Parameterized Algorithms on Perfect Graphs for Deletion to (r, ℓ)-Graphs*

Sudeshna Kolay[1], Fahad Panolan[2], Venkatesh Raman[3], and Saket Saurabh[4]

1 **Institute of Mathematical Sciences, Chennai, India**
2 **Department of Informatics, University of Bergen, Norway**
3 **Institute of Mathematical Sciences, Chennai, India**
4 **Institute of Mathematical Sciences, Chennai, India; and Department of Informatics, University of Bergen, Norway**

Abstract

For fixed integers $r, \ell \geq 0$, a graph G is called an (r, ℓ)-*graph* if the vertex set $V(G)$ can be partitioned into r independent sets and ℓ cliques. Such a graph is also said to have *cochromatic number* $r + \ell$. The class of (r, ℓ) graphs generalizes r-colourable graphs (when $\ell = 0$) and hence not surprisingly, determining whether a given graph is an (r, ℓ)-graph is NP-hard even when $r \geq 3$ or $\ell \geq 3$ in general graphs.

When r and ℓ are part of the input, then the recognition problem is NP-hard even if the input graph is a perfect graph (where the Chromatic Number problem is solvable in polynomial time). It is also known to be fixed-parameter tractable (FPT) on perfect graphs when parameterized by r and ℓ. I.e. there is an $f(r+\ell) \cdot n^{\mathcal{O}(1)}$ algorithm on perfect graphs on n vertices where f is a function of r and ℓ. Observe that such an algorithm is unlikely on general graphs as the problem is NP-hard even for constant r and ℓ.

In this paper, we consider the parameterized complexity of the following problem, which we call Vertex Partization. Given a perfect graph G and positive integers r, ℓ, k decide whether there exists a set $S \subseteq V(G)$ of size at most k such that the deletion of S from G results in an (r, ℓ)-graph. This problem generalizes well studied problems such as Vertex Cover (when $r = 1$ and $\ell = 0$), Odd Cycle Transversal (when $r = 2$, $\ell = 0$) and Split Vertex Deletion (when $r = 1 = \ell$).

1. Vertex Partization on perfect graphs is FPT when parameterized by $k + r + \ell$.
2. The problem, when parameterized by $k + r + \ell$, does not admit any polynomial sized kernel, under standard complexity theoretic assumptions. In other words, in polynomial time, the input graph cannot be compressed to an equivalent instance of size polynomial in $k + r + \ell$. In fact, our result holds even when $k = 0$.
3. When r, ℓ are universal constants, then Vertex Partization on perfect graphs, parameterized by k, has a polynomial sized kernel.

1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases Graph deletion, FPT algorithms, Polynomial kernels

Digital Object Identifier 10.4230/LIPIcs.MFCS.2016.75

* The research leading to these results has received funding from the European Research Council (ERC) via grants Rigorous Theory of Preprocessing, reference 267959 and PARAPPROX, reference 306992.

1 Introduction

For fixed integers $r, \ell \geq 0$, a graph G is called an (r, ℓ)-*graph* if the vertex set $V(G)$ can be partitioned into r independent sets and ℓ cliques. Many special subclasses of this graph class, such as bipartite, chordal, interval, split and permutation, are well studied in various areas of algorithm design and intractability. For example, $(2, 0)$- and $(1, 1)$-graphs correspond to bipartite graphs and split graphs respectively. A $(3, 0)$-graph is a 3-colourable graph. Hence, there is a rich dichotomy even with respect to recognition algorithms for (r, ℓ)-graphs. It is well known that we can recognize $(2, 0)$- and $(1, 1)$-graphs, on n vertices and m edges, in $\mathcal{O}(m + n)$ time. In fact, one can show that recognizing whether a graph G is an (r, ℓ)-graph, when $r, \ell \leq 2$, can be done in polynomial time [4, 9]. On the other hand, when either $r \geq 3$ or $\ell \geq 3$, the recognition problem is as hard as the celebrated 3-Colouring problem, which is NP-complete [12]. Thus, the following problem is NP-hard when r or ℓ are at least 3:

Partization Recognition
Input: A graph G and positive integers r, ℓ
Question: Is G an (r, ℓ)-graph?

Partization Recognition has also been studied when the input is restricted to be a chordal graph. This restricted problem has a polynomial time algorithm [10]. On the other hand, when the input graphs are restricted to perfect graphs, Partization Recognition is NP-complete [23]. It was shown in [15], that the problem, when parameterized by $r + \ell$, has an FPT algorithm, i.e, an algorithm that runs in time $f(r + \ell) \cdot n^{\mathcal{O}(1)}$ for a computable function f. A natural extension to Partization Recognition is the Vertex Partization problem. The problem is formally stated below:

Vertex Partization	**Parameter:** r, ℓ, k
Input: A graph G and positive integers r, ℓ, k	
Question: Is there a vertex subset $S \subseteq V(G)$ of size at most k such that $G - S$ is an (r, ℓ)-graph?	

Because of the NP-hardness of the 3-Colouring problem, we do not expect to obtain an FPT algorithm on general graphs, parameterized by $k + r + \ell$, for Vertex Partization, when $r \geq 3$ or $\ell \geq 3$. It has been shown in [16, 2] that, for all other combinations of r and ℓ, namely when $0 \leq r, \ell \leq 2$, Vertex Partization has an FPT algorithm with running time $3.3146^k n^{\mathcal{O}(1)}$. Various special cases of this problem are very well studied. When $r = 1$ and $\ell = 0$, the problem is the same as the celebrated Vertex Cover problem, which has been extensively studied in parameterized complexity, and the current fastest algorithm runs in time $1.2738^k n^{\mathcal{O}(1)}$ and has a kernel with $2k$ vertices [5] (A brief overview of paramaterized complexity is given in the preliminaries).

For $r = 2$ and $\ell = 0$, the problem is the same as Odd Cycle Transversal (OCT) whose parameterized complexity was settled by Reed et al. [22] by using the iterative compression technique (which is also the technique we use to show that Vertex Partization on perfect graphs has an FPT algorithm in this paper) for the first time. The current best algorithm for the problem is by Lokshtanov et al. [20] with a running time of $2.3146^k n^{\mathcal{O}(1)}$ that uses a branching algorithm based on linear programming.

When $r = \ell = 1$, the Vertex Partization problem is the same as Split Vertex Deletion (SVD) for which Ghosh et al. [13] designed an algorithm with running time $2^k n^{\mathcal{O}(1)}$. They also gave the best known polynomial kernel for SVD. Later, Cygan and Pilipczuk [7] designed an algorithm for SVD running in time $1.2738^{k+o(k)} n^{\mathcal{O}(1)}$.

As in the case of Partization Recognition, the Vertex Partization problem has also been studied when the input graph is restricted to a non-trivial graph class. By a celebrated result of Lewis and Yannakakis [19], this problem is NP-complete even when restricted to the perfect graph class. In fact, Odd Cycle Transversal (OCT) restricted to perfect graphs is NP-hard, because of this result. Thus, we cannot expect an FPT algorithm for Vertex Partization parameterized by $r + \ell$, unless P = NP. Moreover, because of the NP-hardness of Partization Recognition on perfect graphs, we do not expect Vertex Partization on perfect graphs to be FPT when parameterized by k alone, again under the assumption that P $\neq$ NP. Krithika and Narayanaswamy [17] studied Vertex Partization problems on perfect graphs, and among several results, they obtain an $(r+1)^k n^{\mathcal{O}(1)}$ algorithm for Vertex $(r, 0)$-Partization on perfect graphs. In this paper, we generalize this for all values of r and ℓ. In other words, we show that Vertex Partization on perfect graphs, parameterized by $k + r + \ell$, is FPT.

Our Results and Methods. For Vertex Partization on perfect graphs, parameterized by $k + r + \ell$, we give an FPT algorithm using the method of iterative compression. This algorithm is inspired by the FPT algorithm for Cochromatic Number on Perfect Graphs, given in [15].

Then, we obtain a negative result for *kernelization* for Vertex Partization on perfect graphs. We show that Vertex Partization cannot have a polynomial kernel unless NP $\subseteq$ co-NP/poly. This is shown by exhibiting a polynomial parameter transformation from CNF-SAT. In fact, our result holds even when $k = 0$ and either r or ℓ is one. Thus, we show that Partization Recognition, parameterized by r, ℓ (also known as the Cochromatic Number problem [15]), does not admit a polynomial kernel on perfect graphs unless NP $\subseteq$ co-NP/poly. See Section 2 for the definition of polynomial parameter transformation and kernelization.

Finally, we consider the following parameterized problem:

Vertex (r, ℓ) Partization	**Parameter:** k
Input: A graph G and a positive integer k	
Question: Is there a vertex subset $S \subseteq V(G)$ of size at most k such that $G - S$ is an (r, ℓ)-graph?	

For each pair of constants r and ℓ, we give a polynomial kernelization algorithm for the above parameterized problem. To arrive at the kernelization algorithm, we consider a slightly larger class of graphs called (r, ℓ)-*split graphs*. A graph G is an (r, ℓ)-split graph if its vertex set can be partitioned into V_1 and V_2, such that the size of a largest clique in $G[V_1]$ is bounded by r and the size of a largest independent set in $G[V_2]$ is bounded by ℓ. Such a two-partition for the graph G is called as (r, ℓ)-split partition. The notion of (r, ℓ)-split graphs was introduced in [14]. For any fixed r and ℓ, there is a finite forbidden set $\mathbb{F}_{r,\ell}$ of graphs for (r, ℓ)-split graphs [14]. That is, a graph G is an (r, ℓ)-split graph if and only if G does not contain any graph $H \in \mathbb{F}_{r,\ell}$ as an induced subgraph. The size of the largest forbidden graph is bounded by $f(r, \ell)$, for some function f depending only on r and ℓ [14]. We use this to design the kernelization algorithm.

2 Preliminaries

We use $[n]$ to denote $\{1, \ldots, n\}$. We use standard notations from graph theory [8]. The vertex set and edge set of a graph are denoted as $V(G)$ and $E(G)$ respectively. The complement

of the graph G, denoted by $\overline{G}$, has $V(G)$ as its vertex set and $\binom{V(G)}{2} \setminus E(G)$ as its edge set. Here, $\binom{V(G)}{2}$ denotes the family of two sized subsets of $V(G)$. The neighbourhood of a vertex v is represented as $N_G(v)$, or, when the context of the graph is clear, simply as $N(v)$. An induced subgraph of G on the vertex set $V' \subseteq V$ is written as $G[V']$. For a vertex subset $V' \subseteq V$, $G[V \setminus V']$ is also denoted as $G - V'$. We denote by $\omega(G)$ the size of a maximum clique in G. Similarly, $\alpha(G)$ denotes the size of a maximum independent set in G. The chromatic number of G is denoted by $\chi(G)$. In this paper, we consider the class of (r, ℓ)-graphs, The following is the formal definition of this graph class.

▶ **Definition 1** ((r, ℓ)-graph)**.** A graph G is an (r, ℓ)-graph if its vertex set can be partitioned into r independent sets and ℓ cliques. We call such a partition of $V(G)$ an (r, ℓ)-partition.

For a graph G, we say $S \subseteq V(G)$ is an (r, ℓ)-*vertex deletion set*, if $G - S$ is an (r, ℓ)-graph.

▶ **Definition 2** (IC-partition)**.** An IC-partition, of an (r, ℓ)-graph G, is a partition (V_1, V_2) of $V(G)$ such that $G[V_1]$ can be partitioned into r independent sets and $G[V_2]$ can be partitioned into ℓ cliques.

A graph G is a *perfect graph* if, for every induced subgraph H, $\chi(H) = \omega(H)$. We also need the following characterisation of perfect graphs – also known as strong perfect graph theorem.

▶ **Proposition 3** ([6])**.** *A graph G is perfect if and only if G does not have, as an induced subgraph, an odd cycle of length at least* 5 *or its complement.*

This tells us that perfect graphs are closed under complementation. However, this was proved earlier and was called weak perfect graph theorem.

▶ **Lemma 4** ([21])**.** *G is a perfect graph if and only if $\overline{G}$ is a perfect graph.*

Moreover, this class is well known for its tractability for several NP-hard problems.

▶ **Proposition 5** ([15, Lemma 3])**.** *Given a perfect graph G and an integer ℓ, there is a polynomial time algorithm to output*
(a) *either a partition of $V(G)$ into at most ℓ independent sets or a clique of size $\ell + 1$, and*
(b) *either a partition of $V(G)$ into at most ℓ cliques or an independent set of size $\ell + 1$.*

Parameterized Complexity. A parameterized problem Π is a subset of $\Gamma^* \times \mathbb{N}$, where Γ is a finite alphabet. An instance of a parameterized problem is a tuple (x, k), where x is a classical problem instance, and k is called the parameter. A central notion in parameterized complexity is *Fixed Parameter Tractability (FPT)*. A parameterized problem Π is in FPT if there is an algorithm that takes an instance (x, k) and decides if $(x, k) \in \Pi$ in time $f(k) \cdot |x|^{\mathcal{O}(1)}$. Here, f is an arbitrary function of k. Such an algorithm is called a *Fixed Parameter Tractable* algorithm and, in short, an FPT algorithm.

Kernelization. A kernelization algorithm for a parameterized problem $\Pi \subseteq \Gamma^* \times \mathbb{N}$ is an algorithm that, given $(x, k) \in \Gamma^* \times \mathbb{N}$, outputs, in time polynomial in $|x| + k$, a pair $(x', k') \in \Gamma^* \times \mathbb{N}$ such that (a) $(x, k) \in \Pi$ if and only if $(x', k') \in \Pi$ and (b) $|x'|, k' \leq g(k)$, where g is some computable function. The output instance x' is called the kernel, and the function g is referred to as the size of the kernel. If $g(k) = k^{\mathcal{O}(1)}$ then we say that Π has a polynomial kernel.

Lower bounds in Kernelization. In recent years, several techniques have been developed to show that certain parameterized problems cannot have any polynomial sized kernel unless some classical complexity assumptions are violated. One such technique is the *polynomial parameter transformation*.

▶ **Definition 6.** Let Π, Γ be two parameterized problems. A polynomial time algorithm $\mathcal{A}$ is called a polynomial parameter transformation (or ppt) from Π to Γ if , given an instance (x, k) of Π, $\mathcal{A}$ outputs in polynomial time an instance (x', k') of Γ such that $(x, k) \in \Pi$ if and only if $(x', k') \in \Gamma$ and $k' \leq k^{\mathcal{O}(1)}$.

We use the following theorem together with ppt reductions to rule out polynomial kernels.

▶ **Proposition 7** ([3]). *Let Π, Γ be two parameterized problems such that Π is* NP*-hard, $\Gamma \in$ NP and there exists a polynomial parameter transformation from Π to Γ. Then, if Π does not admit a polynomial kernel neither does Γ.*

▶ **Proposition 8** ([11]). CNF-SAT *is* FPT *parameterized by the number of variables; however, it does not admit a polynomial kernel unless* NP $\subseteq$ co-NP*/poly.*

3 FPT algorithm for Vertex Partization

In this section, we show that Vertex Partization on perfect graphs is FPT, using the iterative compression technique. Let (G, r, ℓ, k) be an input instance of Vertex Partization on perfect graphs, and let $V(G) = \{v_1, \ldots, v_n\}$. We define, for every $1 \leq i \leq n$, the vertex set $V_i = \{v_1, \ldots, v_i\}$. Let G_i denote $G[V_i]$. Let $i_0 = r + \ell + k + 1$. We iterate through the instances (G_i, r, ℓ, k) starting from $i = i_0$. Given the i^{th} instance and a known (r, ℓ)-vertex deletion set S'_i of size at most $k + 1$, our objective is to obtain an (r, ℓ)-vertex deletion set S_i of size at most k. The formal definition of this compression problem is as follows.

Vertex Partization Compression	**Parameter:** r, ℓ, k
Input: A perfect graph G, non-negative integers r, ℓ, k and a $k+1$-sized vertex subset S' such that $G - S'$ is an (r, ℓ)-graph, along with an IC-partition (Q_1, Q_2) of $G - S'$.	
Output: A vertex subset $S \subseteq V(G)$ of size at most k such that $G - S$ is an (r, ℓ)-graph, and an IC-partition (P_1, P_2) of $G - S$.	

Before we solve Vertex Partization Compression, we explain how to reduce the Vertex Partization problem to $n-(r+\ell+k+1)+1$ instances of the Vertex Partization Compression problem on G_i, from $i = i_0$ to $i = n$. For the graph G_{i_0}, the set V_{k+1} is a (r, ℓ)-vertex deletion set, S'_i, of size $k + 1$. The graph $G_{i_0} - V_{k+1}$ has $r + \ell$ vertices. We set $Q_1^{i_0}$ to be a set of any r vertices of $V_{i_0} - V_{k+1}$ and $Q_2^{i_0}$ to be the remaining set of ℓ vertices; that is, $Q_2^{i_0} = V_{i_0} - V_{k+1} - Q_1^{i_0}$. Now, let $I_i = (G_i, r, \ell, k, S'_i, (Q_1^i, Q_2^i))$ be the i^{th} instance of Vertex Partization Compression. If S_{i-1} is a k-sized solution for I_{i-1}, then $S_{i-1} \cup \{v_i\}$ is a $(k + 1)$-sized (r, ℓ)-vertex deletion set for G_i. So, the iteration begins with the instance $I_{i_0} = (G_{i_0}, r, \ell, k, V_{k+1}, (Q_1^{i_0}, Q_2^{i_0}))$ and we try to obtain an (r, ℓ)-vertex deletion set of size at most k. If such a solution S_{i_0} exists, we set $S'_{i_0+1} = S_{i_0} \cup \{v_{i_0+1}\}$ and ask for a k-sized solution for the instance I_{i_0+1}. We continue in this manner. If, during any iteration, the corresponding instance does not have an (r, ℓ)-vertex deletion set of size at most k, then this implies that the original instance (G, r, ℓ, k) is a NO instance for Vertex Partization. If S_n is a k-sized (r, ℓ)-vertex deletion set for G_n, where $G_n = G$, then clearly (G, r, ℓ, k) is YES instance of Vertex Partization. Since there are at most $n - (r + \ell + k + 1) + 1$ iterations, the total time taken by the algorithm to solve Vertex Partization is at most

$n - (r + \ell + k + 1) + 1$ times the time taken to solve Vertex Partization Compression. Thus, if Vertex Partization Compression is FPT, it follows that Vertex Partization is FPT.

We can also *view* the input graph G of an instance of Vertex Partization Compression as an $(r+k+1, \ell)$-graph with IC-partition $(Q_1 \cup S', Q_2)$. Equivalently, Vertex Partization Compression has an input positive integers r, ℓ, k and a graph G that is an $(r + k + 1, \ell)$-graph and the objective is to decide whether there is a k-sized set $S \subseteq V(G)$ such that $G - S$ is an (r, ℓ)-graph. This view point allows us to design an FPT algorithm for Vertex Partization Compression. Towards that we first define some notations. Let G be a graph and $V(G) = \{v_1, \ldots, v_n\}$. For partition $P = (A, B)$ of $V(G)$ we define an n-length bit vector B_P^G corresponding to P as follows. We set the i^{th} bit to 0 if $v_i \in A$ and to 1 otherwise. For two n-length bit vectors $a = a_1 \ldots a_n$ and $b = b_1 \ldots b_n$, the Hamming distance between a and b, denoted by $\mathcal{H}(a, b)$, is the number of indices on which a and b mismatches. That is, $\mathcal{H}(a, b) = |\{(a_i, b_i) \ | a_i \neq b_i, i \in [n] \ \}|$. The Hamming distance for the bitvectors corresponding to two IC-partitions, of a graph, is bounded as given by the following proposition.

▶ **Proposition 9** ([15]). *Let G be a graph. Let Q be an IC-partition of G that realizes G as an (r', ℓ')-graph and P is an IC-partition of G that realizes G as an (r, ℓ)-graph. Then $\mathcal{H}(B_Q^G, B_P^G) \leq r'\ell + r\ell'$.*

The following lemma follows from Proposition 9.

▶ **Lemma 10.** *Let G be a perfect graph and (Q_1, Q_2) be an IC-partition that realizes G as an (r', ℓ')-graph. Let S be an (r, ℓ)-vertex deletion set for G such that P is an IC-partition of G that realizes $G - S$ as an (r, ℓ)-graph and let $Q = (Q_1 \setminus S, Q_2 \setminus S)$. Then $\mathcal{H}(B_Q^{G-S}, B_P^{G-S}) \leq r'\ell + r\ell'$.*

Lemma 10 implies that to solve Vertex Partization Compression it is enough to solve the following problem.

Short Vertex Partization	**Parameter:** r, ℓ, k, ρ
Input: A perfect graph G, positive integers r, ℓ, k, ρ, and a partition $Q = (Q_1, Q_2)$ of $V(G)$.	
Output: A vertex subset $S \subseteq V(G)$ of size at most k such that $G - S$ is a (r, ℓ)-graph, and an IC-partition (P_1, P_2) of $G - S$ such that $\mathcal{H}(B_P^{G-S}, B_{Q'}^{G-S}) \leq \rho$ where $Q' = (Q_1 \setminus S, Q_2 \setminus S)$.	

▶ **Lemma 11.** Short Vertex Partization *is* FPT.

Proof. We design a recursive algorithm $\mathcal{A}$ for Short Vertex Partization which takes a tuple $(G, r, \ell, k, \rho, Q = (Q_1, Q_2))$ as input, where G is a graph, Q is a partition of $V(G)$ and r, ℓ, k, ρ are integers. It outputs a k-sized (r, ℓ)-vertex deletion set S of G and an IC-partition P that realizes that $G - S$ is (r, ℓ)-graph such that $\mathcal{H}(B_P^{G-S}, B_{Q'}^{G-S}) \leq \rho$, where $Q' = (Q_1 \setminus S, Q_2 \setminus S)$, if such a tuple (S, P) exists, otherwise returns NO. The following are the steps of the recursive algorithm $\mathcal{A}$ on input $(G, r, \ell, k, Q = (Q_1, Q_2), \rho)$.

1. If $k < 0$ or $\rho < 0$ then output NO.
2. If $G[Q_1]$ is r-colorable and $\overline{G}[Q_2]$ is ℓ-colorable, then return $(\emptyset, Q)$.
3. If $G[Q_1]$ is not r-colorable, then there is an $r + 1$-sized clique in $G[Q_1]$. By Proposition 5, we can find such a $r + 1$-sized clique C in polynomial time. Make the following recursive calls to $\mathcal{A}$:

(a) For every vertex $v \in V(C)$, do a recursive call $\mathcal{A}(G-v, r, \ell, k-1, \rho, (Q_1 \setminus \{v\}, Q_2 \setminus \{v\}))$ and if the recursive call returns (S', P) then return $(S' \cup \{v\}, P)$ as output.

(b) For every vertex $v \in V(C)$, do a recursive call $\mathcal{A}(G, r, \ell, k, \rho-1, (Q_1 \setminus \{v\}, Q_2 \cup \{v\}))$ and if the recursive call returns (S', P) then return (S', P) as output.

If all the recursive calls above (in Step 3) return NO, then return NO.

4 If $\overline{G}[Q_2]$ is not ℓ-colorable, then there is clique of size $\ell+1$ in $\overline{G}[Q_2]$. Thus, using Proposition 5, we can find an $\ell+1$-sized independent set I in $G[Q_2]$. Make the following recursive calls of the algorithm:

(a) For every vertex $v \in V(I)$, do a recursive call $\mathcal{A}(G-v, r, \ell, k-1, \rho, (Q_1 \setminus \{v\}, Q_2 \setminus \{v\}))$ and if the recursive call returns (S', P) then return $(S' \cup \{v\}, P)$ as output.

(b) For every vertex $v \in V(I)$, do a recursive call $\mathcal{A}(G, r, \ell, k, \rho-1, (Q_1 \cup \{v\}, Q_2 \setminus \{v\}))$ and if the recursive call returns (S', P) then return (S', P) as output.

If all the recursive calls above (in Step 4) return NO, then return NO.

We now prove that the recursive algorithm $\mathcal{A}$ is correct. We show that if $(G, r, \ell, k, \rho, (Q_1, Q_2))$ is a YES instance of Short Vertex Partization, then the algorithm $\mathcal{A}$ will output correct solution. We prove this by induction on $k+\rho$.

1. Base case: $k=0$ and $\rho=0$. Since $(G, r, \ell, k, \rho, (Q_1, Q_2))$ is a YES instance, (Q_1, Q_2) is an IC-partition which realizes that G is an (r, ℓ)-graph. In Step 2 of the algorithm $\mathcal{A}$ we output $(\emptyset, (Q_1, Q_2))$ as the output.
2. By induction hypothesis we assume that $\mathcal{A}$ outputs the correct answer when $k+\rho < \gamma$, where $\gamma \geq 0$. Now we need to show that $\mathcal{A}$ outputs correct answer when $k+\rho = \gamma$. Let $(G, r, \ell, k, \rho, Q = (Q_1, Q_2))$ be the input of $\mathcal{A}$ such that $k+\rho = \gamma$. Let $(S, (P_1, P_2))$ be a solution of Short Vertex Partization. If $S = \emptyset$ and $(P_1, P_2) = (Q_1, Q_2)$ then in Step 2, algorithm $\mathcal{A}$ will output $(\emptyset, Q)$. Otherwise either $G[Q_1]$ is not r-colorable or $\overline{G}[Q_2]$ is not ℓ-colorable.
3. Case 1: Suppose $G[Q_1]$ is not r-colorable. Then there is a clique C of size $r+1$ in $G[Q_1]$. In this case at least one vertex v from C either belongs to S or does not belong to P_1. If $v \in S$, then consider the recursive call $\mathcal{A}(G-v, r, \ell, k-1, \rho, (Q_1 \setminus \{v\}, Q_2 \setminus \{v\})$. By induction hypothesis $\mathcal{A}(G-v, r, \ell, k-1, \rho, (Q_1 \setminus \{v\}, Q_2 \setminus \{v\})$ will return (S', P') such that S' is a $k-1$ sized (r, ℓ)-vertex deletion set of $G-\{v\}$ such that $\mathcal{H}(B_{P'}^{G-S'}, B_{Q'}^{G-S'}) \leq \rho$, where $Q' = (Q_1 \setminus (S' \cup \{v\}), Q_2 \setminus (S' \cup \{v\}))$. Hence in Step 3$(a)$, algorithm $\mathcal{A}$ will output $(S' \cup \{v\}, P)$ and this is a solution for Short Vertex Partization on input $(G, r, \ell, k, \rho, (Q_1, Q_2))$.
   If $v \notin S$, then $v \notin P_1$ as well. Now consider the recursive call $\mathcal{A}(G, r, \ell, k, \rho-1, (Q_1 \setminus \{v\}, Q_2 \cup \{v\})$. By induction hypothesis $\mathcal{A}(G, r, \ell, k, \rho-1, (Q_1 \setminus \{v\}, Q_2 \cup \{v\})$ will return (S'', P'') such that S'' is a k sized (r, ℓ)-vertex deletion set of G such that $\mathcal{H}(B_{P''}^{G-S''}, B_{Q'}^{G-S''}) \leq \rho-1$, where $Q' = ((Q_1 \setminus \{v\}) \setminus S'', (Q_2 \cup \{v\}) \setminus S'')$. Hence in Step 3$(b)$, algorithm $\mathcal{A}$ will output (S'', P''). Clearly S'' is a k sized (r, ℓ)-vertex deletion set of G''. Now we show that $\mathcal{H}(B_{P''}^{G-S''}, B_{Q''}^{G-S''}) \leq \rho$, where $Q'' = (Q_1 \setminus S'', Q_2 \setminus S'')$. Note that $\mathcal{H}(B_{Q'}^{G-S''}, B_{Q''}^{G-S''}) \leq 1$. Since $\mathcal{H}(B_{P''}^{G-S''}, B_{Q'}^{G-S''}) \leq \rho-1$ and $\mathcal{H}(B_{Q'}^{G-S''}, B_{Q''}^{G-S''}) \leq 1$, we have that
   $\mathcal{H}(B_{P''}^{G-S''}, B_{Q''}^{G-S''}) \leq \rho$
4. Case 2: $\overline{G}[Q_2]$ is not ℓ-colorable. This case is symmetric to Case 1.

In the reverse direction we need to show that if the algorithm $\mathcal{A}$ returns YES then indeed the given instance is a YES instance. The proof of reverse direction is similar to the proof of forward direction. Again, we induct on $k+\rho$. The algorithm returns a bipartition as

evidence of a YES instance. We show that this bipartition is the required IC-partition. Such a bipartition is returned in Steps 2, $3(a), (b)$ and $4(a), (b)$. By description of the algorithm, both $k, \rho \geq 0$ in order for the algorithm to return YES. In the base case, $k = \rho = 0$. Here, it must be the case that Step 2 is executed and the output bipartition is Q itself, while the output vertex deletion set is $\emptyset$. In this case, by definition Q is an IC-partition. Hence, Q is evidence that the input graph G is already an (r, ℓ)-graph and does not require any vertex to be deleted. Thus, in the base case, we correctly determine that the given input instance is a YES instance.

By induction hypothesis we assume that $\mathcal{A}$ outputs the correct answer when $k + \rho < \gamma$, where $\gamma \geq 0$. Now we need to show that if $\mathcal{A}$ outputs a bipartition P and a vertex set S when $k + \rho = \gamma$, then the vertex set S is an (r, ℓ)-vertex deletion set while the bipartition is an IC-partition of $G - S$. If the bipartition is output in Step 2, then by definition the input graph is already an (r, ℓ)-graph. Otherwise, a recursive call is made to an instance where $k + \rho$ is strictly smaller. By induction hypothesis, the recursive call returns an (r, ℓ)-vertex deletion set S' and a witnessing IC-partition of $G - S'$. It follows that the vertex set S and the bipartition output by the algorithm, on the current input instance, is an (r, ℓ)-vertex deletion set of size at most k, and an IC-partition respectively for the input instance. This completes the proof of correctness of the algorithm.

What remains is the running time analysis. Note that when $k < 0$ or $\rho < 0$, then the algorithm will stop in a single step. Each recursive call either decreases k by 1 or ρ by 1. Hence the depth of the recursion tree is bounded by $k + \rho + 1$. Note that in Step 3 we make at most $2(r + 1)$ recursive calls and in Step 4 we make at most $2(\ell + 1)$ recursive calls. Hence the total running time of the algorithm $\mathcal{A}$ is bounded by $\mathcal{O}(\max\{(2(r + 1))^{k+\rho+1} n^{\mathcal{O}(1)}, (2(\ell + 1))^{k+\rho+1} n^{\mathcal{O}(1)}\})$. ◀

Vertex Partization Compression is a special case of Short Vertex Partization when $\rho = (r + k + 1)\ell + r\ell$. Therefore, we have the following theorem.

▶ **Theorem 12.** Vertex Partization *on perfect graphs has an* FPT *algorithm with running time* $2^{\mathcal{O}((k+r)\ell \log(r+\ell))} n^{\mathcal{O}(1)}$.

4 Kernel lower bound for Vertex Partization

In this section, we show that Vertex Partization on perfect graphs does not have polynomial kernels. Partization Recognition on perfect graphs, when parameterized by $r + \ell$, was shown to be FPT in [15]. Our proof implies that Partization Recognition on perfect graphs cannot have a polynomial kernel, when parameterized by $r + \ell$, unless NP $\subseteq$ co-NP/poly.

▶ **Theorem 13.** Partization Recognition *on perfect graphs, when parameterized by* $r + \ell$, *does not have a polynomial kernel unless* NP $\subseteq$ co-NP*/poly.*

Proof. We prove the theorem by giving a polynomial parameter transformation from CNF-SAT parameterized by the number of variables. From Proposition 8, we know that CNF-SAT, parameterized by the number of variables, does not have a polynomial kernel unless NP $\subseteq$ co-NP/poly [11]. Then the proof of the theorem follows from Proposition 7. We start with an instance (ϕ, n) of CNF-SAT, where ϕ is a CNF formula with m clauses and n variables. Without loss of generality, we assume that there is no clause where both literals of a variable appear together, since such a clause will be satisfied by any assignment and hence can be removed. The polynomial parameter transformation produces an instance $(G, n, 1)$ of

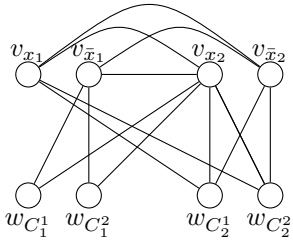

Figure 1 An illustration of the construction of the graph G in Theorem 13 for the formula $\phi = (x_1 \vee \overline{x}_2) \wedge (\overline{x}_1)$. Here $C_1 = (x_1 \vee \overline{x}_2)$ and $C_2 = (\overline{x}_1)$.

Partization Recognition, where G is a perfect graph, such that (ϕ, n) is a YES instance of CNF-SAT if and only if $(G, n, 1)$ is a YES instance of Partization Recognition. Let $\mathcal{C} = \{C_1, \ldots, C_m\}$, $X = \{x_1, \ldots, x_n\}$ and $L = \{x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n\}$ be the set of clauses, variables and literals of ϕ respectively. The construction of the graph G from the formula ϕ is as follows (illustrated in Figure 1):

1. For each variable x, we create two vertices $v_x, v_{\bar{x}}$ which represent the literals $x, \bar{x}$. We call them the literal vertices. More specifically, we call v_x the positive literal vertex and $v_{\bar{x}}$ the negative literal vertex.
2. For each clause C, we create two vertices w^1_C, w^2_C. We call these the clause vertices corresponding to the clause C.
3. For each pair of variables x, y, we add the edges $(v_x, v_y), (v_{\bar{x}}, v_y), (v_x, v_{\bar{y}})$, and $(v_{\bar{x}}, v_{\bar{y}})$. Notice that $(v_x, v_{\bar{x}})$ and $(v_y, v_{\bar{y}})$ are non-edges.
4. For each clause C and each literal $q \in L$, if $q \notin C$, we add edges (x_q, w^1_C) and (x_q, w^2_C). In other words if a literal q' belongs to a clause C, then $(q', w^1_C), (q', w^2_C) \notin E(G)$ So, there is a complete bipartite graph between $L \setminus C$ and $\{w^1_C, w^2_C\}$.

In short, the vertex set and edge set of G is defined as follows (note that for a literal x, if $x = \bar{y}$, then $y = \bar{x}$).

$$\begin{aligned} V(G) &= \{v_x, v_{\bar{x}} \mid x \in X\} \cup \{w^1_C, w^2_C \mid C \in \mathcal{C}\} \\ E(G) &= \{(v_x, v_y) \mid x, y \in L, x \neq \bar{y}\} \cup \{(w^1_C, v_x), (w^2_C, v_x) \mid x \in L, x \notin C, C \in \mathcal{C}\} \end{aligned}$$

Let $V_X = \{v_x, v_{\bar{x}} \mid x \in X\}$ and $V_{\mathcal{C}} = \{w^1_C, w^2_C \mid C \in \mathcal{C}\}$. Note that the set of vertices $V_{\mathcal{C}}$ corresponding to the clauses forms an independent set in G. First we show that G is a perfect graph.

▶ **Claim 14.** *The graph $\overline{G}$ does not contain an induced odd cycle of length ≥ 5.*

Proof. We first prove that there is no path of length 2 (path on 3 vertices) in $\overline{G}[V_X]$. Note that $E(\overline{G}[V_X]) = \{(v_x, v_{\bar{x}}) \mid x \in X\}$. This implies that the degree of each vertex in the graph $\overline{G}[V_X]$ is exactly equal to 1. Hence, there is no path of length 2 in $\overline{G}[V_X]$. Also, since $V_{\mathcal{C}}$ forms a clique in $\overline{G}$, any induced cycle of length at least 5 in $\overline{G}$ will either contain a vertex or an edge from $V_{\mathcal{C}}$.

Let C' be an induced odd cycle of length at least 5 in $\overline{G}$. There will be at most two vertices from $V_{\mathcal{C}}$ which are part of C' and these vertices will appear consecutively in C'. This implies that C' contains a path of length at least 2 using only vertices from V_X in $\overline{G}$, which is a contradiction. ◀

▶ **Claim 15.** *The graph G does not contain an induced odd cycle of length ≥ 5.*

Proof. We first show that any induced odd cycle of length at least 5 can contain at most 3 vertices from V_X. Suppose not. Let C' be an induced odd cycle of length at least 5 such that $|V(C') \cap V_X| \geq 4$. Let v_w, v_x, v_y, v_z be four distinct vertices from $V(C') \cap V_X$ appearing in that order, if we go around the cycle in a clockwise manner. That is, there are paths $v_w - v_x$, $v_x - v_y$, $v_y - v_z$ in C'. Since C' is an induced cycle, there is no edge (v_w, v_y) in $E(G)$. This implies that $y = \bar{w}$. By similar arguments, we can show that $x = \bar{z}$. This implies that $v_w v_x v_y v_z v_w$ form a cycle of length 4 in G, contradicting the fact that C' is induced odd cycle containing v_w, v_x, v_y and v_z. Hence any induced odd cycle of length at least 5 can contain at most 3 vertices from V_X.

Since $V_{\mathcal{C}}$ is an independent set in G, no two vertices from $V_{\mathcal{C}}$ can occur as consecutive vertices in any cycle. Let C' be an induced odd cycle of length at least 5 in G. Since $|V(C') \cap V_X| \leq 3$ and no two vertices from $V_{\mathcal{C}}$ appear as consecutive vertices in C', it must be the case that $|V(C') \cap V_{\mathcal{C}}| \leq 2$. This implies that the length of C' is exactly equal to 5 and C' is of the form $v_x w^i_{C_1} v_y w^j_{C_2} v_z v_x$, where $i, j \in \{1, 2\}$. Since C' is an induced cycle $(v_x, v_y), (v_y, v_z) \notin E(G)$. This implies that $y = \bar{x} = z$ and hence $v_y = v_z$. This contradicts the fact that C' is a cycle. This completes the proof of the claim. ◀

Proposition 3 and Claims 14 and 15 imply that G is a perfect graph. We now show that (ϕ, n) is a YES instance of CNF-SAT if and only if $(G, n, 1)$ is a YES instance of Partization Recognition.

First, suppose that (ϕ, n) is a YES instance of CNF-SAT. Then there is an assignment τ, such that every clause has at least one literal set to 1. Let $f : \mathcal{C} \to X$ be a map that arbitrarily maps one such satisfying literal to each clause. Note that for a clause C, $(w^1_C, v_{f(C)}), (w^2_C, v_{f(C)}) \notin E(G)$, because $f(C) \in C$. Now we construct n independent sets as follows. For each literal y, if $\tau(y) = 1$, let $I_y = \{w^1_C, w^2_C \mid f(C) = y\} \cup \{v_y\}$. Since $V_{\mathcal{C}}$ is an independent set $I_y \setminus \{v_y\}$ is an independent set. Note that for all $w^i_C \in I_y$, $i \in \{1, 2\}$ we have that $(w^i_C, v_y) \notin E(G)$, because $f(C) = y$ and $y \in C$. This implies that I_y is an independent set. Since τ is an assignment, exactly one of the literals of each variable is assigned 1 by τ. Thus, in this way we form n independent sets. Since τ is a satisfying assignment for ϕ, the function f maps each clause C to a literal which is assigned 1 by τ. This implies that all vertices in $V_{\mathcal{C}}$ are covered by the independent sets constructed above. The vertices in the graph G, which are not covered by the independent sets constructed, correspond to the literals that have been set to 0 by τ. By construction of G and by the definition of an assignment τ, these vertices form a clique. Hence, $(G, n, 1)$ is an YES instance of Partization Recognition.

Conversely, suppose $(G, n, 1)$ is a YES instance of Partization Recognition. Then there is an (r, ℓ)-partition $\mathcal{P}$ of G. Let $I_1, \ldots, I_n$ be n independent sets and K be a clique in the (r, ℓ)-partition $\mathcal{P}$. It is not possible, by construction of G, that there is a variable x such that both v_x and $v_{\bar{x}}$ belong to the clique K, because $(v_x, v_{\bar{x}}) \notin E(G)$. As there is only one clique in $\mathcal{P}$, at most one literal of each variable can be contained in the clique K of $\mathcal{P}$. Hence, for each variable x either v_x or $v_{\bar{x}}$ is part of an independent set in $\mathcal{P}$. Furthermore, since for two literals p and q such that $p \neq \bar{q}$, $(v_p, v_q) \in E(G)$, any independent set I in $\mathcal{P}$ cannot contain both v_p and v_q. This implies that each of the n independent sets can be identified by a variable $x \in X$. Since there are only n independent sets in $\mathcal{P}$, there cannot be a variable x such that both v_x and $v_{\bar{x}}$ are part of distinct independent sets in $\mathcal{P}$. Thus the construction of G forces only two possibilities for each variable:

(a) there is exactly one literal vertex that is part of an independent set while the other one belong to the clique K, or

(b) both literals together form an independent set.

Now we construct an assignment τ and show that τ is a satisfying assignment for ϕ. For a literal z, if $v_z \in K$, then we set $\tau(z) = 0$. If for a variable x, both vertices v_x and $v_{\bar{x}}$ do not belong to K then we set $\tau(x) = 1$. Now we show that τ is a satisfying assignment for ϕ. Let C be a clause in the formula ϕ. Since $(w_C^1, w_C^2) \notin E(G)$ at least one of w_C^1 or w_C^2 belongs to an independent set I in $\mathcal{P}$. Let $w_C^i \in I$, where $i \in \{1, 2\}$. We have shown that each independent set contains at least one vertex corresponding to a literal y. Since v_y and w_C^i belong to I, we have that $(v_z, w_C^i) \notin E(G)$. This implies that $y \in C$. Furthermore, this implies that $v_{\bar{y}} \notin I$ as no clause contains both y and $\bar{y}$. Hence, $v_{\bar{y}}$ is in K and $\tau(\bar{y}) = 0$. This implies that y is set to 1 and hence the clause C is satisfied. This proves that τ is a satisfying assignment for ϕ. ◀

Note that Partization Recognition is same as Vertex Partization, when $k = 0$. Hence we get the following corollary.

▶ **Corollary 16.** Vertex Partization *parameterized by* $k + r + \ell$ *on perfect graphs does not have a polynomial kernel unless* NP $\subseteq$ co-NP*/poly.*

5 Polynomial kernel when r and ℓ are constants

We saw that there is no polynomial kernel for Vertex Partization, unless NP $\subseteq$ co-NP/poly. The parameter for this problem is $k+r+\ell$, where the size of the deletion set is at most k and the final graph is an (r, ℓ)-graph. In this section, we consider the Vertex (r, ℓ) Partization problem on perfect graphs, which is a special case of Vertex Partization on perfect graphs. Here, for a given pair of fixed positive constants (r, ℓ), we take a perfect graph G and a positive integer k as input and decide whether there is a vertex set S of size at most k the deletion of which results in an (r, ℓ)-graph. This simplified problem has a polynomial kernel, as shown below.

We first observe that when perfect graphs are (r, ℓ)-graphs, this class coincides with another graph class, namely the class of perfect graphs that are (r, ℓ)-split graphs. The notion of (r, ℓ)-split graphs was introduced in [14].

▶ **Definition 17** ((r, ℓ)-split graph)**.** A graph G is an (r, ℓ)-split graph if its vertex set can be partitioned into V_1 and V_2 such that the size of a largest clique in $G[V_1]$ is bounded by r and the size of a largest independent set in $G[V_2]$ is bounded by ℓ. We call such a partition, (V_1, V_2), an (r, ℓ)-split partition.

From the definition of (r, ℓ)-split graph, it follows that any (r, ℓ)-graph is also an (r, ℓ)-split graph.

▶ **Lemma 18.** *Let G be a perfect graph. If G is an (r, ℓ)-split graph, then G is also an (r, ℓ)-graph.*

Proof. Since G is a perfect graph, for any induced subgraph G' of G, the chromatic number of G' ($\chi(G')$) is equal to the cardinality of a maximum sized clique of G' ($\omega(G')$). We know that G is an (r, ℓ)-split graph. Let (P_1, P_2) be an (r, ℓ)-split partition with $\omega(G[P_1]) \leq r$ and $\alpha(G[P_2]) \leq \ell$ Now we show that (P_1, P_2) is indeed an (r, ℓ)-partition of G. Since G is a perfect graph, the graphs $G[P_1]$ and $\overline{G}[P_2]$ are perfect graphs. Since $G[P_1]$ is a perfect graph and $w(G[P_1]) \leq r$, $\chi(G[P_1]) \leq r$. This implies that P_1 can be partitioned into r independent sets. Since $\overline{G}[P_2]$ is a perfect graph and $\alpha(G[P_2]) \leq \ell$, $w(\overline{G}[P_2]) \leq \ell$ and hence $\chi(\overline{G}[P_2]) \leq \ell$. This implies that P_2 can be partitioned into ℓ sets such that each set is independent in $\overline{G}[P_2]$. Hence P_2 can be partitioned into ℓ cliques in $G[P_2]$. So (P_1, P_2) is an (r, ℓ)-partition of G. This completes the proof of the lemma. ◀

For any fixed r and ℓ, there is a finite forbidden set $\mathbb{F}_{r,\ell}$ for (r,ℓ)-split graphs [14]. That is, a graph G is an (r,ℓ)-split graph if and only if G does not contain any graph $H \in \mathbb{F}_{r,\ell}$ as an induced subgraph. The size of the largest forbidden graph is bounded by $f(r,\ell)$, f being a function given in [14]. In [18], a bound was given for the number of minimal forbidden perfect graphs. This function, say $g(r,\ell)$, has a bound of $2(\ell+1)(R(r(\ell+1),(r(l+1))^2+r(l+1)+3)+1)$. Since $g(r,\ell)$ is a constant, it is possible to compute the forbidden set $\mathbb{F}'_{r,\ell}$ of perfect forbidden graphs in polynomial time. Thus, the class of perfect (r,ℓ)-graphs has a refined finite forbidden characterization. This implies that Vertex (r,ℓ) Partization on perfect graphs reduces to the d-Hitting Set problem, where d is the constant $g(r,\ell)$. In an equivalent d-Hitting Set instance, the universe will be the set of vertices of the input graph G, while the family of sets will be the vertex sets of induced subgraphs of G that are isomorphic to a forbidden graph. The set sizes are bounded by $g(r,\ell)$. Hence, by [1], this problem has a polynomial kernel. This gives us the following theorem.

▶ **Theorem 19.** Vertex (r,ℓ) Partization *on perfect graphs admits a kernel of size* $k^{\mathcal{O}(d)}$ *and has an algorithm with running time* $d^k n^{\mathcal{O}(d)}$. *Here,* $d = g(r,\ell)$.

6 Conclusion

In this paper we studied the Vertex Partization problem on perfect graphs, and showed that it is FPT and does not admit a polynomial kernel. Furthermore, we observed that Vertex (r,ℓ) Partization has an induced finite forbidden characterization and utilized that to give a faster FPT algorithm and a polynomial kernel for the problem. However, the algorithms for Vertex (r,ℓ) Partization has a factor of $n^{\mathcal{O}(d)}$, where d depends on the size of a largest graph in the finite forbidden set. It would be interesting to replace the factor $n^{\mathcal{O}(d)}$ by $\tau(d) \cdot n^{\mathcal{O}(1)}$.

References

1. Faisal N. Abu-Khzam. A kernelization algorithm for d-hitting set. *J. Comput. Syst. Sci.*, 76(7):524–531, 2010.
2. Julien Baste, Luerbio Faria, Sulamita Klein, and Ignasi Sau. Parameterized complexity dichotomy for (r, ℓ)-vertex deletion. *CoRR*, abs/1504.05515, 2015.
3. Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. Kernel bounds for disjoint cycles and disjoint paths. *Theor. Comput. Sci.*, 412(35):4570–4578, 2011.
4. Andreas Brandstädt, Van Bang Le, and Thomas Szymczak. The complexity of some problems related to graph 3-colorability. *Discrete Applied Mathematics*, 89(1–3):59–73, 1998.
5. Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theor. Comput. Sci.*, 411(40-42):3736–3756, 2010.
6. Maria Chudnovsky, Neil Robertson, Paul D. Seymour, and Robin Thomas. The strong perfect graph theorem. *Annals of Mathematics*, 164:51–229, 2006.
7. Marek Cygan and Marcin Pilipczuk. Split vertex deletion meets vertex cover: New fixed-parameter and exact exponential-time algorithms. *Inf. Process. Lett.*, 113(5-6):179–182, 2013.
8. Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.
9. Tomas Feder, Pavol Hell, Sulamita Klein, and Rajeev Motwani. List partitions. *STOC*, 16:464–472, 2003.
10. Tomás Feder, Pavol Hell, and Shekoofeh Nekooei Rizi. Partitioning chordal graphs. *Electronic Notes in Discrete Mathematics*, 38:325–330, 2011.

11 Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct pcps for NP. *J. Comput. Syst. Sci.*, 77(1):91–106, 2011.

12 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

13 Esha Ghosh, Sudeshna Kolay, Mrinal Kumar, Pranabendu Misra, Fahad Panolan, Ashutosh Rai, and M. S. Ramanujan. Faster parameterized algorithms for deletion to split graphs. *Algorithmica*, 71(4):989–1006, 2015.

14 András Gyárfás. Generalized split graphs and ramsey numbers. *J. Comb. Theory, Ser. A*, 81(2):255–261, 1998.

15 Pinar Heggernes, Dieter Kratsch, Daniel Lokshtanov, Venkatesh Raman, and Saket Saurabh. Fixed-parameter algorithms for cochromatic number and disjoint rectangle stabbing via iterative localization. *Inf. Comput.*, 231:109–116, 2013.

16 Sudeshna Kolay and Fahad Panolan. Parameterized algorithms for deletion to (r, ell)-graphs. In *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2015, December 16-18, 2015, Bangalore, India*, pages 420–433, 2015.

17 R. Krithika and N. S. Narayanaswamy. Parameterized algorithms for (r, l)-partization. *J. Graph Algorithms Appl.*, 17(2):129–146, 2013.

18 André E. Kézdy, Hunter S. Snevily, and Chi Wang. Partitioning permutations into increasing and decreasing subsequences. *Journal of Combinatorial Theory, Series A*, 73(2):353–359, 1996.

19 John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is np-complete. *J. Comput. Syst. Sci.*, 20(2):219–230, 1980.

20 Daniel Lokshtanov, N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Faster parameterized algorithms using linear programming. *ACM Transactions on Algorithms*, 11(2):15, 2014.

21 László Lovász. A characterization of perfect graphs. *J. Comb. Theory, Ser. B*, 13(2):95–98, 1972.

22 Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Oper. Res. Lett.*, 32(4):299–301, 2004. doi:10.1016/j.orl.2003.10.009.

23 Klaus W. Wagner. Monotonic coverings of finite sets. *Elektronische Informationsverarbeitung und Kybernetik*, 20(12):633–639, 1984.