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

Sudeshna Kolay and Fahad Panolan

Institute of Mathematical Sciences, Chennai, India
{skolay,fahad}@imsc.res.in

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. This brings us to the following natural parameterized questions: Vertex (r, ℓ)-Partization and Edge (r, ℓ)-Partization. An input to these problems consist of a graph G and a positive integer k and the objective is to decide whether there exists a set $S \subseteq V(G)$ ($S \subseteq E(G)$) such that the deletion of S from G results in an (r, ℓ)-graph. These problems generalize well studied problems such as Odd Cycle Transversal, Edge Odd Cycle Transversal, Split Vertex Deletion and Split Edge Deletion. We do not hope to get parameterized algorithms for either Vertex (r, ℓ)-Partization or Edge (r, ℓ)-Partization when either of r or ℓ is at least 3 as the recognition problem itself is NP-complete. This leaves the case of $r, \ell \in \{1, 2\}$. We almost complete the parameterized complexity dichotomy for these problems by obtaining the following results:

1. We show that Vertex (r, ℓ)-Partization is fixed parameter tractable (FPT) for $r, \ell \in \{1, 2\}$. Then we design an $\mathcal{O}(\sqrt{\log n})$-factor approximation algorithms for these problems. These approximation algorithms are then utilized to design polynomial sized randomized Turing kernels for these problems.
2. Edge (r, ℓ)-Partization is FPT when $(r, \ell) \in \{(1, 2), (2, 1)\}$. However, the parameterized complexity of Edge $(2, 2)$-Partization remains open.

For our approximation algorithms and thus for Turing kernels we use an interesting finite forbidden induced graph characterization, for a class of graphs known as (r, ℓ)-split graphs, properly containing the class of (r, ℓ)-graphs. This approach to obtain approximation algorithms could be of an independent interest.

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

Keywords and phrases FPT, Turing kernels, Approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2015.420

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. Although the problem has an abstract setting, some special cases are well known graph classes and have been widely studied. 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. Already, we get a hint of an interesting dichotomy for this graph class, even with respect to recognition algorithms. Throughout the paper we will use m and n to denote the number of edges and the number of vertices, respectively, in the input graph G. It is well known that we can recognize $(2, 0)$- and $(1, 1)$-graphs 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 [2, 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]. These problems are also studied when the input is restricted to be a chordal graph, in which case we can get polynomial time recognition algorithms for every r and ℓ [10].

The topic of this paper is to design recognition algorithms for *almost* (r, ℓ)-graphs in the realm of parameterized algorithms. In particular, we study the following natural parameterized questions on (r, ℓ)-graphs: Vertex (r, ℓ)-Partization and Edge (r, ℓ)-Partization.

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?	

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

These problems generalize some of the most well studied problems in parameterized complexity, such as Vertex Cover, Odd Cycle Transversal (OCT), Edge Odd Cycle Transversal (EOCT), Split Vertex Deletion (SVD) and Split Edge Deletion (SED). Vertex Cover, in particular, has been extensively studied in the parameterized complexity, and the current fastest algorithm runs in time $1.2738^k n^{\mathcal{O}(1)}$ and has a kernel with $2k$ vertices [4]. The parameterized complexity of OCT was a well known open problem for a long time. In 2003, in a breakthrough paper, Reed et al. [25] showed that OCT is FPT by developing an algorithm for the problem running in time $\mathcal{O}(3^k mn)$. In fact, this was the first time that the iterative compression technique was used. However, the algorithm for OCT had seen no further improvements in the last 9 years, though several reinterpretations of the algorithm have been published [16, 22]. Only recently, Lokshtanov et al. [21] obtained a faster algorithm for the problem running in time $2.3146^k n^{\mathcal{O}(1)}$ using a branching algorithm based on linear programming. Guo et al. [14] designed an algorithm for EOCT running in time $2^k n^{\mathcal{O}(1)}$. There is another theme of research in parameterized complexity, where the objective is to minimize the dependence of n at the cost of a slow growing function of k. A well known open problem, in the area, is whether OCT admits a linear time parameterized algorithms. Only recently, the first linear time FPT algorithms for OCT on general graphs were obtained, both of which run in time $\mathcal{O}(4^k k^{\mathcal{O}(1)}(m+n))$ [24, 17]. Kratsch and Wahlström [19] obtained a randomized polynomial kernel for OCT and EOCT. Ghosh et al. [13] studied SVD and SED and designed algorithms with running time $2^k n^{\mathcal{O}(1)}$ and $2^{\mathcal{O}(\sqrt{k}\log k)} n^{\mathcal{O}(1)}$. They also gave the best known polynomial kernel for these problems. Later, Cygan and Pilipczuk [7] designed an algorithm for SVD running in time $1.2738^{k+o(k)} n^{\mathcal{O}(1)}$. Krithika and Narayanaswamy [20] studied Vertex (r, ℓ)-Partization problems on perfect graphs, and among several results they obtain $(r+1)^k n^{\mathcal{O}(1)}$ algorithm for Vertex $(r, 0)$-Partization on perfect graphs.

Our Results and Methods. The instance of a parameterized problem is a pair containing the actual problem instance of size n and a positive integer called a parameter, usually represented as k. The problem is said to be in FPT if there exists an algorithm that solves the problem in $f(k)n^{\mathcal{O}(1)}$ time, where f is a computable function. An algorithm with such a running time is called an FPT algorithm. Readers are requested to refer [11] for more details. We do not hope to get FPT algorithms for either Vertex (r, ℓ)-Partization or Edge (r, ℓ)-Partization when either of r or ℓ is at least 3 as the recognition problem itself is

r, ℓ	Problem Name	FPT	Kernel
$(1, 0)$	Vertex Cover	1.2738^k	Poly
$(0, 1)$	Vertex Cover on $\overline{G}$	1.2738^k	Poly
$(1, 1)$	SVD	$1.2738^{k+o(k)}$	Poly
$(2, 0)$	OCT	2.3146^k	Randomized Poly
$(0, 2)$	OCT on $\overline{G}$	2.3146^k	Randomized Poly
$(2, 1)$, $(1, 2)$, $(2, 2)$	Vertex $(2, 1)$-partization Vertex $(1, 2)$-partization Vertex $(2, 2)$-partization	3.3146^k	**Randomized Turing Poly**

Figure 1 Summary of known and new results for the family of Vertex (r, ℓ)-Partization problems. New results are highlighted in green (last row).

r, ℓ	Problem Name	FPT	Kernel
$(1, 0)$	*Recognizable in polynomial time.*		
$(0, 1)$	*Recognizable in polynomial time.*		
$(1, 1)$	SED	$2^{\mathcal{O}(\sqrt{k} \log k)}$	Poly
$(2, 0)$	EOCT	2^k	Randomized Poly
$(0, 2)$	*Recognizable in polynomial time.*		
$(2, 1)$	Edge $(2, 1)$-partization	$2^{k+o(k)}$	**Open**
$(1, 2)$	Edge $(1, 2)$-partization	**FPT**	**Open**
$(2, 2)$	Edge $(2, 2)$-partization	**Open**	

Figure 2 Summary of known and new results for the family of Edge (r, ℓ)-Partization problems. New results are highlighted in green.

NP-complete. This leaves the case of $r, \ell \in \{0, 1, 2\}$. We almost complete the parameterized complexity dichotomy for these problems by either obtaining new results or using the existing results. We refer to Figures 1 and 2 for a summary of new and old results. Due to paucity of space, some proofs have had to be omitted from the paper. However, all such results (marked with a $\star$) have their complete proofs in the full version.[1]

For both Vertex (r, ℓ)-Partization and Edge (r, ℓ)-Partization, the only new cases for which we need to design new parameterized algorithms to complete the dichotomy is when $r, \ell \in \{1, 2\}$. Apart from the algorithmic results indicated in the Figures 1 and 2, we also obtain the following results. When $r, \ell \in \{1, 2\}$, we obtain an $\mathcal{O}(\sqrt{\log n})$-approximation for these special cases. Finally, we obtain randomized *Turing kernels* for Vertex (r, ℓ)-Partization using this approximation algorithms. In other words, we give a polynomial time algorithm that produces polynomially many instances, $n^{\mathcal{O}(1)}$ of Vertex (r, ℓ)-Partization of size $k^{\mathcal{O}(1)}$ such that with very high probability (G, k) is a YES instance of Vertex (r, ℓ)-Partization if and only if one of the polynomially many instances of Vertex (r, ℓ)-Partization of size $k^{\mathcal{O}(1)}$ is a YES instance. The question of existence of polynomial kernels for these special cases as well as for Edge (r, ℓ)-Partization is open. Even the parameterized complexity of Edge $(2, 2)$-Partization remains open.

[1] We would like to mention that one of our results, namely, $3.3146^k n^{\mathcal{O}(1)}$ time FPT algorithm for Vertex $(2, 2)$-Partization (and hence for Vertex $(1, 2)$-Partization and Vertex $(2, 1)$-Partization) were independently and simultaneously obtained by Baste et al. (`http://arxiv.org/abs/1504.05515`).

Our methods. Most of the FPT algorithms are based on the iterative compression technique and use an algorithm for either OCT or EOCT as a subroutine. One of the algorithms also uses methods developed in [23]. To arrive at the approximation algorithm, we needed to take a detour. We start by looking at 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 the largest independent set in $G[V_2]$ is bounded by ℓ. Such a bipartition for the graph G is called as (r,ℓ)-split partition. The notion of (r,ℓ)-split graphs was introduced in [15]. For any fixed r and ℓ, there is a finite forbidden set $\mathbb{F}_{r,\ell}$ for (r,ℓ)-split graphs [15]. That is, a graph G is a (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 [15]. Since the class (r,ℓ)-graphs is a sub class of (r,ℓ)-split graphs, each graph in $\mathbb{F}_{r,\ell}$ will not appear as an induced subgraph in any (r,ℓ)-graph. For our approximation algorithm we first make the given graph (r,ℓ)-split graph by removing the induced subgraphs that are isomorphic to some graph in $\mathbb{F}_{r,\ell}$. Once we have (r,ℓ)-split graph, we generate a (r,ℓ)-split partition (V_1,V_2) of G. Then we observe that for $r,\ell \in \{1,2\}$ the problem reduces to finding an approximate solution to Odd Cycle Transversal in $G[V_1]$ and $\overline{G}[V_2]$. Finally, we use the known $\mathcal{O}(\sqrt{\log n})$-approximation algorithm for Odd Cycle Transversal [1] to obtain a $\mathcal{O}(\sqrt{\log n})$-approximation algorithm for our problems. The Turing kernel for Vertex (r,ℓ)-Partization, when $r,\ell \in \{1,2\}$, uses the approximation algorithm and depends on the randomized kernelization algorithm for Odd Cycle Transversal [19].

2 Preliminaries

We use standard notations from graph theory([8]) throughout this paper. The vertex set and edge set of a graph G are denoted as $V(G)$ and $E(G)$ respectively. The complement of the graph G is denoted by $\overline{G}$ and has $V(G')=V(G)$ and $\binom{V(G)}{2} - E(G)$ 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 $G[V']$. An induced subgraph of G on the edge set $E' \subseteq E$ is written as $G[E']$. For a vertex subset $V' \subseteq V$, $G[V-V']$ is also denoted as $G-V'$. Similarly, for an edge set $E' \subseteq E$, $G-E'$ denotes the subgraph $G'=(V,E\setminus E')$.

The Ramsey number for a given pair of positive integers (a,b) is the minimum number such that any graph with the Ramsey number of vertices either has an independent set of size a or a clique of size b. The Ramsey number for (a,b) is denoted by $R(a,b)$.

We have already seen what (r,ℓ)-graphs are. Below, is a formal definition of the graph class as well as some related definitions.

▶ **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. 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.

For fixed $r,\ell \geq 0$, the class of (r,ℓ)-graphs is closed under induced subgraphs. The following observation is useful in the understanding of the algorithms presented in the paper

▶ **Observation 2 (⋆).** [2] *Let $P=(P_I,P_C)$ and $P'=(P'_I,P'_C)$ be two IC-partitions of an (r,ℓ)-graph G. Then $|P_I \cap P'_C| \leq r\ell$ and $|P'_I \cap P_C| \leq r\ell$.*

[2] Proofs of results marked with ⋆ can be found in the full version.

3 Vertex Deletion to (r, ℓ)-graphs

In this section we first show that Vertex $(2,2)$-Partization is in FPT, using iterative compression. Then we explain how to reduce Vertex $(2,1)$-Partization and Vertex $(1,2)$-Partization to Vertex $(2,2)$-Partization. Our algorithm for Vertex $(2,2)$-Partization combines the iterative compression technique with a polynomial bound on the number of IC-partitions of a $(2,2)$-graph. The following Lemma tells about an algorithm to recognize whether a graph is a $(2,2)$-graph and also about an algorithm to compute all such IC-partitions. These results were shown in several papers [2, 9].

▶ **Lemma 3.** *Given a graph G on n vertices and m edges we can recognize whether G is a $(2,2)$-graph in $\mathcal{O}((n+m)^2)$ time. Also, a $(2,2)$-graph can have at most n^8 IC-partitions and all the IC-partitions can be enumerated in $\mathcal{O}(n^8)$ time.*

For a graph G, we say $S \subseteq V(G)$ is a $(2,2)$-vertex deletion set, if $G - S$ is a $(2,2)$-graph. Now we describe the iterative compression technique and its application to the Vertex $(2,2)$-Partization problem.

Iterative Compression for Vertex $(2,2)$-Partizipation. Let (G,k) be an input instance of Vertex $(2,2)$-Partization and let $V(G) = \{v_1, \ldots, v_n\}$. We define, for every $1 \leq i \leq |V(G)|$, the vertex set $V_i = \{v_1, \ldots, v_i\}$. Denote $G[V_i]$ as G_i. We iterate through the instances (G_i, k) starting from $i = k+5$. Given the i^{th} instances and a known $(2,2)$-vertex deletion set S'_i of size at most $k+1$, our objective is to obtain a $(2,2)$-vertex deletion set S_i of size at most k. The formal definition of this compression problem is as follows.

Vertex $(2,2)$-Partization Compression	**Parameter:** k
Input: A graph G and a $k+1$ sized vertex subset $S' \subseteq V(G)$ such that $G - S'$ is a $(2,2)$-graph	
Output: A vertex subset $S \subseteq V(G)$ of size at most k such that $G - S$ is a $(2,2)$-graph	

We reduce the Vertex $(2,2)$-Partization problem to $n-k-4$ instances of the Vertex $(2,2)$-Partization Compression problem in the following manner. When $i = k+5$, the set V_{k+1} is a $(2,2)$-vertex deletion set of size at most $k+1$ for G_{k+5}. Let $I_i = (G_i, S'_i, k)$ be the i^{th} instance of Vertex $(2,2)$-Partization Compression. If S_{i-1} is a k-sized solution for I_i, then $S_{i-1} \cup \{v_i\}$ is a $(k+1)$-sized $(2,2)$-vertex deletion set for G_i. Hence, we start the iteration with the instance $I_{k+5} = (G_{k+5}, V_{k+1}, k)$ and try to obtain a $(2,2)$-vertex deletion set of size at most k. If such a solution S_{k+5} exists, we set $S'_{k+5} = S_{k+5} \cup \{v_{k+6}\}$ and ask of a k-sized solution for the instance I_{k+6}, and so on. If, during any iteration, the corresponding instance does not have a $(2,2)$-vertex deletion set of size at most k, it implies that the original instance (G,k) is a NO instance for Vertex $(2,2)$-Partization. If the input instance (G,k) is a YES instance, then S_n is a k-sized $(2,2)$-vertex deletion set for G, where $n = |V(G)|$. Since there are at most n iterations, the total time taken by the algorithm to solve Vertex $(2,2)$-Partization is at most n times the time taken to solve Vertex $(2,2)$-Partization Compression. The above explained template for doing iterative compression will be used for approximation algorithms as well as for parameterized algorithms for edge versions of these problems.

The following Lemma shows that Vertex $(2,2)$-Partization Compression is in FPT. The arguments above imply that Vertex $(2,2)$-Partization is also in FPT.

▶ **Lemma 4 ($\star$).** Vertex $(2,2)$-Partization Compression *can be solved deterministically in time* $3.3146^k |V(G)|^{\mathcal{O}(1)}$.

Proof. (Proof sketch) We design an algorithm for Vertex $(2,2)$-Partization Compression. Let (G, S') be the instance of the problem and let (P'_I, P'_C) be an IC-partition of $G - S'$. Let S be a *hypothetical solution* of size k for the problem, which the algorithm suppose to compute. Let (P_I, P_C) be an IC-partition of $G - S$. The algorithm first guesses a partition (Y, N) of S' such that $Y = S' \cap S$ and $N = S' - S$. After this guess, the objective is to compute a set Z of size at most $k' = k - |Y|$ such that $G - (Z \cup Y)$ is a $(2,2)$-graph. Also note that since N is not part of the solution S, $G[N]$ is a $(2,2)$-graph. Consider the two IC-partitions $(P_I - (S \cup S'), P_C - (S \cup S'))$ and $(P'_I - (S \cup S'), P'_C - (S \cup S'))$ of the $(2,2)$-graph $G - (S \cup S')$. By Observation 2 we know that the cardinality of each of the set $(P_I \cap P'_C) - (S \cup S')$ and $(P_C \cap P'_I) - (S \cup S')$ are bounded by 4. So now the algorithm guesses the set $V_I = (P_I \cap P'_C) - (S \cup S')$ and $V_C = (P_C \cap P'_I) - (S \cup S')$, each of them having size at most 4. After the guess of V_I and V_C, any vertex in $P'_C - V_I$ either belongs to P_C or belongs to the hypothetical solution S. Similarly any vertex in $P'_I - V_C$ either belongs to P_I or belongs to the hypothetical solution S. By Lemma 3 we know that the number of IC-partitions of $G[N]$ is at most $\mathcal{O}(k^8)$ and these partitions can be enumerate in time $\mathcal{O}(k^8)$. The algorithm now guesses an IC-partition (N_I, N_C) of $G[N]$ such that $N_I \subseteq P_I$ and $N_C \subseteq P_C$. Now consider the partition $(A, B) = ((P'_I \cup N_I \cup V_I) - V_C, (P'_C \cup N_C \cup V_C) - V_I)$. Any vertex $v \in A$ either belongs to P_I or belongs to the hypothetical solution S and any any vertex $v \in B$ either belongs to P_C or belongs to the solution S. So the objective is to find two sets $U \subseteq A$ and $W \subseteq B$ such that $G[A - U]$ is a bipartite graph, $G[B - W]$ is the complement of a bipartite graph and $|U| + |W| \leq k'$. As a consequence, the algorithm guesses the sizes k_1 of U and k_2 of W. Then the problem reduced to finding an odd cycle transversal(OCT) of size k_1 for $G[A]$ and an OCT of size k_2 for the complement of the graph $G[B]$. Hence, our algorithm runs the current best algorithm for Odd Cycle Transversal, presented in [21] for finding an OCT U of size k_1 in $G[A]$ and for finding an OCT W of size k_2 in the complement of $G[B]$. This completes the algorithm and it leads to the mentioned running time in the lemma. The running time analysis can be found in the full version. ◀

Lemma 4 and the discussions preceding it imply the following theorem.

▶ **Theorem 5.** Vertex $(2,2)$-Partization *can be solved in time* $3.3146^k |V(G)|^{\mathcal{O}(1)}$.

Vertex $(2,1)$-Partization: The Vertex $(2,1)$-Partization problem can be reduced to Vertex $(2,2)$-Partization. Suppose we are given a graph G, where $|V(G)| = n$. We construct a graph $G' = G \uplus \hat{C}$, where $\hat{C}$ is a clique on $n+3$ new vertices. That is, G' is the disjoint union of G and $\hat{C}$. The next lemma relates the graphs G and G'.

▶ **Lemma 6** ($\star$). *For any integer $t \leq n$, (G, t) is a* YES *instance of* Vertex $(2,1)$-Partization *if and only if (G', t) is a* YES *instance of* Vertex $(2,2)$-Partization. *Here $G' = G \uplus \hat{C}$ such that $\hat{C}$ is a clique on $n+3$ new vertices that are independent from G.*

Now if we are given an instance (G, k) of Vertex $(2,1)$-Partization, Lemma 6 tells us that it is enough to solve Vertex $(2,2)$-Partization on (G', k). Notice that solving the Vertex $(1,2)$-Partization problem on an input instance (G, k) is equivalent to finding a Vertex $(1,2)$-Partization on $(\overline{G}, k)$, where $\overline{G}$ is the complement graph of G. Thus, we get the following as a corollary of Theorem 5.

▶ **Corollary 7.** Vertex $(1,2)$-Partization *and* Vertex $(2,1)$-Partization *have* FPT *algorithms that run in* $3.3146^k n^{\mathcal{O}(1)}$ *time.*

4 Approximation algorithm for Vertex (r, ℓ)-Partization

In this section we give a polynomial time approximation algorithm for Vertex $(2,2)$-Partization. That is, we design an algorithm for Vertex $(2,2)$-Partization, which takes an instance (G, k), runs in polynomial time and outputs either a solution of size $\mathcal{O}(k^{3/2})$ or concludes that (G, k) is a NO instance. Since the reduction from Vertex $(2,1)$-Partization to Vertex $(2,2)$-Partization, given in Lemma 6, is an approximation preserving reduction, we can get a similar approximate algorithm for Vertex $(2,1)$-Partization. Similarly, since Vertex $(1,2)$-Partization on a graph is equivalent to Vertex $(2,1)$-Partization in the complement graph, we can get an approximation algorithm for Vertex $(1,2)$-Partization. The approximation algorithm we discuss in this section, is useful for obtaining Turing kernels for Vertex (r, ℓ)-Partization, when $1 \leq r, \ell \leq 2$. Finally, we design a factor $\mathcal{O}(\sqrt{\log n})$ approximation algorithms for these problems.

We know that (r, ℓ)-graphs is a subclass of (r, ℓ)-split graphs (See Introduction for definition). Now we give a polynomial time algorithm which takes a graph G as input and outputs an (r, ℓ)-split partition if G is an (r, ℓ)-split graph. We design such an algorithm using iterative compression. Essentially we show that the following problem, (r, ℓ)-split partition Compression, can be solved in polynomial time.

(r, ℓ)-split partition Compression
Input: A graph G with $V(G) = V \cup \{v\}$ and an (r, ℓ)-split partition (A, B) of $G[V]$
Output: An (r, ℓ)-split partition of G, if G is an (r, ℓ)-split graph, and NO otherwise

Like in the case of the FPT algorithm for Vertex $(2,2)$-Partization given in Section 3, we can show that by running the algorithm for (r, ℓ)-split partition Compression at most $n - 2$ times we can get an algorithm which outputs an (r, ℓ)-split partition of a given (r, ℓ)-split graph. Our algorithm for (r, ℓ)-split partition Compression uses the following simple lemma.

▶ **Lemma 8** (⋆). *Let G be an (r, ℓ)-split graph. Let (A, B) and (A', B') are two (r, ℓ)-split partitions of G. Then $|A \cap B'| \leq R(\ell + 1, r + 1) - 1$ and $|A' \cap B| \leq R(\ell + 1, r + 1) - 1$, where $R(r + 1, \ell + 1)$, is the Ramsey number.*

Using Lemma 8, we show that (r, ℓ)-split partition Compression can be solved in polynomial time for any fixed constants r and ℓ.

▶ **Lemma 9** (⋆). *For any fixed constants r and ℓ, (r, ℓ)-split partition Compression can be solved in polynomial time.*

By applying Lemma 9, at most $n - 2$ times, we can get the following lemma.

▶ **Lemma 10.** *For any fixed constants r and ℓ, there is an algorithm which takes a graph G as input, runs in polynomial time, and decides whether G is an (r, ℓ)-split graph. Furthermore, if G is an (r, ℓ)-split graph then the algorithm outputs an (r, ℓ)-split partition (V_1, V_2) of G.*

We know that any (r, ℓ)-graph is also an (r, ℓ)-split graph. The following lemma gives a relation between an (r, ℓ)-split partition and an IC-partition of a (r, ℓ)-graph.

▶ **Lemma 11** (⋆). *Let G be an (r, ℓ)-graph. Let (A, B) be an IC-partition of G and (A', B') be an (r, ℓ)-split partition of G. Then $|A \cap B'| \leq r\ell$ and $|A' \cap B| \leq r\ell$.*

Before giving an approximation algorithm for Vertex (r, ℓ)-Partization, we need to mention about a polynomial time approximation algorithm for Odd Cycle Transversal and finite forbidden characterization of (r, ℓ)-graphs. Using the FPT algorithm

for OCT [19], and an $\mathcal{O}(\sqrt{\log n})$-approximation algorithm for OCT [1], one can prove the following proposition.

▶ **Proposition 12** ([19]). *There is a polynomial time algorithm which takes a graph G and an integer k as input and outputs either an OCT of G of size at most $\mathcal{O}(k^{3/2})$ or concludes that there is no OCT of size k for G.*

For any fixed r and ℓ, there is a finite forbidden set $\mathbb{F}_{r,\ell}$ for (r,ℓ)-split graphs [15]. 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 [15]. Since $f(2,2)$ is a constant, it is possible to compute the forbidden set $\mathbb{F}_{r,\ell}$ in polynomial time: The forbidden graphs are of size at most $f(2,2)$. Since the class (r,ℓ)-graphs is a sub class of (r,ℓ)-split graphs, each graph in $\mathbb{F}_{r,\ell}$ will not appear as an induced subgraph in any (r,ℓ)-graph. Now we are ready to design a polynomial time approximation algorithm for Vertex $(2,2)$-Partization.

▶ **Theorem 13.** *There is an algorithm which takes a graph G and an integer k as input, runs in polynomial time and outputs either a set S of size $\mathcal{O}(k^{3/2})$ such that $G-S$ is a $(2,2)$-graph or concludes that (G,k) is a* NO *instance of* Vertex $(2,2)$-Partization.

Proof. The algorithm first finds a maximal set $\mathcal{T}$ of vertex disjoint subgraphs of G such that each subgraph in $\mathcal{T}$ is isomorphic to a graph in $\mathbb{F}_{2,2}$. If $|\mathcal{T}| > k$, then clearly (G,k) is a NO instance of Vertex $(2,2)$-Partization. So the algorithm will output NO if $|\mathcal{T}| > k$. Now consider the graph $G' = G - V(\mathcal{T})$. Here, $V(\mathcal{T})$ denotes the set of vertices appearing in graphs in $\mathcal{T}$. Since $\mathcal{T}$ is a maximal set of vertex disjoint subgraphs in G which are isomorphic to a graphs in $\mathbb{F}_{2,2}$ we have that G' is a $(2,2)$-split graph.

Now our algorithm will find a set $S \subseteq V(G')$ of size bounded by $\mathcal{O}(k^{3/2})$ such that $G'-S$ is a $(2,2)$-graph. Since G' is a subgraph of G, if (G,k) is a YES instance of Vertex $(2,2)$-Partization, then (G',k) is also a YES instance. Let S^* be an hypothetical solution of the instance (G',k) of Vertex $(2,2)$-Partization and let (A,B) be an IC-partition of $G'-S^*$. Now our algorithm applies Lemma 10 on graph G' and computes a $(2,2)$-split partition (A',B') of G' in polynomial time. By Lemma 11, we know that $|A \cap B'| \leq 4$ and $|A' \cap B| \leq 4$. So the algorithm will guess the set $U = A \cap B'$ and $W = A' \cap B$. The number of possible guesses for U and W is bounded by n^8. For the correct guess U and W, we know that $A = (A' \cup U) \setminus (W \cup S^*)$ and $B = (B' \cup W) \setminus (U \cup S^*)$. Now consider the partition (V_1, V_2) of $V(G')$, where $V_1 = (A' \cup U) \setminus W$ and $V_2 = (B' \cup W) \setminus U$. So for the correct guess U and W, we know that each vertex in V_1 either belongs to A or belongs to S^* and each vertex in V_2 either belongs to B or belongs to S^*. Now to compute a solution for (G',k), it is enough to find an OCT S_1 in $G[V_1]$ and an OCT S_2 in the complement graph of $G'[V_2]$ such that $|S_1| + |S_2| = k$. Our algorithm applies Proposition 12 on $G'[V_1]$ and on the complement graph of $G'[V_2]$. If these algorithms output an OCT S_1 and an OCT S_2 for graphs $G'[V_1]$ and $\overline{G'}[V_2]$, then $S_1 \cup S_2$ is of size bounded by $\mathcal{O}(k^{3/2})$ and $G' - (S_1 \cup S_2)$ is a $(2,2)$-graph. Since $G' = G - V(\mathcal{T})$ and $G' - (S_1 \cup S_2)$ is a $(2,2)$-graph, we have that $G - (S_1 \cup S_2 \cup V(\mathcal{T}))$ is a $(2,2)$-graph. So our algorithm will output $S_1 \cup S_2 \cup V(\mathcal{T})$ as the required output. Since $|V(\mathcal{T})| \leq k \cdot f(2,2)$, we have that $|S_1 \cup S_2 \cup V(\mathcal{T})| = \mathcal{O}(k^{3/2})$. If the algorithm mentioned in Proposition 12 returns NO for all possible guesses of U and W, then our algorithm outputs NO. It is easy to see that the number of steps in our algorithm is bounded by a polynomial in $|V(G)|$. ◀

Using the arguments of Theorem 13, we can also design an approximation algorithm for finding a minimum $(2,2)$-vertex deletion set of a graph G. Let S be an optimum $(2,2)$-vertex

deletion set and (A, B) be the corresponding IC-partition of $G' = G - S$. Let $\mathcal{T}$ be a maximal set of vertex disjoint subgraphs of G, that are each isomorphic to a graph in $\mathbb{F}_{2,2}$. The number of subgraphs in $\mathcal{T}$ is at most $|S|$ and the number of vertices involved in these forbidden subgraphs is at most $f(2,2)|S|$. The remaining graph G' is a $(2,2)$-split graph and using Lemma 10, we can find a $(2,2)$-split partition (A', B') of G'. Let $(\hat{A}, \hat{B})$ be the restriction of (A, B) to G'. As argued above, at most 4 vertices from A' could be part of $\hat{B}$. Let this set of 4 vertices be called U. The rest either belong to $\hat{A}$ or S. $U \cup (S \cap A')$ is an OCT for A', of size at most $2|S \cap A'|$. The algorithm of [1] returns an $\mathcal{O}(\sqrt{\log n})$-approximate Odd Cycle Transversal solution S_1 for $G[A']$, which has to be of size at most $2|S \cap A'| \cdot \mathcal{O}(\sqrt{\log n})$. There is a similar property on the vertices of B'. Applying the algorithm of [1], on $\overline{G'[B']}$, returns an $\mathcal{O}(\sqrt{\log n})$-approximate Odd Cycle Transversal solution S_2, which has to be of size at most $2|S \cap B'| \cdot \mathcal{O}(\sqrt{\log n})$. Thus $V(\mathcal{T}) \cup S_1 \cup S_2$ is a $(2,2)$-vertex deletion set of G, with size at most $(f(2,2) + \mathcal{O}(\sqrt{\log n})|S|$. This together with Lemma 6 and discussion after that lead to the following theorem.

▶ **Theorem 14.** Vertex $(2,1)$-Partization, Vertex $(1,2)$-Partization, *and* Vertex $(2,2)$-Partization *admit polynomial time approximation algorithms with factor* $\mathcal{O}(\sqrt{\log n})$.

5 Turing Kernels for Vertex Deletion to (r, ℓ)-graphs

In this section, we give a randomized Turing kernel for Vertex $(2,2)$-Partization (See introduction for the definition). The equivalence in Lemma 6 ensures that there is a randomized Turing kernel for Vertex $(2,1)$-Partization. Since, Vertex $(1,2)$-Partization on an instance (G,k) is equivalent to Vertex $(2,1)$-Partization on $(\overline{G},k)$, a randomized Turing kernel for Vertex $(1,2)$-Partization follows.

We have seen in Section 3 that eventually the algorithm for Vertex $(2,2)$-Partization runs two instances of OCT. In this section we explain that we can use the kernelization of OCT to get a Turing kernel for Vertex $(2,2)$-Partization. A randomized polynomial kernel for OCT was shown by Kratsch and Wahlström [18], using the concept of representative family. They showed that it is possible to find $k^{\mathcal{O}(1)}$ "relevant" vertices from the input graph which contains the optimum solution. This leads to a randomized kernel for OCT. In fact, the following lemma follows from the work of Kratsch and Wahlström. We give a proof for the lemma in the full version.

▶ **Lemma 15** ($\star$). *Let G be a graph and X be an OCT of G. There is a randomized polynomial time algorithm which computes a set $Z \subseteq V(G)$ of size $\mathcal{O}(|X|^3)$ such that for any $Y \subseteq X$, a minimum sized OCT, of $G - Y$, is fully contained in Z.*

Now we are ready to explain our Turing kernel for Vertex $(2,2)$-Partization using Lemma 15. Given an instance (G,k) of Vertex $(2,2)$-Partization, first we construct $|V(G)|^{\mathcal{O}(1)}$ many instances of a problem which is in NP and each of them have size bounded by polynomial in k. Then, by using the Cook-Levin theorem [5], we can reduce each of these intances to instances of Vertex $(2,2)$-Partization and thus arrive at a Turing kernelization for Vertex $(2,2)$-Partization. We first run the polynomial time approximation algorithm described in Theorem 13. If the approximation algorithm outputs NO, then the algorithm will output a trivial NO instance of the problem. Otherwise let X be the solution returned by the approximation algorithm on input (G,k). We know that the cardinality of X is bounded by $\mathcal{O}(k^{3/2})$. Now we fix an IC-partition (P_I, P_C) of $G - X$. Let S be a *hypothetical* solution of size at most k and (Q_I, Q_C) be an IC-partition of $G - S$. It follows from Observation 2 that $|P_I \cap Q_C| \leq 4$ and $|Q_I \cap P_C| \leq 4$. This observation leads to the following lemma.

▶ **Lemma 16.** *(G, k) is a* YES *instance of* Vertex $(2,2)$-Partization *if and only if there exist $V_C \subseteq P_I$ and $V_I \subseteq P_C$, each of cardinality at most 4 such that $X' = X \cup V_C \cup V_I$ can be partitioned into X'_I, X'_D, X'_C, with the following properties:*

1. *There is a set $Z_I \subseteq (P_I \setminus V_C) \cup X'_I$ such that $Z_I \cup X'_D \cup X'_C$ is an OCT for $G[P_I \cup X']$. In other words, Z_I is an OCT for $G[P_I \cup X'_I]$.*
2. *There is a set $Z_C \subseteq (P_C \setminus V_I) \cup X'_C$ such that $Z_C \cup X'_D \cup X'_I$ is an OCT for $\overline{G}[P_C \cup X']$. In other words, Z_C is an OCT for $\overline{G}[P_C \cup X'_C]$.*
3. *$|Z_I \cup Z_C \cup X'_D| \leq k$.*

Proof. Suppose (G, k) is a YES instance of Vertex $(2,2)$-Partization. Then there is a k-sized solution Z such that $G - Z$ is a $(2,2)$-graph. Let (Q_I, Q_C) be an IC-partition of $G - Z$. Let $V_C = P_I \cap Q_C$ and $V_I = P_C \cap Q_I$. It follows from Observation 2 that $|V_I| \leq 4$ and $|V_C| \leq 4$. Notice that any vertex in $P_I \setminus V_C$ either belongs to Q_I or to Z. Similarly, any vertex in $P_C \setminus V_I$ either belongs to Q_C or to Z. Let $X' = X \cup V_I \cup V_C$. Now we define $X'_I = X' \cap Q_I$, $X'_C = X' \cap Q_C$ and $X'_D = X' \cap Z$. Let $Z_I = Z \cap P_I$ and $Z_C = Z \cap P_C$. Note that $Z_I \cap V_C = \emptyset$ and $Z_C \cap V_I = \emptyset$. From the definition of X', V_I and V_C, it is clear that $V_I \subseteq X'_I$ and $V_C \subseteq X'_C$. Since $V_C \subseteq X'_I$ and $V_C \subseteq X'_C$, we have that $(P_I \cup X') \setminus (Z_I \cup X'_D \cup X'_C) = Q_I$. Also since, $G[Q_I]$ is a bipartite graph we have that $(Z_I \cup X'_D \cup X'_C)$ is an OCT of $G[P_I \cup X']$. By similar arguments we can show that $(Z_C \cup X'_D \cup X'_I)$ is an OCT of $\overline{G}[P_C \cup X']$. Since $Z_I \cup Z_C \cup X'_D = Z$ and $|Z| = k$, the set $Z_I \cup Z_C \cup X'_D$ satisfies condition 3 in the lemma. This completes the proof of the forward direction.

Conversely, suppose there is a $V_C \subseteq P_I$ and $V_I \subseteq P_C$, each of size at most 4 such that the $X' = X \cup V_I \cup V_C$ has a 3-partition $(X'_I \cup X'_D \cup X'_C)$ with the properties mentioned in the lemma. That is, there is an OCT Z_I for the graph $G[P_I \cup X'_I]$ and an OCT Z_C for the graph $\overline{G}[P_C \cup X'_C]$ such that $|Z_I \cup Z_C \cup X'_D| \leq k$. Then we claim that $Z = Z_I \cup Z_C \cup X'_D$ is a $(2,2)$-vertex deletion set of G. Consider the sets $Q_I = (P_I \cup X'_I) \setminus Z_I$ and $Q_C = (P_C \cup X'_C) \setminus Z_C$. By our assumption $G[Q_I]$ and $\overline{G}[Q_C]$ are bipartite graphs. Also note that $Q_I \cup Q_C \cup Z = V(G)$. Hence Z is a $(2,2)$-vertex deletion set of G and (Q_I, Q_C) is an IC-partition of $G - Z$. ◀

The Lemma 16 allows us to reduce an instance of Vertex $(2,2)$-Partization to polynomially many instances of a problem which is in NP. Consider the following problem.

Twin Odd Cycle Transversal (TOCT) **Parameter:** k
Input: Two graphs G_1 and G_2, terminals $X \subseteq V(G_1)$, $Y \subseteq V(G_2)$, a bijection Φ between X and Y, and an integer k
Question: Is there a partition of X into three parts (X_1, X_D, X_2) such that there is an OCT $Z_1 \subseteq V(G_1) \setminus (X_D \cup X_2)$ for the graph $G_1 - (X_D \cup X_2)$, an OCT $Z_2 \subseteq V(G_2) \setminus (\Phi(X_D) \cup \Phi(X_1))$ for the graph $G_2 - (\Phi(X_D) \cup \Phi(X_1))$ and $\lvert Z_1 \cup X_D \cup Z_2 \rvert \leq k$?

Clearly the problem TOCT is in NP. Because of Lemma 16, for each $V_C \subseteq P_1$ and $V_I \subseteq P_C$ of cardinality at most 4, we construct an instance of TOCT, of size bounded by a polynomial in k, using Lemma 15. After this, we fix a $V_I \subseteq P_C$ and a $V_C \subseteq P_I$, each of cardinality at most 4. Now let $X' = X \cup V_I \cup V_C$. Note that X' is a $(2,2)$-vertex deletion set of G and $(P_I \setminus V_C, P_C \setminus V_I)$ is an IC-partition of $G - X'$. The following observation is derived from the fact that $(P_I \setminus V_C, P_C \setminus V_I)$ is an IC-partition of $G - X'$ and $V_I \cup V_C \subseteq X'$.

▶ **Observation 17.** *The set X' is an OCT of $G[P_I \cup X']$ and also an OCT of $\overline{G}[P_C \cup X']$.*

For a particular choice of $V_C \subseteq P_I$ and $V_I \subseteq P_C$ of cardinality at most 4, we construct an instance of TOCT as follows. Let $X' = X \cup V_I \cup C_C$, where X is the approximate solution of size bounded by $\mathcal{O}(k^{3/2})$. Let (P_I, P_C) be an IC-partition of $G - X$. Let $G_1 = G[P_I \cup X']$ and $G_2 = \overline{G}[P_C \cup X']$. By Observation 17, X' is an OCT in graphs G_1 and G_2. Now we

apply Lemma 15 and get a set of relevant vertices $Z_1 \subseteq V(G_1)$ of size bounded by $\mathcal{O}(k^{9/2})$. Next, we construct a graph G_1^* as follows: delete all the vertices $V(G_1) \setminus (X' \cup Z_1)$ from G_1. Add two length (three length) path between two vertices in $V(G_1^*)$, if there is an even length (odd length) path between the corresponding vertices in G_1 using only vertices from $V(G) \setminus (X' \cup Z_1)$. Similarly, we construct a graph G_2^* from G_2. Now we output $H = (G_1, G_2, X', X', k)$ as the reduced instance of TOCT, with the bijection between X' and X' be the natural identity map. Since there are $\mathcal{O}(n^4)$ choices for selecting V_C and V_I, our algorithm will output instances $H_1, H_2, \ldots H_t$ where $t = \mathcal{O}(n^4)$ and the size of each H_i is bounded by $\mathcal{O}(k^9)$.

Using Lemmata 15 and 16 we can prove that in fact the above Turing reduction is correct.

▶ **Lemma 18** (⋆). *(G, k) is a* YES *instance of* Vertex $(2,2)$-Partization *if and only if there exists i such that H_i is a* YES *instance of* TOCT.

The problem TOCT is in NP and Vertex $(2,1)$-Partization is NP-complete. Therefore, by Cook-Levin theorem each instance H_i of TOCT can be reduced to an an instance of Vertex $(2,2)$-Partization in polynomial time. Also note that size of each instance H_i is bounded by $\mathcal{O}(k^9)$. Thus we have the following theorem.

▶ **Theorem 19.** *There exists a randomized polynomial Turing kernel for* Vertex $(2,2)$-Partization.

Since there is parameter preserving reduction from Vertex $(2,1)$-Partization and Vertex $(1,2)$-Partization to Vertex $(2,2)$-Partization, the following corollary is derived from Theorem 19.

▶ **Corollary 20.** *There exists a randomized polynomial Turing kernel for* Vertex $(2,1)$-Partization *and* Vertex $(1,2)$-Partization.

6 Edge Deletion to (r, ℓ)-graphs

In this section we show that Edge $(2,1)$-Partization and Edge $(1,2)$-Partization are in FPT.

6.1 Edge $(2,1)$-Partization

In this subsection we show that Edge $(2,1)$-Partization is in FPT, using iterative compression. For Edge $(2,1)$-Partization, the corresponding compression problem is defined as follows.

Edge $(2,1)$-Partization Compression	**Parameter:** k
Input: A graph G with $V(G) = V \cup \{v\}$, an integer k and an edge set $S' \subseteq E(G - \{v\})$, of size at most k, such that $G[V] - S'$ is a $(2,1)$-graph	
Output: A subset $S \subseteq E$ of size at most k such that $G - S$ is a $(2,1)$-graph	

Similar to Vertex $(2,2)$-Partization, we can show that Edge $(2,1)$-Partization can be solved, by running Edge $(2,1)$-Partization Compression at most $|V(G)|$ times, for an input instance (G, k). The following lemma is useful for our purpose.

▶ **Lemma 21** (⋆). *Let G be a graph on n vertices, $v \in V(G)$ and $|E(G - \{v\})| \leq k$. Then the number of cliques in G is bounded by $2^{\mathcal{O}(\sqrt{k})}n$ and these cliques can be enumerated in time $2^{\mathcal{O}(\sqrt{k})}n$.*

Next we show that Edge $(2,1)$-Partization Compression is in FPT.

▶ **Lemma 22** ($\star$). *Edge $(2,1)$-Partization Compression is solved in time $2^{k+o(k)}n^{\mathcal{O}(1)}$.*

Thus, by using Lemma 22, we prove the following theorem.

▶ **Theorem 23.** *Edge $(2,1)$-Partization can be solved in time $2^{k+o(k)}n^{\mathcal{O}(1)}$.*

6.2 Edge $(1,2)$-Partization

In this subsection we show that Edge $(1,2)$-Partization is in FPT. Again we use the iterative compression technique to solve the problem. For our algorithm, we need an algorithm for a version of Odd Cycle Transversal. Let $\mathbb{G}$ be an hereditary graph class (hereditary means that if $G \in \mathbb{G}$, then every induced subgraph of G is in $\mathbb{G}$ as well) and $\mathbb{G}$ is decidable. Then the problem $\mathbb{G}$-Weighted Bipartition is defined as follows.

$\mathbb{G}$-Weighted Bipartition	**Parameter:** $k+W$
Input: A graph G, $w : V(G) \to \mathbb{N}^+$ and integers k and W	
Output: An OCT O of G, of size at most k such that $w(O) \leq W$ and $G[O] \in \mathbb{G}$	

Marx et al. [23] showed that the unweighted version of the problem, $\mathbb{G}$-Bipartition can be solved in FPT time. The proof by Marx et al., constructs an "equivalent graph" with treewidth bounded by a function of k. The problem is then solved in the equivalent graph, using Courcelle's theorem [6] by expressing the problem as an MSO predicate. Since we can express whether the weight of a subset of vertices is at most W using an MSO predicate of length bounded by a function of W, the following theorem follows from the results of Marx et al. [23].

▶ **Theorem 24.** *If $\mathbb{G}$ is hereditary and decidable, then $\mathbb{G}$-Weighted Bipartition is in* FPT.

Now we are ready to define compression version of the problem Edge $(1,2)$-Partization and prove that it is in FPT, which in turn will imply that non-compression version of the problem is in FPT.

Edge $(1,2)$-Partization Compression	**Parameter:** k
Input: A Graph G with $V(G) = V \cup \{v\}$, an integer k and an edge set $S' \subseteq E(G-v)$, of size at most k, such that $G[V] - S'$ is a $(1,2)$-graph	
Output: A subset $S \subseteq E$ of size at most k such that $G - S$ is a $(1,2)$-graph	

▶ **Lemma 25** ($\star$). *Edge $(1,2)$-Partization Compression is in* FPT.

Thus by using Lemma 25, we get the following theorem.

▶ **Theorem 26.** *Edge $(1,2)$-Partization is in* FPT.

7 Conclusion

In this paper we explored parameterized complexity of a family of partition problems, namely Vertex (r,ℓ)-Partization and Edge (r,ℓ)-Partization. Whether there exists a polynomial kernel for these problems remains an interesting open problem. Also, the parameterized complexity of Edge $(2,2)$-Partization remains unresolved.

References

1 Amit Agarwal, Moses Charikar, Konstantin Makarychev, and Yury Makarychev. $O(\sqrt{\log n})$ approximation algorithms for min uncut, min 2cnf deletion, and directed cut problems. In *STOC*, pages 573–581, 2005.

2 Andreas Brandstädt, Van Bang Le, and Thomas Szymczak. The complexity of some problems related to graph 3-colorability, 1998.

3 Chandra Chekuri, editor. *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*. SIAM, 2014.

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

5 Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC*, pages 151–158, New York, NY, USA, 1971. ACM.

6 Bruno Courcelle. The monadic second-order logic of graphs. I. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.

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 R. Diestel. *Graph Theory*. Springer, Berlin, second ed., electronic edition, February 2000.

9 Tomás 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 J. Flum and M. Grohe. *Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

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 Jiong Guo, Jens Gramm, Falk Hüffner, Rolf Niedermeier, and Sebastian Wernicke. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *J. Comput. Syst. Sci.*, 72(8):1386–1396, 2006.

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

16 Falk Hüffner. Algorithm engineering for optimal graph bipartization. *J. Graph Algorithms Appl.*, 13(2):77–98, 2009.

17 Yoichi Iwata, Keigo Oka, and Yuichi Yoshida. Linear-time FPT algorithms via network flow. In Chekuri [3], pages 1749–1761.

18 Stefan Kratsch and Magnus Wahlström. Representative sets and irrelevant vertices: New tools for kernelization. In *FOCS*, pages 450–459, 2012.

19 Stefan Kratsch and Magnus Wahlström. Compression via matroids: A randomized polynomial kernel for odd cycle transversal. *ACM Transactions on Algorithms*, 10(4):20, 2014.

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

21 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.

22 Daniel Lokshtanov, Saket Saurabh, and Somnath Sikdar. Simpler parameterized algorithm for oct. In Jirí Fiala, Jan Kratochvíl, and Mirka Miller, editors, *IWOCA*, volume 5874 of *Lecture Notes in Computer Science*, pages 380–384. Springer, 2009.

23 Dániel Marx, Barry O'Sullivan, and Igor Razgon. Finding small separators in linear time via treewidth reduction. *ACM Transactions on Algorithms*, 9(4):30, 2013.

24 M. S. Ramanujan and Saket Saurabh. Linear time parameterized algorithms via skew-symmetric multicuts. In Chekuri [3], pages 1739–1748.

25 Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Oper. Res. Lett.*, 32(4):299–301, 2004.