On fixed-parameter algorithms for SPLIT VERTEX DELETION

Marek Cygan * Marcin Pilipczuk †

Abstract

In the SPLIT VERTEX DELETION problem, given a graph G and an integer k, we ask whether one can delete k vertices from the graph G to obtain a *split graph* (i.e., a graph, whose vertex set can be partitioned into two sets: one inducing a clique and the second one inducing an independent set). In this paper we study fixed-parameter algorithms for SPLIT VERTEX DELETION parameterized by k: we show that, up to a factor quasipolynomial in k and polynomial in n, the SPLIT VERTEX DELETION problem can be solved in the same time as the well-studied VERTEX COVER problem. Plugging the currently best fixed-parameter algorithm for VERTEX COVER due to Chen et al. [TCS 2010], we obtain an algorithm that solves SPLIT VERTEX DELETION in time $\mathcal{O}(1.2738^k k^{\mathcal{O}(\log k)} + n^{\mathcal{O}(1)})$.

To achieve our goal, we prove the following structural result that may be of independent interest: for any graph G we may compute a family $\mathcal{P}$ of size $n^{\mathcal{O}(\log n)}$ containing partitions of $V(G)$ into two parts, such for any two disjoint sets $X_C, X_I \subseteq V(G)$ where $G[X_C]$ is a clique and $G[X_I]$ is an independent set, there is a partition in $\mathcal{P}$ which contains all vertices of X_C on one side and all vertices of X_I on the other.

1 Introduction

The family of vertex deletion, or, more generally, graph modification problems, has been studied very intensively, both in theory and in practice. As in many cases we expect the number of allowed modifications to be small, compared to the size of the input graph, and most graph modification problems turned out to be NP-hard (e.g., all vertex deletion problems for nontrivial hereditary graph classes, by the classical result of Lewis and Yannakakis [9]), it is natural to study these problems from the parameterized point of view, considering parameterization by the solution size (the number of allowed modifications).

In the parameterized setting we assume that each instance is equipped with an additional value k — a parameter which aims to reflect the instance complexity. The goal is

*IDSIA, University of Lugano, Switzerland, marek@idsia.ch. Partially supported by the ERC Starting Grant NEWNET 279352 and Foundation for Polish Science.

†Institute of Informatics, University of Warsaw, Poland, malcin@mimuw.edu.pl, Partially supported by NCN grant N206567140 and Foundation for Polish Science.

to provide an algorithm (called a fixed-parameter algorithm) with $f(k)n^{\mathcal{O}(1)}$ time complexity, where n is the instance size and f is a function independent of n. Observe that such an algorithm is polynomial for any constant value of k and moreover the degree of the polynomial is independent of the parameter value. For more information about the parameterized complexity in general, we refer to three monographs [5, 6, 11].

In this paper we focus on one particular graph modification problem, namely the SPLIT VERTEX DELETION problem (SPLITVD for short). Here, we are given an n-vertex graph G and an integer k and the task is to delete k vertices from G to obtain a *split graph*: a graph H is called a *split graph* if $V(H)$ can be partitioned into two parts X_C and X_I, such that $H[X_C]$ is a clique and $H[X_I]$ is an independent set.[1] Note that the partition (X_C, X_I) does not need to be unique; for example, an n-vertex clique is a split graph with $n+1$ different valid partitions.

As the class of split graphs is hereditary, by the result of Lewis and Yannakakis [9], SPLITVD is NP-hard. Földes and Hammer [7] proved that the class of split graphs is exactly the class of $\{2K_2, C_4, C_5\}$-free graphs; by the general result of Cai [2], this observation yields a fixed-parameter algorithm with running time $\mathcal{O}(5^k n^{\mathcal{O}(1)})$. The dependency on k has been subsequently improved to $\mathcal{O}(2.32^k n^{\mathcal{O}(1)})$ by Lokshtanov et al. [10] and $\mathcal{O}(2^k n^{\mathcal{O}(1)})$ by Ghosh et al. [8]. In this paper we show that SPLITVD can be solved essentially in the same time as the well-studied VERTEX COVER problem.

Theorem 1.1. *If there exists an algorithm that solves the* VERTEX COVER *problem parameterized by the solution size k on n-vertex graphs in $f(k,n)$ time and $g(k,n)$ space, then the* SPLIT VERTEX DELETION *problem on n-vertex graphs can be solved in $\mathcal{O}(f(k,n)k^{\mathcal{O}(\log k)} + n^{\mathcal{O}(1)})$ time and $\mathcal{O}(g(k,n) + n^{\mathcal{O}(1)})$ space.*

By plugging in the currently fastest known algorithm for VERTEX COVER [3], we obtain the following.

Corollary 1.2. *The* SPLIT VERTEX DELETION *problem can be solved in $\mathcal{O}(1.2738^k k^{\mathcal{O}(\log k)} + n^{\mathcal{O}(1)})$ time and polynomial space.*

Note that there exists a straightforward reverse reduction: given a VERTEX COVER instance (G,k) (i.e., we ask for a vertex cover of size k in the graph G), it is easy to see that an equivalent SPLITVD instance (G',k) can be created by defining the graph G' to be a disjoint union of the graph G and a clique on $k+2$ vertices. Thus, we obtain that — up to a factor quasipolynomial in k and polynomial in n — the optimal time complexities of fixed-parameter algorithms for VERTEX COVER and SPLITVD are equal.

The core difficulty of the proof of Theorem 1.1 lies in the following structural result that may be of independent interest.

Theorem 1.3. *For any n-vertex graph G there exists a family $\mathcal{P}$ of partitions (V_C, V_I) of the vertex set $V(G)$, such that the following holds.*

[1] Through the paper we use standard graph notation, see e.g. [4]. In particular, for a given graph G, by $V(G)$ and $E(G)$ we denote its vertex and edge set, respectively. For a set $X \subseteq V(G)$, $G[X]$ is a subgraph induced by X. For a vertex $v \in V(G)$, $N_G(v)$ denotes the set of neighbours of v and $N_G[v] = N_G(v) \cup \{v\}$.

1. *For any set $X \subseteq V(G)$ such that $G[X]$ is a split graph, and any partition (X_C, X_I) of X, such that $G[X_C]$ is a clique and $G[X_I]$ is an independent set, there exists a partition $(V_C, V_I) \in \mathcal{P}$ such that $X_C \subseteq V_C$ and $X_I \subseteq V_I$.*

2. $|\mathcal{P}| \leq 4 \cdot (2n)^{2\lfloor \log n \rfloor + 1}$.

Moreover, there exists an algorithm that enumerates (with possible repetitions) the family $\mathcal{P}$ and runs in time $\mathcal{O}(n^{2\lfloor \log n \rfloor + \mathcal{O}(1)})$ and polynomial space.

Theorem 1.3 is proven in Section 2. Equipped with this structural result, in Section 3 we show that Theorem 1.1 follows easily by combining an already known preprocessing routine for SPLITVD that outputs an equivalent instance of size polynomial in k (called a *polynomial kernel*), Theorem 1.3 and a simple observation that, if we seek for a resulting split induced subgraph that is covered by a fixed partition $(V_C, V_I) \in \mathcal{P}$, SPLITVD naturally reduces to a VERTEX COVER instance with the same parameter.

2 Small family of reasonable partitions: proof of Theorem 1.3

In this section we prove Theorem 1.3. To this end, we describe a branching algorithm that computes the family $\mathcal{P}$. The algorithm maintains a partition (called a *state*) of $V(G)$ into three parts V_C^0, V_I^0 and A; intuitively, the vertices of V_C^0 and V_I^0 are already assigned to V_C and V_I, whereas the set A consists of remaining (*active*) vertices. At each step, given a state $\mathcal{S} = (V_C^0, V_I^0, A)$, the algorithm outputs two partitions $(V_C^0 \cup A, V_I^0)$ and $(V_C^0, V_I^0 \cup A)$ and branches (calls itself recursively) into $2|A|$ subcases, creating two new states for each $v \in A$: a state $\mathcal{S}_{v \to C} = (V_C^0 \cup \{v\}, V_I^0 \cup (A \setminus N_G[v]), A \cap N_G(v))$ and a state $\mathcal{S}_{v \to I} = (V_C^0 \cup (A \cap N_G(v)), V_I^0 \cup \{v\}, A \setminus N_G[v])$. Informally speaking, in the first branch the vertex v is assigned to the clique part; consequently, all its non-neighbours are assigned to the independent set part, as they cannot be together with v in the clique part of a split induced subgraph of G. The second branch symmetrically assigns v to the independent set part and all neighbours of v to the clique part.

Moreover, the recurrence is trimmed at depth $2\lfloor \log n \rfloor + 1$. The algorithm is described on Pseudocode 1.

Function $\mathrm{Generator}(G, d, \mathcal{S} = (V_C^0, V_I^0, A))$ {$n = |V(G)|$ and $\mathcal{S} = (V_C^0, V_I^0, A)$ is a partition of $V(G)$}
1: output $(V_C^0 \cup A, V_I^0)$ and $(V_C^0, V_I^0 \cup A)$.
2: **if** $d < 2\lfloor \log n \rfloor + 1$ **then**
3: **for all** vertices $v \in A$ **do**
4: $\mathrm{Generator}(G, d+1, \mathcal{S}_{v \to C} = (V_C^0 \cup \{v\}, V_I^0 \cup (A \setminus N_G[v]), A \cap N_G(v)))$
5: $\mathrm{Generator}(G, d+1, \mathcal{S}_{v \to I} = (V_C^0 \cup (A \cap N_G(v)), V_I^0 \cup \{v\}, A \setminus N_G[v]))$
Function $\mathrm{GeneratePartitions}(G)$
6: $\mathrm{Generator}(G, 0, (\emptyset, \emptyset, V(G)))$.

Pseudocode 1: Algorithm that generates the family $\mathcal{P}$ from Theorem 1.3.

Since the algorithm trims the recurrence at depth $2\lfloor \log n \rfloor + 1$, the bounds on the running time and the size of the family $\mathcal{P}$ follow: at each step, $2|A| \leq 2n$ new subcases are created, the search tree contains at most $(2n)^{2\lfloor \log n \rfloor + 1}$ leaves and less than twice as much vertices, and each call to the procedure $\mathrm{Generator}$ outputs two partitions. To finish the proof of Theorem 1.3, we need to show the computed family $\mathcal{P}$ admits the first property of Theorem 1.3.

To this end, let us fix a set $X \subseteq V(G)$ that induces a split graph in G and a partition (X_C, X_I) of X such that $G[X_C]$ is a clique and $G[X_I]$ is an independent set. We show that the algorithm outputs a partition (V_C, V_I) with $X_C \subseteq V_C$ and $X_I \subseteq V_I$.

We say that a state $\mathcal{S} = (V_C^0, V_I^0, A)$ is *promising* if $X_C \subseteq V_C^0 \cup A$ and $X_I \subseteq V_I^0 \cup A$; note that this is a necessary condition to output a desired partition in any subcase generated from the state $\mathcal{S}$. Moreover, note that the initial state $(\emptyset, \emptyset, V(G))$ is clearly promising.

Consider a promising state $\mathcal{S} = (V_C^0, V_I^0, A)$. Denote $X_C^A = X_C \cap A$ and $X_I^A = X_I \cap A$. Note that if $X_C^A = \emptyset$, then the partition $(V_C^0, V_I^0 \cup A)$ is a desired partition. Symmetrically, if $X_I^A = \emptyset$, then the partition $(V_C^0 \cup A, V_I^0)$ is a desired partition; both these partitions are output by the algorithm.

Consider now the remaining case where X_C^A and X_I^A are nonempty. Note that for $v \in X_C^A$, the state $\mathcal{S}_{v \to C}$ is also promising, as $G[X_C]$ is a clique and $X_C \subseteq \{v\} \cup N_G(v)$. Symmetrically, for any $v \in X_I^A$, the state $\mathcal{S}_{v \to I}$ is also promising, as $G[X_I]$ is an independent set and $X_I \subseteq V(G) \setminus N_G(v)$. However, our recurrence is trimmed at depth $2\lfloor \log n \rfloor + 1$. To cope with this obstacle, we show that there exists a choice of $v \in A$ that efficiently reduces the sizes of X_C^A and X_I^A.

Let F be the set of edges of G that have one endpoint in X_C^A and second endpoint in X_I^A. If $|F| > |X_C^A| \cdot |X_I^A|/2$ (i.e., there are more edges between X_C^A and X_I^A than non-edges) then, by standard averaging argument, there exists a vertex $v \in X_I^A$ such that $|N_G(v) \cap X_C^A| > |X_C^A|/2$ (i.e., more than half of the vertices of X_C^A are neighbours of v) Otherwise, if $|F| \leq |X_C^A| \cdot |X_I^A|/2$, then there exists a vertex $v \in X_C^A$ such that $|X_I^A \setminus N_G(v)| \geq |X_I^A|/2$ (i.e., at least half of the vertices of X_I^A are not neighbours of v). In the first case, in the promising state $\mathcal{S}_{v \to I}$ the size of the set X_C^A is reduced by at least half; in the second case, in the promising state $\mathcal{S}_{v \to C}$ the size of the set X_I^A is reduced by at least half. At the beginning, $|X_C^A|, |X_I^A| \leq n$, thus the recurrence reaches a promising state where X_C^A or X_I^A is empty at depth at most $2\lfloor \log n \rfloor + 1$. This finishes the proof of Theorem 1.3.

3 The algorithm: proof of Theorem 1.1

Equipped with Theorem 1.3, we are now ready to show the proof of Theorem 1.1. Consider a SPLITVD instance (G, k). First, we invoke one of the known preprocessing (kernelization) routines for SPLITVD that reduces the number of vertices of the graph to a polynomial in k, without increasing the parameter. Here, we can either use the generic framework of the d-HITTING SET problem [1] (recall that the class of split graphs has a finite set of forbidden induced subgraphs) or use the recent $\mathcal{O}(k^3)$-vertex kernel by Ghosh et al. [8]. This step adds an additive factor of polynomial order in $|V(G)|$ both to time and space complexity of the algorithm.

Second, we invoke Theorem 1.3 and process the output partitions one by one. For a given partition (V_C, V_I), we seek for a set $X \subseteq V(G)$, such that $G[V_C \cap X]$ is a clique, $G[V_I \cap X]$ is an independent set and $|V(G) \setminus X| \leq k$. By Theorem 1.3 this is sufficient to solve the initial SPLITVD instance (G, k), and this step adds an $k^{\mathcal{O}(\log k)}$ multiplicative factor to the time complexity and a polynomial in k additive factor to the space complexity.

Fix a partition (V_C, V_I). We are to delete at most k vertices from the graph G to make $G[V_C]$ a clique and $G[V_I]$ an independent set. Let G' be defined as a disjoint union of $G[V_I]$ and a complement of $G[V_C]$. Note that our task becomes the classical vertex cover problem in the graph G' with parameter k: we need to cover all edges of $G[V_I]$ and non-edges of $G[V_C]$. Therefore, for a fixed partition (V_C, V_I), the problem can be solved in the same time as the VERTEX COVER problem for a graph of the same size and parameter k. This finishes the proof of Theorem 1.1.

4 Conclusions

We have shown that the dependencies on the parameter k in the optimal time complexity of fixed-parameter algorithms for VERTEX COVER and SPLIT VERTEX DELETION are essentially equal. This result can be considered as a tight bound on the time complexity of fixed-parameter algorithms for SPLIT VERTEX DELETION.

However, note that our reduction adds a polynomial in the size of the input graph additive factor to the time complexity that results from the application of a kernelization algorithm. The algorithm of Chen et al. [3] for the VERTEX COVER problem has linear dependency on n. We leave as an open problem to obtain a linear-time polynomial kernel for SPLITVD; such a result would automatically yield a linear-time dependency on n in our algorithm.

References

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

[2] Leizhen Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Inf. Process. Lett.*, 58(4):171–176, 1996.

[3] Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theor. Comput. Sci.*, 411(40-42):3736–3756, 2010.

[4] Reinhard Diestel. *Graph Theory*. Springer, 2005.

[5] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Springer, 1999.

[6] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. 2006.

[7] S. Foldes and P. Hammer. Split graphs. *Congressus Numerantium*, 19:311–315, 1977.

[8] Esha Ghosh, Sudeshna Kolay, Mrinal Kumar, Pranabendu Misra, Fahad Panolan, Ashutosh Rai, and M. S. Ramanujan. Faster parameterized algorithms for deletion to split graphs. In Fedor V. Fomin and Petteri Kaski, editors, *SWAT*, volume 7357 of *Lecture Notes in Computer Science*, pages 107–118. Springer, 2012.

[9] John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, 20(2):219–230, 1980.

[10] Daniel Lokshtanov, N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Faster parameterized algorithms using linear programming. *CoRR*, abs/1203.0833, 2012.

[11] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*, volume 31 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press, 2006.