \setminus (X \cup N_G(v))$.

First we claim that $|N'(v)| \geq |K \setminus (X \cup N_G(v))|$. Since both sets are cliques in G, this claim implies $|E^1| \geq |E^4|$. Suppose that the claim is not true. Then, by the pigeonhole principle, there exist at least two distinct vertices $x, y \in (K \setminus (X \cup N_G(v)))$ with $\{x, a\} \notin E$ and $\{y, a\} \notin E$ for a vertex $a \in N'(v)$. This implies that we have a $2K_2$ with edges $\{x, y\}$ and $\{a, v\}$ in G, a contradiction to the fact that v does not occur in a $2K_2$ in G.

Next we show that $|E^2| \geq |E^3|$. Consider a vertex $x \in (K \setminus (X \cup N_G(v)))$ with a neighbor y in I. From the definition of X, we have $N'(v) \setminus N_G(x) \neq \emptyset$ and let $a \in (N'(v) \setminus N_G(x))$. We have $\{a, y\} \in E$, since, otherwise, $\{x, y\}$ and $\{a, v\}$ would induce a $2K_2$ in G. Thus, for each edge from E^3 that is incident to x, there exists at least one edge from E^2 incident to a' for every vertex $a' \in (N'(v) \setminus N_G(x))$. Moreover, for each two distinct vertices $w, w' \in (K \setminus (X \cup N_G(v)))$, the sets $N'(v) \setminus N_G(w)$ and $N'(v) \setminus N_G(w')$ are disjoint, since, otherwise, there would be a $2K_2$ involving v, w, w' in G. Therefore, we can conclude that $|E^2| \geq |E^3|$ and $|S'| \leq |S|$.

The running time of Rule 1 follows from the observation that all $2K_2$'s, C_4's, and C_5's of G can be enumerated in $O(mn^3)$ time. □

> **Rule 2.** If two edges $\{u, v\}$ and $\{u, w\}$ occur together in more than k C_4's, then delete $\{u, v\}$ and $\{u, w\}$ from G and add two edges $\{a, v\}$ and $\{b, w\}$ to G with a, b being two new degree-one vertices.

Lemma 4. *Rule 2 is correct and one application of Rule 2 needs $O(m^2 n)$ time.*

Proof. Let $\{u, v\}$ and $\{u, w\}$ be the two edges to which Rule 2 has been applied. We use $X := \{x_1, x_2, \ldots, x_{k+1}\}$ to denote the set of the fourth vertices of $k + 1$ arbitrarily chosen C_4's containing both $\{u, v\}$ and $\{u, w\}$. Let $G' = (V', E')$ denote the resulting graph after applying Rule 2 to $\{u, v\}$ and $\{u, w\}$. We prove the correctness of Rule 2 by showing that we can transform a split deletion set S of the original graph G into a split deletion set S' of G' with $|S'| = |S| \leq k$ and vice versa. We show here only the direction from S to S'; the reverse direction can be proven in a similar way.

We use K and I to denote the clique and the independent set of the resulting split graph after deleting S from G. Clearly, at least one of v and w has to be in I. Then, at least one of $x_1, \ldots, x_{k+1}$ has to be in K and $u \in I$. If $v \in I$ and $w \in K$, then $S' := S \setminus \{\{u, v\}\} \cup \{\{a, v\}\}$ is a split deletion set for G'. To see this, observe that, due to the existence of $x_1, \ldots, x_{k+1}$, both a and b have to be in the independent set of any split graph resulting by deleting at most k edges from G'. This argument works also for the case $v, w \in I$ with the only difference lying in the construction of S', namely, $S' := S \setminus \{\{u, v\}, \{u, w\}\} \cup \{\{a, v\}, \{b, w\}\}$.

To apply Rule 2, we iterate over all two adjacent edges $\{u, v\}$ and $\{u, w\}$ with $\{v, w\} \notin E$. For each such edge pair, delete the vertices in $N_G[u] \setminus \{v, w\}$

and count the common neighbors of v and w in the resulting graph. If the number of common neighbors exceeds k, then Rule 2 is applicable to these two edges. □

Rule 3. If two adjacent edges e and e' occur together in more than k C_5's that, with the exception of e and e', are pairwisely edge-disjoint, then delete e and e', add e and e' to the split deletion set, and decrease the parameter k by two.

Lemma 5. *Rule 3 is correct and one application Of Rule 3 needs $O(m^3\sqrt{n})$ time.*

Proof. Suppose that edges $e = \{u, v\}$ and $e' = \{u, w\}$ are two edges that satisfy the precondition of Rule 3. Let $\{x_1, y_1\}, \{x_2, y_2\}, \ldots, \{x_{k+1}, y_{k+1}\}$ be the edges which are from $k + 1$ arbitrary C_5's containing e and e' and whose endpoints are not from $\{v, w\}$. Clearly, at least one of $x_1, \ldots, x_{k+1}$ and at least one of $y_1, \ldots, y_{k+1}$ have to be in the clique of any split graph resulting by deleting at most k edges from G. Then, both v, w have to be in the independent set that contains u as well. Hence, we have to delete e and e'.

To apply this rule, we iterate over all two adjacent edges whose endpoints do not induce a triangle. For each pair $e = \{u, v\}$ and $e' = \{u, w\}$, we construct a subgraph of G having only the vertices in $N(v) \setminus (N[u] \cup N(w))$ and $N(w) \setminus (N[u] \cup N(v))$ and the edges between these two sets. If this bipartite subgraph has a matching of size more than k, then this rule is applicable to e and e'. The running time follows from the running time $O(m\sqrt{n})$ for computing maximum bipartite matchings [4]. □

Rule 4. If an edge e occurs in more than k $2K_2$'s, then delete e from G, add e to the split deletion set, and decrease the parameter k by one.

Rule 5. If an edge e occurs in more than k C_4's that, with the only exception of e, are pairwisely edge-disjoint, then delete e from G, add e to the split deletion set, and decrease the parameter k by one.

Rule 6. If an edge e occurs in more than k C_5's that, with the only exception of e, are pairwisely edge-disjoint, then delete e from G, add e to the split deletion set, and decrease the parameter k by one.

Lemma 6. *Rules 4, 5, and 6 are correct and one application of these rules needs $O(mn)$, $O(m^2\sqrt{n})$, and $O(mn^3)$ time, respectively.*

Proof. Clearly, these rules are correct. We give here only a proof of the running time $O(m^2\sqrt{n})$ of Rule 5. To apply Rule 5, we iterate over all graph edges and, for each edge $e = \{u, v\}$, keep only the vertices in $N(u)\backslash N[v]$ and $N(v)\backslash N[u]$ and the edges between these two sets. Finally, we compute a maximum matching of the remaining bipartite graph. If the matching has a size more than k, then Rule 5 can be applied to e. The running time follows from the running time $O(m\sqrt{n})$ for computing maximum bipartite matchings [4]. □

Rule 7. If three edges $\{u, v\}$, $\{v, w\}$, and $\{w, x\}$ occur together in more than k C_5's, then delete $\{v, w\}$ from G, add $\{v, w\}$ to the split deletion set, and decrease the parameter k by one.

The following lemma can be shown with the same arguments as used in the proofs of Lemmas 4 and 6.

Lemma 7. *Rule 7 is correct and one application of Rule 7 needs $O(m^3n)$ time.*

Now, we arrive at the central result of this section.

Theorem 2. *If a reduced instance $(G = (V, E), k)$ of* Split Deletion *is a yes-instance, then $|V| = O(k^4)$. The kernelization runs in $O(m^4\sqrt{n})$ time.*

Proof. Suppose that a reduced graph G has a split deletion set S with $|S| \leq k$. Due to Rule 1, every vertex v in G has to be in a $2K_2$, C_4, or C_5. In the following we give an upper bound on the number of the vertices which are contained in C_5's. Clearly, every C_5 in G has to contain at least one edge from S. Due to Rule 6 every edge e of S can be in at most k C_5's that have only e in common. Let A denote the set of these C_5's. There are at most $4k + 1$ edges in the C_5's from A. Since G is reduced with respect to Rule 3, each edge e' from the C_5's in A with $e' \neq e$ can form together with e at most k C_5's that, with the exception of e and e', are pairwisely edge-disjoint. Add these C_5's with both e and e' to A. Now A contains at most $4k^2$ many C_5's. These C_5's contain at most $12k^2$ many induced length-two paths that contain e. Due to Rule 7, each of these P_3's can be contained in at most k C_5's. Therefore, edge e can be contained in at most $12k^3$ many C_5's. For $|S| \leq k$, we have at most $12k^4$ many C_5's in G which gives an upper bound of $36k^4 + 2k$ on the number of the vertices contained in C_5's: $|V \setminus V(S)| \leq 36k^4$ and $|V(S)| \leq 2k$ with $V(S)$ being the vertex set of S. With similar arguments, the number of the vertices in C_4's and $2K_2$'s can also be upper-bounded by $O(k^3)$ and $O(k^2)$, respectively. This gives the size bound claimed in the theorem. The running time follows from Lemmas 3 to 7 and the fact that the seven rules can altogether be executed at most m times. □

5 Threshold Deletion and Co-trivially Perfect Deletion

A graph G is a *threshold graph* iff G contains no induced $2K_2$, C_4, and P_4, while a *co-trivially perfect graph* contains no induced $2K_2$ and P_4 [1]. Threshold graphs have been extensively studied in literature [1]. See the monograph of Mahadev and Peled [12] for more results on threshold graphs. Margot [13] showed that Threshold Deletion is NP-complete. Yannakakis [19] showed that Co-Trivially Perfect Deletion is NP-complete. Clearly, threshold graphs form a subclass of both split graphs and co-trivially perfect graphs. We prove that both problems admit polynomial-time kernelizations with $O(k^3)$-vertex problem kernels. These results hold for Threshold Completion and Trivially Perfect Completion as well, since threshold graphs are closed under complementation and the complement graph of a co-trivially perfect graph is a trivially perfect graph.

The basic idea behind the kernelizations is almost the same as for Split Deletion. Therefore we present only the data reduction rules for Threshold Deletion. The explicit description of the data reduction rules for Co-Trivially Perfect Deletion is deferred to the full version of this paper.

Rule 1. Delete the vertices from G that are not in any $2K_2$, C_4, and P_4.
Rule 2. If an edge e occurs in more than k $2K_2$'s, then delete e from G, add it the threshold deletion set, and decrease the parameter k by one.
Rule 3. If an edge e occurs in more than k C_4's that, with the only exception of e, are pairwisely edge-disjoint, then the given instance has no solution and report "No".
Rule 4. If two edges e and e' occur together in more than k C_4's, then delete e and e' from G, add both to the threshold deletion set, and decrease the parameter k by two.
Rule 5. If an edge e is the middle edge of more than k P_4's that, with the only exception of e, are pairwisely edge-disjoint, then the given instance has no solution and report "No".
Rule 6. If an edge e occurs in more than k P_4's as a side edge, then delete e, add e to the threshold deletion set, and decrease the parameter k by one.
Rule 7. If two edges e and e' occur together in more than k P_4's where e' is the middle edge, then remove e from G, add e to the threshold deletion set, and decrease the parameter k by one.

Theorem 3. *If a reduced instance* $(G = (V, E), k)$ *of* Threshold Deletion *is a yes-instance, then* $|V| = O(k^3)$. *The kernelization runs in* $O(kn^4)$ *time.*

Based on five data reduction rules that are very similar to Rules 1, 2, 5, 6, and 7 described above, we can also achieve a kernelization for Co-Trivially Perfect Deletion.

Theorem 4. *If a reduced instance* $(G = (V, E), k)$ *of* Co-Trivially Perfect Deletion *is a yes-instance, then* $|V| = O(k^3)$. *The kernelization runs in* $O(kn^4)$ *time.*

6 Outlook

In this work, we studied several edge deletion problems for generating $2K_2$-free graphs and obtained polynomial-time kernelization algorithms for them. An interesting open problem is whether the approach adopted here, namely, first to get rid of the vertices that are not in any forbidden subgraph and then to delete the edges which are contained in too many forbidden subgraphs, also applies to edge deletion problems for generating $2K_2$-free graphs, even when the set of forbidden subgraphs is infinite. Moreover, another challenging task would be to improve the kernel sizes achieved here to linear size (if possible) and the running times of the kernelizations, as done for Cluster Editing [6,10,17].

References

1. Brandstädt, A., Le, V.B., Spinrad, J.P.: Graph Classes: a Survey. In: SIAM Monographs on Discrete Mathematics and Applications (1999)
2. Burzyn, P., Bonomo, F., Durán, G.: NP-completeness results for edge modification problems. Discrete Applied Mathematics 154, 1824–1844 (2006)

3. Cai, L.: Fixed-parameter tractability of graph modification problems for hereditary properties. Information Processcing Letters 58, 171–176 (1996)
4. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT Press and McGraw-Hill (2001)
5. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer, Heidelberg (1999)
6. Fellows, M.R., Langston, M.A., Rosamond, F., Shaw, P.: Polynomial-time linear kernelization for Cluster Editing. In: Proc. 16th FCT. LNCS, Springer, Heidelberg (2007)
7. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer, Heidelberg (2006)
8. Földes, S., Hammer, P.L.: Split graphs. Congressus Numerantium 19, 311–315 (1977)
9. Gramm, J., Guo, J., Hüffner, F., Niedermeier, R.: Graph-modeled data clustering: Exact algorithms for clique generation. Theory of Computing Systems 38(4), 373–392 (2005)
10. Guo, J.: A more effective linear kernelization for Cluster Editing. In: Proc. 1st ESCAPE. LNCS, vol. 4614, pp. 36–47. Springer, Heidelberg (2007)
11. Guo, J., Niedermeier, R.: Invitation to data reduction and problem kernelization. ACM SIGACT News 38(1), 31–45 (2007)
12. Mahadev, N.V.R., Peled, U.N.: Threshold Graphs und Related Topics. Annals of Discrete Mathematics 56 (1995)
13. Margot, F.: Some complexity results about threshold graphs. Discrete Applied Mathematics 49(1-3), 299–308 (1994)
14. Natanzon, A., Shamir, R., Sharan, R.: Complexity classification of some edge modification problems. Discrete Applied Mathematics 113, 109–128 (2001)
15. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford University Press, Oxford (2006)
16. Niedermeier, R., Rossmanith, P.: A general method to speed up fixed-parameter-tractable algorithms. Inf. Process. Lett. 73, 125–129 (2000)
17. Protti, F., da Silva, M.D., Szwarcfiter, J.L.: Applying modular decomposition to parameterized bicluster editing. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 1–12. Springer, Heidelberg (2006)
18. Sharan, R.: Graph Modification Problems and Their Applications to Genomic Research. PhD thesis, School of Computer Science, Tel-Aviv University (2002)
19. Yannakakis, M.: Computing the minimum fill-in is NP-complet. SIAM Journal on Algebraic and Discrete Methods 2(1), 297–309 (1981)
20. Yannakakis, M.: Edge-deletion problems. SIAM Journal on Computing 10(2), 297–309 (1981)