Algorithmica (2015) 71:989–1006
DOI 10.1007/s00453-013-9837-5

Faster Parameterized Algorithms for Deletion to Split Graphs

Esha Ghosh · Sudeshna Kolay · Mrinal Kumar · Pranabendu Misra · Fahad Panolan · Ashutosh Rai · M.S. Ramanujan

Received: 31 October 2012 / Accepted: 11 September 2013 / Published online: 5 October 2013

Abstract An undirected graph is said to be *split* if its vertex set can be partitioned into two sets such that the subgraph induced on one of them is a complete graph and the subgraph induced on the other is an independent set. We initiate a systematic study of parameterized complexity of the problem of deleting the minimum number of vertices or edges from a given input graph so that the resulting graph is split. We give efficient fixed-parameter algorithms and polynomial sized kernels for the problem. More precisely,

1. for SPLIT VERTEX DELETION, the problem of determining whether there are k vertices whose deletion results in a split graph, we give an $\mathcal{O}^*(2^k)$ algorithm ($\mathcal{O}^*()$ notation hides factors that are polynomial in the input size) improving on the pre-

E. Ghosh · S. Kolay · F. Panolan · A. Rai (✉) · M.S. Ramanujan
The Institute of Mathematical Sciences, Chennai, India
e-mail: ashutosh@imsc.res.in

E. Ghosh
e-mail: esha@imsc.res.in

S. Kolay
e-mail: skolay@imsc.res.in

F. Panolan
e-mail: fahad@imsc.res.in

M.S. Ramanujan
e-mail: msramanujan@imsc.res.in

M. Kumar
Indian Institute of Technology, Madras, Chennai, India
e-mail: mrinalk@cse.iitm.ac.in

P. Misra
Chennai Mathematical Institute, Chennai, India
e-mail: pranabendu@cmi.ac.in

vious best bound of $\mathcal{O}^*(2.32^k)$. We also give an $\mathcal{O}(k^3)$-sized kernel for the problem.
2. For SPLIT EDGE DELETION, the problem of determining whether there are k edges whose deletion results in a split graph, we give an $\mathcal{O}^*(2^{\mathcal{O}(\sqrt{k}\log k)})$ algorithm. We also prove the existence of an $\mathcal{O}(k^2)$ kernel.

In addition, we note that our algorithm for SPLIT EDGE DELETION adds to the small number of subexponential parameterized algorithms not obtained through bidimensionality (Demaine et al. in J. ACM 52(6): 866–893, 2005), and on general graphs.

Keywords Parameterized complexity · Deletion problems · Split graphs · Subexponential algorithm

1 Introduction

A graph property Π is a collection of graphs. A graph property Π is *non-trivial* if it has infinitely many graphs in it, and also avoids infinitely many graphs. A non-trivial graph property is said to be *hereditary* if a graph G is in property Π implies that every induced subgraph of G is also in Π. The problem of editing (adding/deleting vertices/edges) to ensure that a graph has some property is a well studied problem in theory and applications of graph algorithms. When we want the resulting graph to be in a non-trivial hereditary graph class Π, the optimization versions of the corresponding vertex deletion problems are known to be NP-hard by a classical result of Lewis and Yannakakis [18]. Many edge deletion problems (including deletion to split graphs) are known to be NP-hard by results of Natanzon et al. [23]. This problem has also been studied in generality under paradigms like approximation [12, 20] and parameterized complexity [3, 14]. When Π is a specific hereditary class like chordal or planar graphs, extensive work has been done to explore tight bounds [11, 17, 21, 22]. In this paper, we initiate a study of these problems from the point of view of parameterized complexity when Π is the class of all split graphs, which is also a non-trivial hereditary graph class.

An undirected graph $G = (V, E)$ is said to be *split* if its vertex set V can be partitioned into two sets such that the induced subgraph on one of them is a complete graph and the induced subgraph on the other is an independent set. Split graphs were first studied by Földes and Hammer [9], and independently introduced by Tyshkevich and Chernyak [24]. In [9], the authors provided the following finite forbidden subgraph characterization of split graphs which gives an easy polynomial time algorithm for recognizing split graphs.

Lemma 1 ([9]) *A graph is a split graph if and only if it contains no induced subgraph isomorphic to $2K_2$, C_4, or C_5. Here, K_2 is the complete graph on two vertices, C_i is a cycle on i vertices.*

In this paper, we study the following two problems.

SPLIT VERTEX DELETION (SVD)	
Input:	Graph $G = (V, E)$, integer k
Parameter:	k
Question:	Does there exist a set of vertices of size at most k whose deletion from G results in a split graph?

SPLIT EDGE DELETION (SED)	
Input:	Graph $G = (V, E)$, integer k
Parameter:	k
Question:	Does there exist a set of edges of size at most k whose deletion from G results in a split graph?

As the size of the forbidden set is finite, these problems become fixed-parameter tractable (see Sect. 2) due to a general result of Cai [3], when parameterized by k. One can also observe from Lemma 1, a fairly straightforward branching algorithm for both SVD and SED with running time $\mathcal{O}^*(5^k)$.

Recently, in [19], the authors obtained an $\mathcal{O}^*(2.32)^k$ algorithm for SVD by reducing the problem to the ABOVE GUARANTEE VERTEX COVER problem and using the fixed-parameter algorithm for it. In this paper, we obtain an $\mathcal{O}^*(2^k)$ algorithm by the combination of a bound on the number of split partitions of a split graph, and the well known technique of iterative compression. We also obtain an $\mathcal{O}(k^3)$ vertex kernel for the problem. Note that, this kernel is smaller than the kernel with $\mathcal{O}(k^4)$ vertices, which can be obtained by an approach similar to d-HITTING SET [1]. We also prove that under certain complexity theoretic assumptions, we cannot obtain a subexponential algorithm for this problem.

For SED, we design a subexponential algorithm running in time $\mathcal{O}^*(2^{\mathcal{O}(\sqrt{k}\log k)})$ by combining the color and conquer approach [2], with the bound on the number of partitions of a split graph. This is one of the very few problems (see [10, 11] for other problems) for which we know a subexponential parameterized algorithm on general graphs which does not use bidimensionality theory. We also revisit the kernelization algorithm for this problem given by Guo [14], and by using only a subset of the rules presented there, we prove a bound of $\mathcal{O}(k^2)$ vertices improving on Guo's bound of $\mathcal{O}(k^4)$. Furthermore, the SPLIT COMPLETION problem of adding at most k edges to a given graph to make it split, is equivalent to deleting at most k edges from the complement of the graph to make it split. Hence, the bound on the kernel and the subexponential algorithm which we prove for SED also holds for SPLIT COMPLETION.

Related Work We also note that though computing a minimum split completion set is NP-complete, there is a linear time algorithm to compute a *minimal* split completion set [16]. Another interesting fact is that even though SED is NP-complete the SPLIT EDGE MODIFICATION problem (where we are allowed to add or delete k edges to get a split graph) is polynomial time solvable [15].

After the conference version of this paper was published, our algorithm for SVD was improved by Cygan and Pilipczuk [5] using a different method. Their algorithm runs in time $\mathcal{O}^*(1.2738^k k^{\mathcal{O}(\log k)})$.

Organization of the Paper In Sect. 3, we first present and analyze our algorithm for SPLIT VERTEX DELETION. Following that, we design a set of reduction rules for the problem which allow us to reduce the number of vertices in the graph to $\mathcal{O}(k^3)$. In Sect. 4, we present the subexponential algorithm for SPLIT EDGE DELETION, which is followed by an improved bound on a kernel for the same problem.

2 Preliminaries

For a graph $G = (V, E)$, and a set $A \subseteq E$ of edges, we denote by $V(A)$ the set of endpoints of the edges in A. For a set $S \subseteq V$, the *subgraph of G induced by S* is denoted by $G[S]$ and it is defined as the subgraph of G with vertex set S and edge set $\{(u, v) \in E : u, v \in S\}$ and the subgraph obtained after deleting S is denoted as $G \setminus S$. Similarly, the *subgraph of G induced by an edge set $A \subseteq E$* is defined as the subgraph of G with edge set A and vertex set $V(A)$ and is denoted by $G[A]$. All vertices adjacent to a vertex v are called neighbors of v and the set of all such vertices is called the neighborhood of v. Similarly, a non-adjacent vertex of v is called a non-neighbor and the set of all non-neighbors of v is called the non-neighborhood of v. The neighborhood of v is denoted by $N(v)$. We say that v is *global* to a set Z if v is adjacent to all vertices of Z and we say that v is non-adjacent (or non-neighbor) to a set Z if v is not adjacent to any vertex of Z. For two sets X and Y, we say that X is *global* to Y if every vertex in X is global to Y and that X is non-adjacent to Y if every vertex in X is non-adjacent to Y.

Given a function $col : V \to C$ from the vertices of the graph G to a set of colors, C, we say that an edge $(u, v) \in E$ is *monochromatic* if $col(u) = col(v)$ and *non-monochromatic* otherwise.

A graph G is called a *split graph* if the vertex set V can be partitioned into two sets V_1 and V_2 such that $G[V_1]$ is a complete graph and $G[V_2]$ is an independent set. We call a set $S \subseteq V$ a *split vertex deletion* (svd) set if the graph $G[V \setminus S]$ is a split graph and a set $A \subseteq E$ is called a *split edge deletion* (sed) set if the graph $G[E \setminus A]$ is a split graph.

Definition 1 Given a split graph $G = (V, E)$, a partition $(C \uplus I)$ of the vertex set into sets C and I is called a *split partition* of this split graph if $G[C]$ is a clique and $G[I]$ is an independent set.

Given a split partition $(C_0 \uplus I_0)$ of a subgraph G' of a split graph G, we say that a split partition $(C \uplus I)$ of G is *consistent* with the partition $(C_0 \uplus I_0)$ if $C_0 \subseteq C$ and $I_0 \subseteq I$.

We refer to an induced subgraph isomorphic to $2K_2$, or C_4 or C_5 as a *forbidden structure*.

Parameterized Complexity For decision problems with input size n, and a parameter k, the goal in parameterized complexity is to design an algorithm with running time $f(k)n^{\mathcal{O}(1)}$ where f is a function of k alone, as contrasted with an $n^{k+\mathcal{O}(1)}$ algorithm. Problems which admit such algorithms are said to be fixed parameter tractable (FPT). We also call an algorithm with a running time of $f(k)n^{\mathcal{O}(1)}$, an FPT algorithm, and such a running time, an FPT running time. The theory of parameterized complexity was developed by Downey and Fellows [7]. For recent developments, see the book by Flum and Grohe [8].

Kernelization A *kernelization* algorithm for a parameterized language L is a polynomial time procedure which takes as input an instance (x, k), where k is the parameter and returns an instance (x_1, k_1) such that $(x, k) \in L$ if and only if $(x_1, k_1) \in L$ and $|x_1| \leq h(k)$ and $k_1 \leq g(k)$, for some computable functions h, g. The returned instance is said to be the kernel for L.

3 Split Vertex Deletion

In this section, we first present an $\mathcal{O}^*(2^k)$ parameterized algorithm for SVD by combining the technique of iterative compression along with a linear bound on the number of split partitions of split graphs. This bound has been improved by Cygan and Pilipczuk [5], but we give this algorithm for completeness. Later, we give a cubic kernel for SVD.

3.1 An $\mathcal{O}^*(2^k)$ Algorithm for Split Vertex Deletion

We start by stating a lemma, that is implied by Theorem 6.2, [13].

Lemma 2 (Theorem 6.2, [13]) *A split graph on n vertices can have at most $n + 1$ split partitions.*

We will now describe the application of the iterative compression technique to the SVD problem.

Iterative Compression for Split Vertex Deletion Given an instance $(G = (V, E), k)$ of SVD, we let $V = \{v_1, \ldots, v_n\}$ and define vertex sets $V_i = \{v_1, \ldots, v_i\}$, and let the graph $G_i = G[V_i]$. We iterate through the instances (G_i, k) starting from $i = k + 3$. For the ith instance, we try to find a solution $\hat{S}_i$ of size at most k, with the help of a *known* solution S_i of size at most $k + 1$. Formally, the compression problem we address is the following.

Split Vertex Deletion Compression (SVD Compression)	
Input:	Graph $G = (V, E)$, an svd set $S \subseteq V$ of size at most $k + 1$, integer k
Parameter:	k
Question:	Does there exist an svd set of size at most k?

We reduce the SVD problem to $n-k-2$ instances of the SVD COMPRESSION problem as follows. Let $I_i = (G_i, S_i, k)$ be the ith instance. Clearly, the set V_{k+1} is a solution of size at most $k+1$ for the instance I_{k+3}. It is also easy to see that if $\hat{S}_{i-1}$ is a solution of size at most k for instance I_{i-1}, then the set $\hat{S}_{i-1} \cup \{v_i\}$ is a solution of size at most $k+1$ for the instance I_i. We use these two observations to start off the iteration with the instance $(G_{k+3}, S_{k+3} = V_{k+1}, k)$ and look for a solution of size at most k for this instance. If there is such a solution $\hat{S}_{k+3}$, we set $S_{k+4} = \hat{S}_{k+3} \cup \{v_{k+4}\}$ and ask for a solution of size at most k for the instance I_{k+4} and so on. If, during any iteration, the corresponding instance does not have a solution of the required size, it implies that the original instance is also a NO instance. This follows from the fact that if a graph G has a split vertex deletion set of size k, then any vertex induced subgraph of G also has a split vertex deletion set of size k. Finally, the solution for the original input instance will be $\hat{S}_n$. Since there can be at most n iterations, the total time taken to solve the original instance is bounded by n times the time required to solve the SVD COMPRESSION problem.

Our algorithm for SVD COMPRESSION is as follows. Let the input instance be $I = (G = (V, E), S, k)$. We guess a subset $Y \subseteq S$ with the intention of picking these vertices in our hypothetical solution for this instance and ignoring the rest of the vertices in S. We delete the set Y from the graph and decrease k appropriately. We then check if the graph $G[S \setminus Y]$ is a split graph and if it is not, then reject this guess of Y as a spurious guess. Suppose that $G[S \setminus Y]$ is indeed a split graph. We now guess and fix a split partition $(C_0 \uplus I_0)$ for this graph. By Lemma 2, we know that there are at most $k+2$ such split partitions. The split partition we fix corresponds to the split partition *induced* by the hypothetical solution on the graph $G[S \setminus Y]$. Hence, it now remains to check if there is an SVD set of the appropriate size which is disjoint from $S \setminus Y$, and results in a split graph with a split partition *consistent* with $(C_0 \uplus I_0)$. More formally, we have an instance of the following problem.

SPLIT VERTEX DELETION COMPRESSION* (SVD COMPRESSION*)	
Input:	Graph $G = (V, E)$, an svd set $S \subset V$ such that $G[S]$ is a split graph, a split partition $(C_0 \uplus I_0)$ for the graph $G[S]$, integer k
Parameter:	k
Question:	Does there exist an svd set X of size at most k, disjoint from S such that $G \setminus X$ has a split partition consistent with $(C_0 \uplus I_0)$?

The following lemma gives a polynomial time algorithm for the above problem.

Lemma 3 SPLIT VERTEX DELETION COMPRESSION* *can be solved in* $\mathcal{O}(n^3)$ *time.*

Proof Let S' be a potential solution, and let $(C' \uplus I')$ be a fixed split partition for the graph $G \setminus S'$ consistent with the split partition $(C_0 \uplus I_0)$. Let $(C_1 \uplus I_1)$ be a split partition of the graph $G \setminus S$.

Since we cannot delete edges, at most one vertex of C_1 can lie in I' and at most one vertex of I_1 can lie in C'. Hence, we initially guess these two vertices (either guess could be empty) v_c and v_i where $v_c = C_1 \cap I'$ and $v_i = I_1 \cap C'$. We move v_c to I_1 and v_i to C_1. For the sake of convenience we refer to the modified sets C_1 and I_1 also as C_1 and I_1. Now, let $\hat{I} = I_0 \cup I_1$ and $\hat{C} = C_0 \cup C_1$. It is clear that any vertex

in $\hat{I}$ which is neighbor to a vertex in $I_0 \cup \{v_c\}$ needs to be deleted and any vertex in $\hat{C}$ which is not global to $C_0 \cup \{v_i\}$ needs to be deleted. Let X be the set of these vertices, that need to be deleted. Now, if X is not disjoint from S, we return NO. On the other hand, we observe that if X is disjoint from S, then deleting X gives us the required kind of split graph. To show that, we look at the partition $(\hat{C} \setminus X) \uplus (\hat{I} \setminus X)$ of $G \setminus X$. The set $\hat{C} \setminus X$ is a clique because $C_0 \cup \{v_i\}$ is a clique (otherwise we would have returned NO earlier), $C_1 \setminus X$ is a clique, and all the edges between $C_0 \cup \{v_i\}$ and $C_1 \setminus X$ are present. Similarly, the set $\hat{I} \setminus X$ is an independent set because $I_0 \cup \{v_c\}$ is an independent set (otherwise we would have returned NO earlier), $I_1 \setminus X$ is an independent set, and no edges between $I_0 \cup \{v_c\}$ and $I_1 \setminus X$ are present. Hence, if $|X| \leq k$, then we return that it is indeed a YES instance and return NO otherwise. Guessing the vertices takes $\mathcal{O}(n^2)$ time and in each iteration we spend at most linear time, hence the algorithm takes $\mathcal{O}(n^3)$ time. □

Given Lemma 3, our algorithm for SVD COMPRESSION has a running time of $\mathcal{O}(\sum_{i=0}^{k} \binom{k+1}{i} \cdot k \cdot n^{\mathcal{O}(1)}) = \mathcal{O}^*(2^k)$, where the factor of k is due to the number of split partitions of $G[S \setminus Y]$ and $n^{\mathcal{O}(1)}$ is due to the time required to execute our algorithm for SVD COMPRESSION*.

Finally, since we solve at most n instances of SVD COMPRESSION, our algorithm for SVD runs in time $\mathcal{O}^*(2^k)$, giving us the following theorem.

Theorem 1 SPLIT VERTEX DELETION *can be solved in* $\mathcal{O}^*(2^k)$ *time.*

We now show that with respect to the asymptotic dependence on k, the above algorithm for SPLIT VERTEX DELETION is essentially the best we can hope for.

Theorem 2 SPLIT VERTEX DELETION *cannot be solved in time* $\mathcal{O}^*(2^{o(k)})$ *time unless ETH fails.*

Proof It is known that VERTEX COVER does not admit a subexponential algorithm unless the Exponential Time Hypothesis (ETH) fails [4]. We prove the analogous statement for SPLIT VERTEX DELETION by a reduction from VERTEX COVER.

Consider an instane (G, k) of VERTEX COVER and let G' be the graph constructed from G by adding a disjoint clique of size $k + 2$. Now, G has a vertex cover of size at most k if and only if G' has a svd set of size at most k. This concludes the proof. □

3.2 A Cubic Kernel for SPLIT VERTEX DELETION

In this subsection, we use the structural claim made in the algorithm for SVD to design a vertex kernel of size $\mathcal{O}(k^3)$ for SVD. We design the kernel by introducing reduction rules which can be applied in polynomial time to reduce the instance. The reduction rules we present here are applied exhaustively and in the order in which they are presented.

We say that a reduction rule that is applied on an instance (G, k) to produce an instance (G', k') is *correct* if (G, k) is a YES instance if and only if (G', k') is a YES instance.

Reduction Rule 3.1 *Compute an inclusion wise maximal set of vertex disjoint forbidden structures greedily, and let the set of vertices involved in this set of forbidden structures be D. If $|D|$ exceeds $5k$, then return a trivial* NO *instance.*

Moving forward, we assume that $|D| \leq 5k$. Note that $G \setminus D$ is a split graph, and let $(C^* \uplus I^*)$ be a split partition of this graph. Before we present the next reduction rule, we need the following definition.

Definition 2 We say that a vertex v of G has a *high clique non-neighborhood* if $|C^* \setminus N(v)| \geq k+2$. Similarly, v is said to have a *high independent set neighborhood* if $|I^* \cap N(v)| \geq k+2$.

Let $H_i = \{x \in V : |C^* \setminus N(v)| \geq k+2\}$, and let $H_c = \{x \in V : |I^* \cap N(v)| \geq k+2\}$.

Lemma 4 *The vertices in H_i will either end up in the independent partition of the resulting split graph, or will get deleted and hence will be in the solution. Similarly for vertices in H_c, will either end up in the clique partition of the resulting split graph, or will get deleted and hence will be in the solution.*

Proof The vertices in H_i have at least $k+2$ non-neighbors in the clique partition. So, if a vertex of H_i is moved to clique partition, out of $k+2$ non-neighbors, at most k can be deleted, and at most one could be moved to the independent partition, which leads to a contradiction to the fact that there exists a set S of size k, such that $G \setminus S$ is a split graph. Similarly, the vertices in H_c will either end up in the clique partition of the remaining split graph or will get deleted and hence will be in the solution. □

The previous lemma justifies the correctness of the next reduction rule.

Reduction Rule 3.2 *If there is a vertex $v \in H_i \cap H_c$, then delete v and decrease k by* 1.

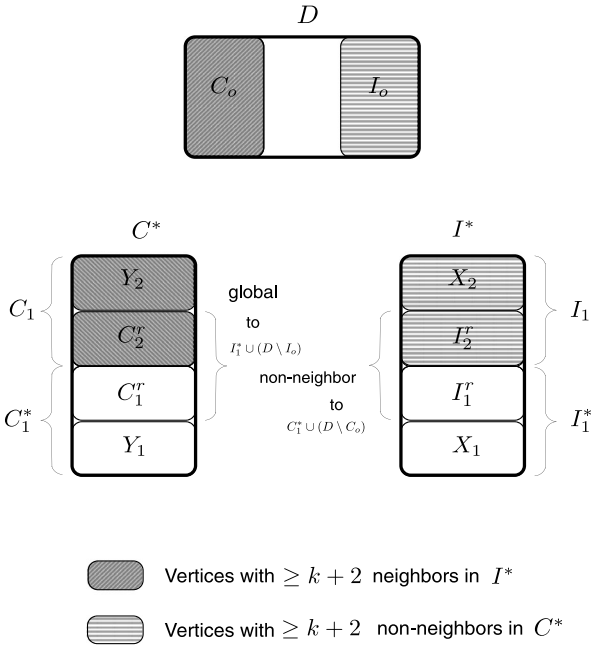
Lemma 5 *Reduction Rule 3.2 is correct.*

Proof Let Z_1 be a clique non-adjacent to v and Z_2 be an independent set adjacent to v, such that $|Z_1| \geq (k+2)$ and $|Z_2| \geq (k+2)$. It is sufficient to show that v is part of every solution of size at most k. Suppose that v is not in some solution S of size at most k. Let $(C \uplus I)$ be a split partition of $G \setminus S$. We first consider the case when $v \in C$. Then, the set $Z_1 \setminus S$, which contains at least two elements, lies in I, which is not possible. Now, consider the case when $v \in I$. In this case, the set $Z_2 \setminus S$, which contains at least two elements, lies in C, which is also a contradiction. □

We now partition the vertex set of the resulting graph G as follows (see Fig. 1 for a schematic diagram).

– Let $C_1 = H_c \cap C^*$ be the set of vertices of C^* which have high independent set neighborhood, and let $I_1 = H_i \cap I^*$, is the set of vertices of I^* which have high clique non-neighborhood.

Fig. 1 Partitions of G



- Similarly $C_o = H_c \cap D$, is the set of vertices of D which have high independent set neighborhood, and $I_o = H_i \cap D$, is the set of vertices of D which have high clique non-neighborhood.
- Let $C_1^* = C^* \setminus C_1$, and $I_1^* = I^* \setminus I_1$.
- Let $Y_1 \subseteq C_1^*$ and $Y_2 \subseteq C_1$ be sets of vertices which have a non-neighbor in $(D \setminus I_o) \cup I_1^*$. Let $C_1^r = C_1^* \setminus Y_1$ and $C_2^r = C_1 \setminus Y_2$ (i.e. C_1^r and C_2^r are global to $(D \setminus I_o) \cup I_1^*$).
- Let $X_1 \subseteq I_1^*$ and $X_2 \subseteq I_1$ be sets of vertices which have a neighbor in $(D \setminus C_o) \cup C_1^*$. Let $I_1^r = I_1^* \setminus X_1$ and $I_2^r = I_1 \setminus X_2$ (i.e. I_1^r and I_2^r do not have any edges to $(D \setminus C_o) \cup C_1^*$).

Before giving further reduction rules, we prove the following lemmas.

Lemma 6 *Let $X \subseteq C^*$, such that $|X| > k+2$ be global to $I_1^* \cup (D \setminus I_o)$. Let G' be the graph obtained by deleting all but $k+2$ vertices of X and deleting all the edges between X and $I_1 \cup I_o$. Then, (G, k) is a* YES *instance of* SVD *if and only if (G', k) is a* YES *instance of* SVD.

Proof Let $\bar{X}$ be the truncated clique. Suppose that (G, k) is a YES instance and let S be a solution to this instance. We claim that there is a solution S_1 for the instance (G, k), disjoint from X. If S itself is disjoint from X, then we are done. Suppose that S contained some vertex v of X. We simply remove this vertex from S and add it to the clique of the split partition of $G \setminus S$. Since the only vertices v is non-adjacent to, are the vertices of $I_1 \cup I_o$ and these do not lie in the clique partition of $G \setminus S$ anyway (by Lemma 4), the resulting partition is also a split partition. Hence, we may

assume that the solution S is disjoint from X. Now, we know that at most one vertex of X can lie in the independent partition of $G \setminus S$. Since this vertex does not have any non-neighbor in the clique partition of $G \setminus S$, we may move this vertex to the clique partition as well. Now, we claim that S is also a solution for the reduced instance (G', k). We have proved that there exists a split partition of $G \setminus S$ such that all the vertices of X lie in the clique. Also, since S is disjoint from X, after truncating X, the clique partition of $G \setminus S$ remains clique (we delete edges only to $I_1 \cup I_o$, which can not be in the clique side), while the independent set side is unaffected. Hence, S is also a solution for (G', k).

For the converse direction, suppose (G', k) has a solution S'. Since the vertices of $I_1 \cup I_o$ have high clique non-neighborhood in G', they are either in S', or in the final independent partition (by Lemma 4). Analogous to the argument in the above paragraph, we can argue that $\bar{X}$ is disjoint from S' and lies in the clique partition. Now, observe that replacing $\bar{X}$ with X results in a split partition of the graph $G \setminus S'$, implying that (G, k) is a YES instance. This concludes the proof of correctness of this reduction rule. □

The above lemma has an analogous counterpart in the case when $X \subseteq I^*$, $|X| > k+2$ and X does not have any edges to $C_1^* \cup (D \setminus C_o)$. The proof is identical, except we now consider independent sets where we considered cliques and we consider neighbors where we considered non-neighbors. We simply state the lemma without proof.

Lemma 7 *Let $X \subseteq I^*$ be such that $|X| > k+2$ and X does not have any edges to $C_1^* \cup (D \setminus C_o)$. Let G' be the graph obtained by deleting all but $k+2$ vertices of X and adding all the edges between X and $C_1 \cup C_o$. Then, (G, k) is a* YES *instance of* SVD *if and only if (G', k) is a* YES *instance of* SVD.

Now,we are ready to give the reduction rules, which will help us bound the size of C^* and I^*.

Reduction Rule 3.3 *If $|C_1^r| > k+2$, then delete all edges between C_1^r and $I_1 \cup I_o$ and delete all but $k+2$ vertices of C_1^r.*

Reduction Rule 3.4 *If $|C_2^r| > k+2$, then delete all edges between C_2^r and $I_1 \cup I_o$ and delete all but $k+2$ vertices of C_2^r.*

Since C_1^r and C_2^r are global to $(D \setminus I_o) \cup I_1^*)$, the correctness of these reduction rules follows immediately from Lemma 6. Analogous to Reduction Rules 3.3 and 3.4, we get the following rules for the independent set side.

Reduction Rule 3.5 *If $|I_1^r| > k+2$, then add all edges between I_1^r and $C_1 \cup C_o$ and delete all but $k+2$ vertices of I_1^r.*

Reduction Rule 3.6 *If $|I_2^r| > k+2$, then add all edges between I_2^r to $C_1 \cup C_o$ and delete all but $k+2$ vertices of I_2^r.*

The correctness of these reduction rules follows from Lemma 7. Now, towards bounding the size of the sets, we first show that the size of at least one of the sets, C_1^* and I_1^* is bounded by a linear function of k.

Lemma 8 *When none of the reduction rules apply,* $|C_1^*| \leq 2k+2$ *or* $|I_1^*| \leq 2k+2$.

Proof As no vertex in C_1^* has high independent set neighborhood, the number of edges between the sets I_1^* and C_1^* is at most $|C_1^*|(k+1)$. Also, since no vertex in I_1^* has high clique non-neighborhood, the number of edges between the sets I_1^* and C_1^* is at least $|I_1^*|(|C_1^*| - (k+1))$. This implies that $(|C_1^*| + |I_1^*|)(k+1) \geq |C_1^*| \cdot |I_1^*|$. Substituting $|C_1^*| = 2k + c_1$ and $|I_1^*| = 2k + c_2$, where $c_1, c_2 > 2$, we get the following.

$$
\begin{aligned}
&(2k + c_1 + 2k + c_2)(k+1) \geq (2k + c_1)(2k + c_2) \\
&\quad \Rightarrow\ 4k^2 + 4k + (c_1 + c_2)k + c_1 + c_2 \geq 4k^2 + 2k(c_1 + c_2) + c_1 c_2 \\
&\quad \Rightarrow\ (c_1 + c_2 - c_1 c_2) + (4k - (c_1 + c_2)k) \geq 0
\end{aligned}
$$

which can not be true, since for $c_1, c_2 > 2$, $(c_1 + c_2) < c_1 c_2$ and $4k < (c_1 + c_2)k$ and hence we get a contradiction. □

Lemma 9 *When none of the reduction rules apply, the number of vertices in* $C_1^* \cup I_1^*$ *is* $\mathcal{O}(k^2)$.

Proof **Case 1: When $|I_1^*| \leq 2k+2$.** We observe that the size of C_1^r is already bounded by Reduction Rule 3.3, so to bound the size of $C_1^* \cup I_1^*$, we only need to bound the size of Y_1. All the vertices in Y_1 have at least one non-neighbor in $(D \setminus I_o) \cup I_1^*$. Also, we know that $(D \setminus I_o) \cup I_1^*$ has $\mathcal{O}(k)$ vertices and each such vertex has at most $\mathcal{O}(k)$ non-neighbors in C^*. Hence, the number of vertices in Y_1 is $\mathcal{O}(k^2)$. This gives a total bound of $\mathcal{O}(k^2)$ on size of $C_1^* \cup I_1^*$.
Case 2: When $|C_1^*| \leq 2k+2$. We observe that the size of I_1^r is already bounded by Reduction Rule 3.5, so to bound the size of $C_1^* \cup I_1^*$, we only need to bound the size of X_1. All the vertices in X_1 have at least one neighbor in $(D \setminus C_o) \cup C_1^*$. Also, we know that $(D \setminus C_o) \cup C_1^*$ has $\mathcal{O}(k)$ vertices and each such vertex has at most $\mathcal{O}(k)$ neighbors in I^*. Hence, the number of vertices in X_1 is $\mathcal{O}(k^2)$. This gives a total bound of $\mathcal{O}(k^2)$ on size of $C_1^* \cup I_1^*$. □

We observe that the only unbounded sets at this point in C^* and I^* are X_2 and Y_2 respectively. All the other sets are bounded by $\mathcal{O}(k^2)$. In the next lemma, we bound the sizes of C^* and I^*.

Lemma 10 *When none of the reduction rules apply, the sets* C^* *and* I^* *contain* $\mathcal{O}(k^3)$ *vertices.*

Proof As stated earlier, bounding the size of X_2 and Y_2 by $\mathcal{O}(k^3)$ will give us the desired result, as all the other sets in C^* and I^* are already bounded by $\mathcal{O}(k^2)$. Notice that all the vertices in Y_2 have a non-neighbor in $(D \setminus I_o) \cup I_1^*$, which has $\mathcal{O}(k^2)$ vertices. Also, any vertex in $(D \setminus I_o) \cup I_1^*$ has $\mathcal{O}(k)$ non-neighbors in C^* and $Y_2 \subseteq C^*$. This gives a bound of $\mathcal{O}(k^3)$ on size of Y_2.

Similarly, all the vertices in X_2 have a neighbor in $(D \setminus C_o) \cup C_1^*$, which has $\mathcal{O}(k^2)$ vertices. Also, any vertex in $(D \setminus C_o) \cup C_1^*$ has $\mathcal{O}(k)$ neighbors in I^* and $X_2 \subseteq I^*$. This gives a bound of $\mathcal{O}(k^3)$ on size of X_2. □

Summing up the bounds we have obtained, leads to the following theorem.

Theorem 3 *There is a vertex kernel for* SVD *with* $\mathcal{O}(k^3)$ *vertices.*

4 Split Edge Deletion

In this section, we present a subexponential algorithm for SED using the *Color and Conquer* approach introduced by Alon, Lokshtanov and Saurabh [2]. We first design a randomized subexponential algorithm for this problem which succeeds with high probability. We then describe a way of derandomizing this algorithm to obtain a deterministic algorithm.

4.1 A Randomized Subexponential Algorithm for SED

This algorithm consists of three steps. In the first step, we reduce the instance (G, k) to an equivalent instance (G', k') with $\mathcal{O}(k^2)$ vertices. In the second step, we color the vertices of the graph uniformly at random and we prove that with a sufficiently high probability, all the edges of some k-sized solution (if one exists) are non-monochromatic. Finally, we give an algorithm to check if a colored instance of SED has a non-monochromatic split edge deletion set of size at most k.

Kernelization We first apply the kernelization algorithm (see Sect. 4.3) which, given an instance (G, k) of SED, in polynomial time, returns an equivalent instance (G', k') of SED such that the number of vertices in G' is $\mathcal{O}(k^2)$ and $k' \leq k$. In the rest of this section, we will assume that the given instance is an instance of this kind.

Probability of a Good Coloring We now color the vertices of G independently and uniformly at random with $\sqrt{8k}$ colors and let A_c be the set of non-monochromatic edges. Suppose that $(G = (V, E), k)$ is a YES instance and let $S \subseteq E$ be a solution to this instance. We now show that the probability of S being contained in A_c is at least $2^{-\mathcal{O}(\sqrt{k})}$. We begin by estimating the probability of obtaining a proper coloring (making all the edges non-monochromatic) when applying the above random experiment on a graph with k edges.

Lemma 11 ([2]) *If the vertices of a graph on q edges are colored independently and uniformly at random with $\sqrt{8q}$ colors then the probability that G is properly colored is at least $(2e)^{-\sqrt{q/8}}$.*

Now, since we colored each vertex of the graph G independently, the graph induced on the set S, of size at most k, will be properly colored with probability at least $2^{-\mathcal{O}(\sqrt{k})}$, which gives us the following lemma.

Lemma 12 *Let $(G = (V, E), k)$ be a* YES *instance of* SED *which is colored by the random process described above, and let $S \subset E$ be a solution for this instance. The probability that no edge in S is monochromatic is at least $2^{-\mathcal{O}(\sqrt{k})}$.*

Solving a Colored Instance. We now present an algorithm to test if there is a *colorful* (all edges non-monochromatic) split edge deletion set in a given colored instance of SED. In the colored instance, every vertex is colored with one of $\sqrt{8k}$ colors. We start with the following simple observation.

Observation 1 *Let $G = (V_1 \cup V_2 \cup \cdots \cup V_t, E)$ be a t-colored graph with color classes $V_1, \ldots, V_t$. If there exists a colorful split edge deletion set S in G, then $G[V_i]$ is a split graph for every V_i.*

We now proceed to the description of the algorithm. Suppose the given instance had a colorful split edge deletion set S. Observation 1 implies that $G[V_i]$ is a split graph and it remains a split graph in $G \setminus S$. Hence, we use Lemma 2 to enumerate the split partitions of $G[V_i]$ for each i. Fixing a split partition for each $G[V_i]$ results in a *combined* split partition for the vertices in V. There are $\mathcal{O}(k^2)$ split partitions for each V_i and $\mathcal{O}(\sqrt{k})$ such sets. Hence, there are $k^{\mathcal{O}(\sqrt{k})}$ combined split partitions. Now, it simply remains to check if there is a combined split partition $(C \uplus I)$ such that the number of edges in the graph $G[I]$ is at most k and return YES if and only if there is such a combined split partition. Hence, we have the following lemma.

Lemma 13 *Given a colored instance (G, k) of* SED *of size $\mathcal{O}(k^2)$, we can test if there is a colorful SED set of size at most k in time $2^{\mathcal{O}(\sqrt{k}\log k)}$.*

Combining Lemmas 12 and 13, we get the following theorem.

Theorem 4 *There is a randomized FPT algorithm for* SED *running in time $2^{\mathcal{O}(\sqrt{k}\log k)} + n^{\mathcal{O}(1)}$ with a success probability of at least $2^{-\mathcal{O}(\sqrt{k})}$.*

4.2 Derandomization with Universal Coloring Families

For integers m, k and r, a family F of functions from $[m]$ to $[r]$ is called a universal (m, k, r)-coloring family if, for any graph G on the set of vertices $[m]$ with at most k edges, there exists a function $f \in F$ which gives a proper vertex coloring of G. Suppose the kernel we obtain has size bounded by ck^2, then an explicit construction of a $(ck^2, k, \sqrt{8k})$-coloring family is known to exist.

Theorem 5 ([2]) *There exists an explicit universal $(ck^2, k, \sqrt{8k})$-coloring family F of size at most $2^{\mathcal{O}(\sqrt{k}\log k)}$.*

Instead of the randomized coloring step of our algorithm, we can try each function in the universal coloring family given by Theorem 5. Hence, we have the following theorem.

Theorem 6 *There is an algorithm which solves* SED *in time* $2^{\mathcal{O}(\sqrt{k}\log k)} + n^{\mathcal{O}(1)}$.

4.3 Improved Kernel for SED

In this subsection, we use a subset of the reduction rules for SED given in [14] to show the existence of a kernel with a quadratic number of vertices. The following are the reduction rules which we will apply on the given instance.

Reduction Rule 4.1 ([14]) *Delete vertices from G which are not part of an induced subgraph isomorphic to $2K_2$, C_4 or C_5.*

From this point on, we refer to an induced subgraph isomorphic to $2K_2$, C_4 or C_5, as an induced $2K_2$, C_4 or C_5 respectively. When Reduction Rule 4.1 no longer applies, every vertex in G is part of some induced $2K_2$, C_4 or C_5.

Reduction Rule 4.2 ([14]) *If two adjacent edges (u, v) and (u, w) occur together in more than k induced C_4s, then delete (u, v) and (u, w) from G and add two edges (a, v) and (b, w), where a and b are two new vertices of degree* 1.

Reduction Rule 4.3 ([14]) *If an edge e occurs in more than k induced $2K_2$'s, then delete e from G and reduce k by one.*

We refer to [14] for the correctness of these reduction rules. We apply the above reduction rules exhaustively, in the order in which they are presented, and obtain a reduced instance (G', k'). For the sake of notational convenience, we denote the reduced instance by (G, k). In the rest of this discussion, we will assume that the reduced instance is a YES instance and prove a bound on the size of the instance with this assumption. Let S be a minimal solution with at most k edges and let $(C \uplus I)$ be a split partition of the graph $G \setminus S$. We call a vertex of G *affected* if some edge in S is incident on it, and *unaffected* otherwise. Observe that there are at most $2k$ affected vertices in G. We now make the following important observation.

Observation 2 *All the affected vertices lie in the independent set I.*

Proof Suppose there was an affected vertex in the clique partition C. Then, adding back the edges in S which are incident on this affected vertex in the clique partition also results in a split partition, which contradicts the minimality of S. □

Lemma 14 *Every induced C_4 in G intersects S in exactly one edge, or in exactly two adjacent edges of C_4 or in all the four edges.*

Proof We prove the lemma by considering an induced C_4, $\{v_1, v_2, v_3, v_4\}$ and the set of affected vertices of this C_4. Since at least one edge of the C_4 has to be deleted, at

least 2 vertices are affected. If exactly 2 vertices are affected, then it must be the case that exactly one edge of the C_4 is present in S. If exactly 3 vertices, say v_1, v_2 and v_3 are affected, then by Observation 2, these vertices lie in the independent partition and hence the edges (v_1, v_2) and (v_2, v_3) are contained in S. Since v_4 is unaffected, no other edge of this C_4 is in S. Finally, in the case when all four vertices are affected, since they all must lie in the independent partition, S contains all 4 edges of this C_4. This completes the proof of the lemma. □

Lemma 15 *Every induced C_5 in G intersects S in exactly two adjacent edges of C_5 or in exactly three adjacent edges of C_5 or in all the five edges.*

Proof We fix a C_5, say $\{v_1, v_2, v_3, v_4, v_5\}$, and prove this lemma in the same way as the previous one. If only one edge, say (v_1, v_2) of the C_5 is in the solution, then the edges (v_2, v_3) and (v_5, v_1) form an induced $2K_2$, disjoint from S, which is not possible. Hence, at least two edges of the C_5 are in the solution, and at least three vertices are affected. If exactly three vertices are affected, then exactly two adjacent edges of the C_5 are in S, because we have to delete two edges from the cycle affecting only these three vertices, and this is the only possible way. Suppose exactly 4 vertices are affected. Then, by Observation 2, the edges between these vertices must lie in S and hence exactly 3 adjacent edges are in S. Finally, when all the vertices are affected then all the edges of the C_5 lie in S. □

We now give a bound on the number of vertices in the set I, using the following lemmas.

Lemma 16 *There are $\mathcal{O}(k^2)$ vertices which are part of an induced $2K_2$ in G.*

Proof Every edge in the graph is contained in at most k induced $2K_2$'s in G (otherwise Reduction Rule 4.3 will be applicable). Since there is a solution of size at most k, these edges can together be contained in at most $\mathcal{O}(k^2)$ many $2K_2$'s and hence the number of vertices of G involved in an induced $2K_2$ is bounded by $\mathcal{O}(k^2)$. This completes the proof of the lemma. □

Lemma 17 *If $v \in I$ is an unaffected vertex, then v is not part of an induced C_4 or C_5 in G.*

Proof Suppose that v is unaffected and v is part of a C_4. Since at least two vertices of the C_4 are affected, at least one neighbor of v on the C_4 is affected. Since this neighbor also lies in I (see Observation 2), the edge between these two vertices is contained in S, contradicting that v was unaffected.

Now, suppose that v is unaffected and v is part of a C_5. By Lemma 15, we know that at least two edges of the C_5 lie in the solution, which means at least three vertices are affected. Hence, some neighbor of v is affected, implying that v is affected as well. This completes the proof of the lemma. □

Since we have bounded the number of both affected and unaffected vertices in I, we have the following lemma.

Lemma 18 *There are $\mathcal{O}(k^2)$ vertices in the independent set I.*

We now proceed to bound the number of vertices inside the clique C. To do so, we introduce the notion of a *sliced* vertex. For every edge $(p, q) \in S$, and a vertex $v \in C$, we say that the edge (p, q) *slices* v if v is adjacent to p but not to q or vice versa. We say that a vertex $v \in C$ is *sliced* if some edge in S slices it and *unsliced* otherwise. Observe that the sets of sliced and unsliced vertices form a partition of C.

Lemma 19 *If $v \in C$ is not sliced by any edge in S, then v is not part of an induced C_4 or C_5 in G.*

Proof Suppose v was part of a C_4, $\{v_1, v_2, v_3, v_4\}$ where $v = v_1$. Since v is in the clique, it is unaffected (by Observation 2) and (v_1, v_2), (v_4, v_1) are not in S. Since at least one edge from C_4 has to be in the solution, the edge (v_2, v_3) or (v_3, v_4) must be in S. But now, either of these two edges slices v, a contradiction. Suppose v was part of a C_5, $\{v_1, v_2, v_3, v_4, v_5\}$ where $v = v_1$. By Lemma 15, at least 2 edges of the C_5 are in S, implying that at least one of them will slice v, a contradiction. □

Since every unsliced vertex must be part of an induced $2K_2$, by Lemma 16 the number of unsliced vertices in the set C is bounded by $\mathcal{O}(k^2)$. To bound the number of sliced vertices in C, we argue that each edge in S can slice $\mathcal{O}(k)$ vertices of C, resulting in the following lemma.

Lemma 20 *There are $\mathcal{O}(k^2)$ sliced vertices in C.*

Proof Let $e = (p, q)$ be an edge in S. By Observation 2, p and q are in the independent set I. Let $X(e)$ be the set of vertices in C sliced by e, $X(p) = X(e) \cap N(p)$ and $X(q) = X(e) \cap N(q)$. Then $X(e) = X(p) \uplus X(q)$. We will count the vertices of $X(e)$ as follows.

We first consider the following case.

Case 1: The sets $X(p)$ and $X(q)$ are both non-empty. We claim that in this case, $|X(p)|, |X(q)| \leq k$. Fix a vertex $w \in X(q)$. Then, for every vertex $v \in X(p)$, the vertices $\{p, q, w, v\}$ induce a C_4 in G. Hence, if there are more than k vertices in $X(p)$, then there are more than k induced C_4's in G which pairwise have the edges (p, q) and (q, w) in common. But this implies that Reduction Rule 4.2 applies, contradicting the irreducibility of G. Hence, we conclude that $X(p)$ must have at most k vertices. Analogously, we can bound the size of the set $X(q)$ by k.

Case 2: One of the two sets, say $X(q)$ is empty and $X(p)$ is non-empty, and suppose that there is an edge $(q, r) \in S$ such that $(p, r) \notin E$.

Since r is affected, we know that $r \in I$. Let $X_1 = X(p) \cap N(r)$ and $X_2 = X(p) \setminus X_1$. Observe that, for any vertex $v \in X_1$, the vertices $\{p, q, r, v\}$ form an induced C_4 in G. Hence, if there were more than k vertices in X_1, then there would be more than k induced C_4's in G which pairwise have only the edges (p, q) and (q, r) in common. But then Reduction Rule 4.2 applies, which contradicts the irreducibility of the instance. We now move on to bounding the size of the set X_2. Let u and v be

two vertices in X_2 (if there are no two vertices in X_2, we already have the required bound). Since $u, v \notin N(q) \cup N(r)$ the edges (u, v) and (q, r) form an induced $2K_2$ in G. Hence, if the number of vertices in X_2 exceeds $2\sqrt{k}+1$, then there would be more than k induced $2K_2$'s in G which have the edge (q, r) in common. But in this case, Reduction Rule 4.3 applies, a contradiction. Hence, the set $X(e)$ contains at most $2k$ vertices.

Now, we first try to associate every sliced vertex of C to some edge satisfying the premise of Case 1 or 2. We have already argued that there are $\mathcal{O}(k^2)$ such vertices.

Now we bound the remaining sliced vertices by showing that they cannot be part of any induced C_4 or C_5 of G, and hence can only be part of $2K_2$s, and hence they add up to $\mathcal{O}(k^2)$ in number.

Consider a vertex v sliced by an edge $(p, q) \in S$, $v \in X(p)$ and has not been associated with an edge satisfying Case 1 or Case 2. Suppose that v was part of a C_4 $\{v_1, v_2, v_3, v_4\}$ where $v = v_1$. Suppose that the edge (v_2, v_3) is in S. Now, if the vertex v_4 is in C, then v_1 and v_4 are sliced by the edge (v_2, v_3), and hence we will be in the case (Case 1) when $X(p)$ and $X(q)$ are non-empty where $(p, q) = (v_2, v_3)$. Similarly, if v_4 is in I, then we will be in the case (Case 2) when even if $X(p)$ or $X(q)$ is empty, there is an edge $(q, r) = (v_3, v_4)$ in G, such that there is no edge $(p, r) = (v_2, v_4)$.

Similarly, we can show that v cannot be part of an induced C_5. Hence, it must be the case that v is part of an induced $2K_2$. Recall that we have already bounded the number of such vertices by $\mathcal{O}(k^2)$. □

We have thus bounded the number of vertices in C and I, and hence bounded the number of vertices of the graph G, leading to the following theorem.

Theorem 7 *There is a kernel for* SED *with* $\mathcal{O}(k^2)$ *vertices.*

5 Conclusion

In this paper we studied the parameterized complexity of deleting k edges/vertices to get to the class of split graphs. We obtained faster parameterized algorithms as well as smaller sized kernels for these problems. Cygan and Pilipczuk [5] have subsequently improved one of the four results presented in this paper. That is, SVD can be solved in time $\mathcal{O}^*(1.2738^k k^{\mathcal{O}(\log k)})$. Finally, an interesting project could be to systematically identify other parameterized problems that admit subexponential parameterized algorithms on general graphs.

Acknowledgements We wish to thank Venkatesh Raman and Saket Saurabh for the insightful discussions which led to this work. We also thank the anonymous referees and the editors for their valuable comments.

References

1. Abu-Khzam, F.: A kernelization algorithm for d-hitting set. J. Comput. Syst. Sci. **76**(7), 524–531 (2010)

2. Alon, N., Lokshtanov, D., Saurabh, S.: Fast FAST. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S.E., Thomas, W. (eds.) Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, 5–12 July 2009, Proceedings, Part I. Lecture Notes in Computer Science, vol. 5555, pp. 49–58. Springer, Berlin (2009)
3. Cai, L.: Fixed-parameter tractability of graph modification problems for hereditary properties. Inf. Process. Lett. **58**(4), 171–176 (1996)
4. Cai, L., Juedes, D.: On the existence of subexponential parameterized algorithms. J. Comput. Syst. Sci. **67**, 789–807 (2003)
5. Cygan, M., Pilipczuk, M.: Split vertex deletion meets vertex cover: new fixed-parameter and exact exponential-time algorithms. Inf. Process. Lett. **113**(5–6), 179–182 (2013)
6. Demaine, E.D., Fomin, F.V., Hajiaghayi, M.T., Thilikos, D.M.: Subexponential parameterized algorithms on bounded-genus graphs and h-minor-free graphs. J. ACM **52**(6), 866–893 (2005)
7. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer, Berlin (1999)
8. Flum, J., Grohe, M.: Parameterized Complexity Theory. Texts in Theoretical Computer Science. An EATCS Series. Springer, Berlin (2006)
9. Földes, S., Hammer, P.: Split graphs. Congr. Numer. **19**, 311–315 (1977)
10. Fomin, F.V., Kratsch, S., Pilipczuk, M., Pilipczuk, M., Villanger, Y.: Tight bounds for parameterized complexity of cluster editing. In: Portier, N., Wilke, T. (eds.) 30th International Symposium on Theoretical Aspects of Computer Science, STACS 2013, 27 February–2 March 2013, Kiel, Germany, LIPIcs, vol. 20, pp. 32–43. Schloss Dagstuhl—Leibniz-Zentrum fuer Informatik, Wadern (2013)
11. Fomin, F.V., Villanger, Y.: Subexponential parameterized algorithm for minimum fill-in. In: Rabani, Y. (ed.) Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, 17–19 January 2012, pp. 1737–1746. SIAM, Philadelphia (2012)
12. Fujito, T.: A unified approximation algorithm for node-deletion problems. Discrete Appl. Math. **86**, 213–231 (1998)
13. Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs. Academic Press, New York (1980)
14. Guo, J.: Problem kernels for NP-complete edge deletion problems: split and related graphs. In: Proceedings of the 18th International Conference on Algorithms and Computation, ISAAC'07, pp. 915–926. Springer, Berlin, Heidelberg (2007)
15. Hammer, P.L., Simeone, B.: The splittance of a graph. Combinatorica **1**(3), 275–284 (1981)
16. Heggernes, P., Mancini, F.: Minimal split completions. Discrete Appl. Math. **157**(12), 2659–2669 (2009)
17. Heggernes, P., van 't Hof, P., Jansen, B.M.P., Kratsch, S., Villanger, Y.: Parameterized complexity of vertex deletion into perfect graph classes. In: Owe, O., Steffen, M., Telle, J.A. (eds.) Proceedings of the Fundamentals of Computation Theory—18th International Symposium, FCT 2011, Oslo, Norway, 22–25 August 2011. Lecture Notes in Computer Science, vol. 6914, pp. 240–251. Springer, Berlin (2011).
18. Lewis, J.M., Yannakakis, M.: The node-deletion problem for hereditary properties is NP-complete. J. Comput. Syst. Sci. **20**(2), 219–230 (1980)
19. Lokshtanov, D., Narayanaswamy, N.S., Raman, V., Ramanujan, M.S., Saurabh, S.: Faster parameterized algorithms using linear programming. CoRR abs/1203.0833 (2012)
20. Lund, C., Yannakakis, M.: On the hardness of approximating minimization problems. J. ACM **41**, 960–981 (1994)
21. Marx, D.: Chordal deletion is fixed-parameter tractable. Algorithmica **57**(4), 747–768 (2010)
22. Marx, D., Schlotter, I.: Obtaining a planar graph by vertex deletion. Algorithmica **62**(3–4), 807–822 (2012)
23. Natanzon, A., Shamir, R., Sharan, R.: Complexity classification of some edge modification problems. Discrete Appl. Math. **113**(1), 109–128 (2001)
24. Tyshkevich, R.I., Chernyak, A.A.: Yet another method of enumerating unmarked combinatorial objects. Math. Notes - Ross. Akad. **48**, 1239–1245 (1990)