



HAL
open science

Finding Cuts of Bounded Degree: Complexity, FPT and Exact Algorithms, and Kernelization

Guilherme C. M. Gomes, Ignasi Sau

► **To cite this version:**

Guilherme C. M. Gomes, Ignasi Sau. Finding Cuts of Bounded Degree: Complexity, FPT and Exact Algorithms, and Kernelization. IPEC 2019 - 14th International Symposium on Parameterized and Exact Computation, Sep 2019, Munich, Germany. pp.19:1–19:15, 10.4230/LIPIcs.IPEC.2019.19.lirmm-02410703

HAL Id: lirmm-02410703

<https://hal-lirmm.ccsd.cnrs.fr/lirmm-02410703>

Submitted on 13 Dec 2019


HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License

Finding Cuts of Bounded Degree: Complexity, FPT and Exact Algorithms, and Kernelization

Guilherme C. M. Gomes 

Universidade Federal de Minas Gerais, Departamento de Ciência da Computação,
Belo Horizonte, Brazil
LIRMM, Université de Montpellier, Montpellier, France
gcm.gomes@dcc.ufmg.br

Ignasi Sau 

CNRS, LIRMM, Université de Montpellier, Montpellier, France
ignasi.sau@lirmm.fr

Abstract

A *matching cut* is a partition of the vertex set of a graph into two sets A and B such that each vertex has at most one neighbor in the other side of the cut. The MATCHING CUT problem asks whether a graph has a matching cut, and has been intensively studied in the literature. Motivated by a question posed by Komusiewicz et al. [IPEC 2018], we introduce a natural generalization of this problem, which we call d -CUT: for a positive integer d , a d -*cut* is a bipartition of the vertex set of a graph into two sets A and B such that each vertex has at most d neighbors across the cut. We generalize (and in some cases, improve) a number of results for the MATCHING CUT problem. Namely, we begin with an NP-hardness reduction for d -CUT on $(2d+2)$ -regular graphs and a polynomial algorithm for graphs of maximum degree at most $d+2$. The degree bound in the hardness result is unlikely to be improved, as it would disprove a long-standing conjecture in the context of internal partitions. We then give FPT algorithms for several parameters: the maximum number of edges crossing the cut, treewidth, distance to cluster, and distance to co-cluster. In particular, the treewidth algorithm improves upon the running time of the best known algorithm for MATCHING CUT. Our main technical contribution, building on the techniques of Komusiewicz et al. [IPEC 2018], is a polynomial kernel for d -CUT for every positive integer d , parameterized by the distance to a cluster graph. We also rule out the existence of polynomial kernels when parameterizing simultaneously by the number of edges crossing the cut, the treewidth, and the maximum degree. Finally, we provide an exact exponential algorithm slightly faster than the naive brute force approach running in time $\mathcal{O}^*(2^n)$.

2012 ACM Subject Classification Theory of computation \rightarrow Parameterized complexity and exact algorithms; Mathematics of computing \rightarrow Matchings and factors

Keywords and phrases matching cut, bounded degree cut, parameterized complexity, FPT algorithm, polynomial kernel, distance to cluster

Digital Object Identifier 10.4230/LIPIcs.IPEC.2019.19

Related Version The full version of this article is permanently available at <https://arxiv.org/abs/1905.03134>.

Funding *Guilherme C. M. Gomes*: Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Ignasi Sau: Projects DEMOGRAPH (ANR-16-CE40-0028) and ESIGMA (ANR-17-CE23-0010).

1 Introduction

A *cut* of a graph $G = (V, E)$ is a bipartition of its vertex set $V(G)$ into two non-empty sets, denoted by (A, B) . The set of all edges with one endpoint in A and the other in B is the *edge cut*, or the set of *crossing edges*, of (A, B) . A *matching cut* is a (possibly empty) edge cut that is a matching, that is, such that its edges are pairwise vertex-disjoint. Equivalently,



© Guilherme C. M. Gomes and Ignasi Sau;
licensed under Creative Commons License CC-BY

14th International Symposium on Parameterized and Exact Computation (IPEC 2019).

Editors: Bart M. P. Jansen and Jan Arne Telle; Article No. 19; pp. 19:1–19:15

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

(A, B) is a matching cut of G if and only if every vertex is incident to at most one crossing edge of (A, B) [7, 15], that is, it has at most one neighbor across the cut.

Motivated by an open question posed by Komusiewicz et al. [18] during the presentation of their article, we investigate a natural generalization that arises from this alternative definition, which we call d -cut. Namely, for a positive integer $d \geq 1$, a d -cut is a cut (A, B) such that each vertex has at most d neighbors across the partition, that is, every vertex in A has at most d neighbors in B , and vice-versa. Note that a 1-cut is a matching cut. As expected, not every graph admits a d -cut, and the d -CUT problem is the problem of deciding, for a fixed integer $d \geq 1$, whether or not an input graph G has a d -cut.

When $d = 1$, we refer to the problem as MATCHING CUT. Graphs with no matching cut first appeared in Graham’s manuscript [15] under the name of *indecomposable graphs*, presenting some examples and properties of decomposable and indecomposable graphs, leaving their recognition as an open problem. In answer to Graham’s question, Chvátal [7] proved that the problem is NP-hard for graphs of maximum degree at least four and polynomially solvable for graphs of maximum degree at most three; in fact, as shown by Moshi [24], every graph of maximum degree three and at least eight vertices has a matching cut.

Chvátal’s results spurred a lot of research on the complexity of the problem [1, 5, 18–21, 25]. In particular, Bonsma [5] showed that MATCHING CUT remains NP-hard for planar graphs of maximum degree four and for planar graphs of girth five; Le and Randerath [21] gave an NP-hardness reduction for bipartite graphs of maximum degree four; Le and Le [20] proved that MATCHING CUT is NP-hard for graphs of diameter at least three, and presented a polynomial-time algorithm for graphs of diameter at most two. Beyond planar graphs, Bonsma’s work [5] also proves that the matching cut property is expressible in monadic second order logic and, by Courcelle’s Theorem [8], it follows that MATCHING CUT is FPT when parameterized by the treewidth of the input graph; he concludes with a proof that the problem admits a polynomial-time algorithm for graphs of bounded cliquewidth.

Kratsch and Le [19] noted that Chvátal’s original reduction also shows that, unless the Exponential Time Hypothesis [16] (ETH) fails¹, there is no algorithm solving MATCHING CUT in time $2^{o(n)}$ on n -vertex input graphs. Also in [19], the authors provide a first branching algorithm, running² in time $\mathcal{O}^*(2^{n/2})$, a single-exponential FPT algorithm when parameterized by the vertex cover number $\tau(G)$, and an algorithm generalizing the polynomial cases of line graphs [24] and claw-free graphs [5]. Kratsch and Le [19] also asked for the existence a single-exponential algorithm parameterized by treewidth. In response, Aravind et al. [1] provided a $\mathcal{O}^*(12^{\text{tw}(G)})$ algorithm for MATCHING CUT using nice tree decompositions, along with FPT algorithms for other structural parameters, namely neighborhood diversity, twin-cover, and distance to split graph.

The natural parameter – the number of edges crossing the cut – has also been considered. Indeed, Marx et al. [23] tackled the STABLE CUTSET problem, to which MATCHING CUT can be easily reduced via the line graph, and through a breakthrough technique showed that this problem is FPT when parameterized by the maximum size of the stable cutset. Recently, Komusiewicz et al. [18] improved on the results of Kratsch and Le [19], providing an exact exponential algorithm for MATCHING CUT running in time $\mathcal{O}^*(1.3803^n)$, as well as FPT algorithms parameterized by the distance to a cluster graph and the distance to a co-cluster graph, which improve the algorithm parameterized by the vertex cover number, since both parameters are easily seen to be smaller than the vertex cover number. For the distance

¹ The ETH states that 3-SAT on n variables cannot be solved in time $2^{o(n)}$; see [16] for more details.

² The $\mathcal{O}^*(\cdot)$ notation suppresses factors that are bounded by a polynomial in the input size.

to cluster parameter, they also presented a quadratic kernel; while for a combination of treewidth, maximum degree, and number of crossing edges, they showed that no polynomial kernel exists unless $\text{NP} \subseteq \text{coNP/poly}$.

A problem closely related to d -CUT is that of INTERNAL PARTITION, first studied by Thomassen [28]. In this problem, we seek a bipartition of the vertices of an input graph such that every vertex has at least as many neighbors in its own part as in the other part. Such a partition is called an *internal partition*. Usually, the problem is posed in a more general form: given functions $a, b : V(G) \rightarrow \mathbb{Z}_+$, we seek a bipartition (A, B) of $V(G)$ such that every $v \in A$ satisfies $\text{deg}_A(v) \geq a(v)$ and every $u \in B$ satisfies $\text{deg}_B(u) \geq b(u)$, where $\text{deg}_A(v)$ denotes the number of neighbors of v in the set A . Such a partition is called an (a, b) -*internal partition*. Originally, Thomassen asked in [28] whether for any pair of positive integers s, t , a graph G with $\delta(G) \geq s + t + 1$ has a vertex bipartition (A, B) with $\delta(G[A]) \geq s$ and $\delta(G[B]) \geq t$, where $\delta(H)$ is the minimum degree of H . Stiebitz [27] answered that, in fact, for any graph G and any pair of functions $a, b : V(G) \rightarrow \mathbb{Z}_+$ satisfying $\text{deg}(v) \geq a(v) + b(v) + 1$ for every $v \in V(G)$, G has an (a, b) -internal partition; see [17, 22] for follow-up results. It is conjectured that, for every positive integer r , there exists some constant n_r for which every r -regular graph with more than n_r vertices has an internal partition [2, 10] (the conjecture for r even appeared first in [26]). The cases $r \in \{3, 4\}$ have been settled by Shafique and Dutton [26]; the case $r = 6$ has been verified by Ban and Linial [2]. This latter result implies that every 6-regular graph of sufficiently large size has a 3-cut.

Our results. We aim at generalizing several of the previously reported results for MATCHING CUT. First, we show in Section 2, by using a reduction inspired by Chvátal's [7], that for every $d \geq 1$, d -CUT is NP-hard even when restricted to $(2d + 2)$ -regular graphs and that, if $\Delta(G) \leq d + 2$ (the maximum degree of G) finding a d -cut can be done in polynomial time. The degree bound in the NP-hardness result is unlikely to be improved: if we had an NP-hardness result for d -CUT restricted to $(2d + 1)$ -regular graphs, this would disprove the conjecture about the existence of internal partitions on r -regular graphs [2, 10, 26] for r odd, unless $P = \text{NP}$. We conclude the section by giving a simple exact exponential algorithm that, for every $d \geq 1$, runs in time $\mathcal{O}^*(c_d^n)$ for some constant $c_d < 2$, hence improving over the trivial brute-force algorithm running in time $\mathcal{O}^*(2^n)$.

We then proceed to analyze the problem in terms of its parameterized complexity. Section 3 begins with a proof, using the treewidth reduction technique of Marx et al. [23], that d -CUT is FPT parameterized by the maximum number of edges crossing the cut. Afterwards, we present a dynamic programming algorithm for d -CUT parameterized by treewidth running in time $\mathcal{O}^*(2^{\text{tw}(G)}(d + 1)^{2\text{tw}(G)})$; in particular, for $d = 1$ this algorithm runs in time $\mathcal{O}^*(8^{\text{tw}(G)})$ and improves the one given by Aravind et al. [1] for MATCHING CUT, which runs in $\mathcal{O}^*(12^{\text{tw}(G)})$ time. By employing the cross-composition framework of Bodlaender et al. [4] and using a reduction similar to the one in [18], we show that, unless $\text{NP} \subseteq \text{coNP/poly}$, there is no polynomial kernel for d -CUT parameterized simultaneously by the number of crossing edges, the maximum degree, and the treewidth of the input graph. We then present a polynomial kernel and an FPT algorithm when parameterizing by the distance to cluster, denoted by $\text{dc}(G)$. This polynomial kernel is our main technical contribution, and it is strongly inspired by the technique presented by Komusiewicz et al. [18] for MATCHING CUT. Finally, we give an FPT algorithm parameterized by the distance to co-cluster, denoted by $\text{d}\bar{\text{c}}(G)$. These results imply the existence of a polynomial kernel for d -CUT parameterized by the vertex cover number $\tau(G)$. We present in Section 4 our concluding remarks and some open questions.

We use standard notation from graph theory and parameterized complexity; see [9, 11–13] for any undefined terminology. Due to space limitations, the proofs of the results marked with ‘(★)’ can be found in the full version of this article, permanently available at <https://arxiv.org/abs/1905.03134>. Some basic preliminaries can also be found there.

1.1 Preliminaries

We use standard graph-theoretic notation, and we consider simple undirected graphs without loops or multiple edges; see [11] for any undefined terminology. When the graph is clear from the context, the degree (that is, the number of neighbors) of a vertex v is denoted by $\deg(v)$, and the number of neighbors of a vertex v in a set $A \subseteq V(G)$ is denoted by $\deg_A(v)$. The minimum degree, the maximum degree, the line graph, and the vertex cover number of a graph G are denoted by $\delta(G)$, $\Delta(G)$, $L(G)$, and $\tau(G)$, respectively. For a positive integer $k \geq 1$, we denote by $[k]$ the set containing every integer i such that $1 \leq i \leq k$.

We refer the reader to [9, 12] for basic background on parameterized complexity, and we recall here only some basic definitions. A *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$. For an instance $I = (x, k) \in \Sigma^* \times \mathbb{N}$, k is called the *parameter*. A parameterized problem is *fixed-parameter tractable* (FPT) if there exists an algorithm \mathcal{A} , a computable function f , and a constant c such that given an instance $I = (x, k)$, \mathcal{A} (called an *FPT algorithm*) correctly decides whether $I \in L$ in time bounded by $f(k) \cdot |I|^c$.

A fundamental concept in parameterized complexity is that of *kernelization*; see [13] for a recent book on the topic. A kernelization algorithm, or just *kernel*, for a parameterized problem Π takes an instance (x, k) of the problem and, in time polynomial in $|x| + k$, outputs an instance (x', k') such that $|x'|, k' \leq g(k)$ for some function g , and $(x, k) \in \Pi$ if and only if $(x', k') \in \Pi$. The function g is called the *size* of the kernel and may be viewed as a measure of the “compressibility” of a problem using polynomial-time preprocessing rules. A kernel is called *polynomial* (resp. *quadratic*, *linear*) if the function $g(k)$ is a polynomial (resp. quadratic, linear) function in k . A breakthrough result of Bodlaender et al. [3] gave the first framework for proving that certain parameterized problems do not admit polynomial kernels, by establishing so-called *composition algorithms*. Together with a result of Fortnow and Santhanam [14] this allows to exclude polynomial kernels under the assumption that $\text{NP} \not\subseteq \text{coNP/poly}$, otherwise implying a collapse of the polynomial hierarchy to its third level [29].

2 NP-hardness, polynomial cases, and exact exponential algorithm

In this section we focus on the classical complexity of the d -CUT problem, and on exact exponential algorithms.

Chvátal [7] proved that MATCHING CUT is NP-hard for graphs of maximum degree at least four. In the next theorem, whose proof is inspired by the reduction of Chvátal [7] from 3-UNIFORM HYPERGRAPH BICOLORING, we prove the NP-hardness of d -CUT for $(2d + 2)$ -regular graphs. In particular, for $d = 1$ it implies the NP-hardness of MATCHING CUT for 4-regular graphs, which is a strengthening of Chvátal’s [7] hardness proof.

► **Theorem 1 (★).** *For every integer $d \geq 1$, d -CUT is NP-hard even when restricted to $(2d + 2)$ -regular graphs.*

The graphs constructed by Theorem 1 are neither planar nor bipartite, but they are regular, a result that we were unable to find in the literature for MATCHING CUT. Note that every planar graph has a d -cut for every $d \geq 5$, so only the cases $d \in \{2, 3, 4\}$ remain open,

as the case $d = 1$ is known to be NP-hard [5]. Concerning graphs of bounded diameter, Le and Le [20] prove the NP-hardness of MATCHING CUT for graphs of diameter at least three by reducing MATCHING CUT to itself. It can be easily seen that the same construction given by Le and Le [20], but reducing d -CUT to itself, also proves the NP-hardness of d -CUT for every $d \geq 1$.

► **Corollary 2.** *For every integer $d \geq 1$, d -CUT is NP-hard for graphs of diameter at least three.*

We leave as an open problem to determine whether there exists a polynomial-time algorithm for d -CUT for graphs of diameter at most two for every $d \geq 2$, as it is the case for $d = 1$ [20].

We now turn to cases that can be solved in polynomial time. Our next result is a natural generalization of Chvátal’s algorithm [7] for MATCHING CUT on graphs of maximum degree three.

► **Theorem 3** (\star). *For any graph G and integer $d \geq 1$ such that $\Delta(G) \leq d + 2$, it can be decided in polynomial time if G has a d -cut. Moreover, for $d = 1$ any graph G with $\Delta(G) \leq 3$ and $|V(G)| \geq 8$ has a matching cut, for $d = 2$ any graph G with $\Delta(G) \leq 4$ and $|V(G)| \geq 6$ has a 2-cut, and for $d \geq 3$ any graph G with $\Delta(G) \leq d + 2$ has a d -cut.*

Theorems 1 and 3 present a “quasi-dichotomy” for d -cut on graphs of bounded maximum degree. Specifically, for $\Delta(G) \in \{d + 3, \dots, 2d + 1\}$, the complexity of the problem remains unknown. However, we believe that most, if not all, of these open cases can be solved in polynomial time; see the discussion in Section 4.

To conclude this section, we present a simple exact exponential algorithm which, for every $d \geq 1$, runs in time $\mathcal{O}^*(c_d^n)$ for some constant $c_d < 2$. For the case $d = 1$, the currently known algorithms [18, 19] exploit structures that appear to get out of control when d increases, and so has a better running time than the one described below.

► **Theorem 4** (\star). *For every fixed integer $d \geq 1$ and n -vertex graph G , there is an algorithm that solves d -CUT in time $\mathcal{O}^*((c_d)^n)$, for some constant $1 < c_d < 2$.*

3 Parameterized algorithms and kernelization

In this section we focus on the parameterized complexity of d -CUT. More precisely, in Section 3.1 we consider as the parameter the number of edges crossing the cut and in Section 3.2 the distance to cluster (in particular, we provide a quadratic kernel). The FPT algorithms parameterized by treewidth and the distance to co-cluster can be found in the full version of the paper.

Before proceeding, we introduce the notion of *monochromatic sets*.

► **Definition 5.** *A set of vertices $X \subseteq V(G)$ is said to be monochromatic if, for any d -cut (A, B) of G , $X \subseteq A$ or $X \subseteq B$. A subgraph H of G is monochromatic if $V(H)$ is monochromatic.*

3.1 Crossing edges

In this section we consider as the parameter the maximum number of edges crossing the cut. In a nutshell, our approach is to use as a black box one of the algorithms presented by Marx et al. [23] for a class of separation problems. Their fundamental problem is \mathcal{G} -MINCUT, for a fixed class of graphs \mathcal{G} , which we state formally, along with their main result, below.

\mathcal{G} -MINCUT

Instance: A graph G , vertices s, t , and an integer k .

Parameter: The integer k .

Question: Is there an induced subgraph H of G with at most k vertices such that $H \in \mathcal{G}$ and H is an $s - t$ separator?

► **Theorem 6** (Theorem 3.1 in [23]). *If \mathcal{G} is a decidable and hereditary graph class, \mathcal{G} -MINCUT is FPT.*

To be able to apply Theorem 6, we first need to specify a graph class to which, on the line graph, our separators correspond. We must also be careful to guarantee that the removal of a separator in the line graph leaves non-empty components in the input graph. To accomplish the latter, for each $v \in V(G)$, we add a private clique of size $2d$ adjacent only to it, choose one arbitrary vertex v' in each of them. The algorithm asks, for each pair v', u' , whether or not a “special” separator of the appropriate size between v' and u' exists. We assume henceforth that these private cliques have been added to the input graph G . For each integer $d \geq 1$, we define the graph class \mathcal{G}_d as follows.

► **Definition 7.** *A graph H belongs to \mathcal{G}_d if and only if its maximum clique size is at most d .*

Note that \mathcal{G}_d is clearly decidable and hereditary for every integer $d \geq 1$.

► **Lemma 8** (*). *G has a d -cut separating v' and u' if and only if the line graph of G has a vertex separator belonging to \mathcal{G}_d that separates e_v and e_u , where e_v corresponds to the edge $vv' \in E(G)$ and e_u to the edge $uu' \in E(G)$.*

► **Theorem 9.** *For every $d \geq 1$, there is an FPT algorithm for d -CUT parameterized by k , the maximum number of edges crossing the cut.*

Proof. For each pair of vertices $s, t \in V(G)$ that do not belong to the private cliques, our goal is to find a subset of vertices $S \subseteq V(L(G))$ of size at most k that separates s and t such that $L(G)[S] \in \mathcal{G}_d$. This is precisely what is provided by Theorem 6, and the correctness of this approach is guaranteed by Lemma 8. Since we perform a quadratic number of calls to the algorithm given by Theorem 6, our algorithm still runs in FPT time. ◀

As to the running time of the FPT algorithm given by Theorem 9, the treewidth reduction technique of [23] relies on the construction of a monadic second order logic (MSOL) expression and Courcelle’s Theorem [8] to guarantee fixed-parameter tractability, and therefore it is hard to provide an explicit running time in terms of k .

3.2 Kernelization and distance to cluster

The proof of the following theorem consists of a simple generalization to every $d \geq 1$ of the construction given by Komusiewicz et al. [18] for $d = 1$.

► **Theorem 10.** *For any fixed $d \geq 1$, d -CUT does not admit a polynomial kernel when simultaneously parameterized by k , Δ , and $\text{tw}(G)$, unless $\text{NP} \subseteq \text{coNP/poly}$.*

Proof. We show that the problem cross-composes into itself. Start with t instances G_1, \dots, G_t of d -CUT. First, pick an arbitrary vertex $v_i \in V(G_i)$, for each $i \in [t]$. Second, for $i \in [t - 1]$, add a copy of K_{2d} , call it $K(i)$, every edge between v_i and $K(i)$, and every edge between $K(i)$ and v_{i+1} . This concludes the construction of G , which for $d = 1$ coincides with that presented by Komusiewicz et al. [18].

Suppose that (A, B) is a d -cut of some G_i and that $v_i \in A$. Note that $(V(G) \setminus B, B)$ is a d -cut of G since the only edges in the cut are those between A and B . For the converse, take some d -cut (A, B) of G and note that every vertex in the set $\{v_t\} \cup_{i \in [t-1]} \{v_i\} \cup K(i)$ is contained in the same side of the partition, say A . Since $B \neq \emptyset$, there is some i such that $B \cap V(G_i) \neq \emptyset$, which implies that there is some i (possibly more than one) such that $(A \cap V(G_i), B \cap V(G_i))$ must be a d -cut of G_i .

That the treewidth, maximum degree, and number of edges crossing the partition are bounded by n , the maximum number of vertices of the graphs G_i , is a trivial observation. ◀

We now proceed to show that d -CUT admits a polynomial kernel when parameterizing by the *distance to cluster* parameter, denoted by dc . A *cluster graph* is a graph such that every connected component is a clique; the *distance to cluster* of a graph G is the minimum number of vertices we must remove from G to obtain a cluster graph. Our results are heavily inspired by the work of Komusiewicz et al. [18]. Indeed, most of our reduction rules are natural generalizations of theirs. However, we need some extra observations and rules that only apply for $d \geq 2$, such as Rule 8.

We denote by $U = \{U_1, \dots, U_t\}$ a set of vertices such that $G - U$ is a cluster graph, and each U_i is called a *monochromatic part* or *monochromatic set* of U , and we will maintain the invariant that these sets are indeed monochromatic. Initially, we set each U_i as a singleton. In order to simplify the analysis of our instance, for each U_i of size at least two, we will have a private clique of size $2d$ adjacent to every vertex of U_i , which we call X_i . The *merge* operation between U_i and U_j is the following modification: delete $X_i \cup X_j$, set U_i as $U_i \cup U_j$, U_j as empty, and add a new clique of size $2d$, $X_{i,j}$, which is adjacent to every element of the new U_i . We say that an operation is *safe* if the resulting instance is a YES instance if and only if the original instance was.

► **Observation 1.** *If $U_i \cup U_j$ is monochromatic, merging U_i and U_j is safe.*

It is worth mentioning that the second case of the following rule is not needed in the corresponding rule in [18]; we need it here to prove the safeness of Rules 7 and 8.

► **Reduction Rule 1.** *Suppose that $G - U$ has some cluster C such that*

1. $(C, V(G) \setminus C)$ is a d -cut, or
2. $|C| \leq 2d$ and there is $C' \subseteq C$ such that $(C', V(G) \setminus C')$ is a d -cut.

Then output YES.

After applying Rule 1, for every cluster C , C has some vertex with at least $d+1$ neighbors in U , or there is some vertex of U with at least $d+1$ neighbors in C . Moreover, note that no cluster C with at least $2d+1$ vertices can be partitioned in such a way that one side of the cut is composed only by a proper subset of vertices of C , i.e., C is monochromatic

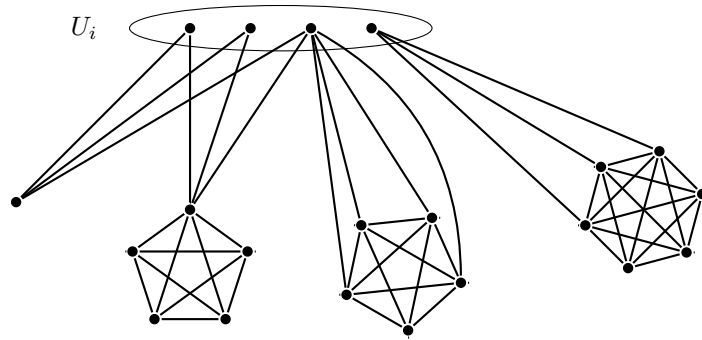
The following definition is a natural generalization of the definition of the set N^2 given by Komusiewicz et al. [18]. Essentially, it enumerates some of the cases where a vertex, or set of vertices, is monochromatic, based on its relationship with U . However, there is a crucial difference that keeps us from achieving equivalent bounds both in terms of running time and size of the kernel, and which makes the analysis and some of the rules more complicated than in [18]. Namely, for a vertex to be forced into a particular side of the cut, it must have at least $d+1$ neighbors in that side; moreover, a vertex of U being adjacent to $2d$ vertices of a cluster C implies that C is monochromatic. Only if $d = 1$, i.e., when we are dealing with matching cuts, the equality $d+1 = 2d$ holds. This gap between $d+1$ and $2d$ is the main difference between our kernelization algorithm for general d and the one shown in [18] for MATCHING CUT, and the main source of the differing complexities we obtain. In particular,

19:8 Finding Cuts of Bounded Degree

for $d = 1$ the fourth case of the following definition is a particular case of the third one, but this is not true anymore for $d \geq 2$. Figure 1 illustrates the set of vertices introduced in Definition 11.

► **Definition 11.** For a monochromatic part $U_i \subseteq U$, let $N^{2d}(U_i)$ be the set of vertices $v \in V(G) \setminus U$ for which at least one of the following holds:

1. v has at least $d + 1$ neighbors in U_i .
2. v is in a cluster C of size at least $2d + 1$ in $G - U$ such that there is some vertex of C with at least $d + 1$ neighbors in U_i .
3. v is in a cluster C of $G - U$ and some vertex in U_i has $2d$ neighbors in C .
4. v is in a cluster C of $G - U$ of size at least $2d + 1$ and some vertex in U_i has $d + 1$ neighbors in C .



■ **Figure 1** The four cases that define membership in $N^{2d}(U_i)$ for $d = 2$, from left to right.

► **Observation 2.** For every monochromatic part U_i , $U_i \cup N^{2d}(U_i)$ is monochromatic.

The next rules aim to increase the size of monochromatic sets. In particular, Rule 2 translates the transitivity of the monochromatic property, while Rule 3 identifies a case where merging the monochromatic sets is inevitable.

► **Reduction Rule 2.** If $N^{2d}(U_i) \cap N^{2d}(U_j) \neq \emptyset$, merge U_i and U_j .

► **Reduction Rule 3.** If there is a set of $2d + 1$ vertices $L \subseteq V(G)$ with two common neighbors u, u' such that $u \in U_i$ and $u' \in U_j$, merge U_i and U_j .

Proof of safeness of Rule 3. Suppose that in some d -cut (A, B) , $u \in A$ and $u' \in B$, this implies that at most d elements of L are in A and at most d are in B , which is impossible since $|L| = 2d + 1$. ◀

We say that a cluster is *small* if it has at most $2d$ vertices, and *big* otherwise. Moreover, a vertex in a cluster is *ambiguous* if it has neighbors in more than one U_i . A cluster is *ambiguous* if it has an ambiguous vertex, and *fixed* if it is contained in some $N^{2d}(U_i)$.

► **Observation 3.** If G is reduced by Rule 1, every big cluster is ambiguous or fixed.

Proof. Since Rule 1 cannot be applied, every cluster C has either one vertex v with at least $d + 1$ neighbors in U or there is some vertex of a set U_i with $d + 1$ neighbors in C . In the latter case, by applying the fourth case in the definition of $N^{2d}(U_i)$, we conclude that C is fixed. In the former case, either v has $d + 1$ neighbors in the same U_i , in which case C is fixed, or its neighborhood is spread across multiple monochromatic sets, and so v and, consequently, C are ambiguous. ◀

Our next goal is to bound the number of vertices outside of U .

► **Reduction Rule 4.** *If there are two clusters C_1, C_2 contained in some $N^{2d}(U_i)$, then add every edge between C_1 and C_2 .*

Proof of safeness of Rule 4. It follows directly from the fact that adding edges between vertices of a monochromatic set preserves the existence of a d -cut. ◀

The next lemma follows from the pigeonhole principle and exhaustive application of Rule 4.

► **Lemma 12.** *If G has been reduced by Rules 1 through 4, then G has $\mathcal{O}(|U|)$ fixed clusters.*

► **Reduction Rule 5.** *If there is some cluster C with at least $2d + 2$ vertices such that there is some $v \in C$ with no neighbors in U , remove v from G .*

Proof of safeness of Rule 5. That G has a d -cut if and only if $G - v$ has a d -cut follows directly from the hypothesis that C is monochromatic in G and the fact that $|C \setminus \{v\}| \geq 2d + 1$ implies that $C \setminus \{v\}$ is monochromatic in $G - v$. ◀

By Rule 5, we now have the additional property that, if C has more than $2d + 1$ vertices, all of them have at least one neighbor in U . The next rule provides a uniform structure between a big cluster C and the sets U_i such that $C \subseteq N^{2d}(U_i)$.

► **Reduction Rule 6.** *If a cluster C has at least $2d + 1$ elements and there is some U_i such that $C \subseteq N^{2d}(U_i)$, remove all edges between C and U_i , choose $u \in U_i$, $\{v_1, \dots, v_{d+1}\} \subseteq C$ and add the edges $\{uv_i\}_{i \in [d+1]}$ to G .*

Proof of safeness of Rule 6. Let G' be the graph obtained after the operation is applied. If G has some d -cut (A, B) , since $U_i \cup N^{2d}(U_i)$ is monochromatic, no edge between U_i and C crosses the cut, so (A, B) is also a d -cut of G' . For the converse, take a d -cut (A', B') of G' . Since C has at least $2d + 1$ vertices and there is some $u \in U_i$ such that $|N(u) \cap C| = d + 1$, $C \in N^{2d}(U_i)$ in G' . Therefore, no edge between C and U_i crosses the cut and (A', B') is also a d -cut of G . ◀

We have now effectively bounded the number of vertices in big clusters by a polynomial in U , as shown below.

► **Lemma 13.** *If G has been reduced by Rules 1 through 6, then G has $\mathcal{O}(d|U|^2)$ ambiguous vertices and $\mathcal{O}(d|U|^2)$ big clusters, each with $\mathcal{O}(d|U|)$ vertices.*

Proof. To show the bound on the number of ambiguous vertices, take any two vertices $u \in U_i$, $u' \in U_j$. Since we have $\binom{|U|}{2}$ such pairs, if we had at least $(2d + 1)\binom{|U|}{2}$ ambiguous vertices, by the pigeonhole principle, there would certainly be $2d + 1$ vertices in $V \setminus U$ that are adjacent to one pair, say u and u' . This, however, contradicts the hypothesis that Rule 3 has been applied, and so we have $\mathcal{O}(d|U|^2)$ ambiguous vertices.

The above discussion, along with Lemma 12 and Observation 3, imply that the number of big clusters is $\mathcal{O}(d|U|^2)$. For the bound on their sizes, take some cluster C with at least $2d + 2$ vertices. Due to the application of Rule 5, every vertex of C has at least one neighbor in U . Moreover, there is at most one U_i such that $C \subseteq N^{2d}(U_i)$, otherwise we would be able to apply Rule 2.

Suppose first that there is such a set U_i . By Rule 6, there is only one $u \in U_i$ that has neighbors in C ; in particular, it has $d + 1$ neighbors. Now, every $v \in U_j$, for every $j \neq i$, has at

19:10 Finding Cuts of Bounded Degree

most d neighbors in C , otherwise $C \subseteq N^{2d}(U_j)$ and Rule 2 would have been applied. Therefore, we conclude that C has at most $(d+1) + \sum_{v \in U \setminus U_i} |N(v) \cap C| \leq (d+1) + d|U| \in \mathcal{O}(d|U|)$ vertices.

Finally, suppose that there is no U_i such that $C \subseteq N^{2d}(U_i)$. A similar analysis from the previous case can be performed: every $u \in U_i$ has at most d neighbors in C , otherwise $C \subseteq N^{2d}(U_i)$ and we conclude that C has at most $\sum_{v \in U} |N(v) \cap C| \leq d|U| \in \mathcal{O}(d|U|)$ vertices. ◀

We are now left only with an unbounded number of small clusters. A cluster C is *simple* if it is not ambiguous, that is, if for each $v \in C$, v has neighbors in a single U_i . Otherwise, C is ambiguous and, because of Lemma 13, there are at most $\mathcal{O}(d|U|^2)$ such clusters. For a simple cluster C and a vertex $v \in C$, we denote by $U(v)$ the monochromatic part of U to which v is adjacent.

► **Reduction Rule 7.** *If C is a simple cluster with at most $d+1$ vertices, remove C from G .*

Proof of safeness of Rule 7. Let $G' = G - C$. Suppose G has a d -cut (A, B) and note that $A \not\subseteq C$ and $B \not\subseteq C$ since Rule 1 does not apply. This implies that $(A \setminus C, B \setminus C)$ is a valid d -cut of G' . For the converse, take a d -cut (A', B') of G' , define $C_A = \{v \in C \mid U(v) \subseteq A\}$, and define C_B similarly; we claim that $(A' \cup C_A, B' \cup C_B)$ is a d -cut of G . To see that this is the case, note that each vertex of C_A (resp. C_B) has at most d edges to C_B (resp. C_A) and, since C is simple, C_A (resp. C_B) has no other edges to B' (resp. A'). ◀

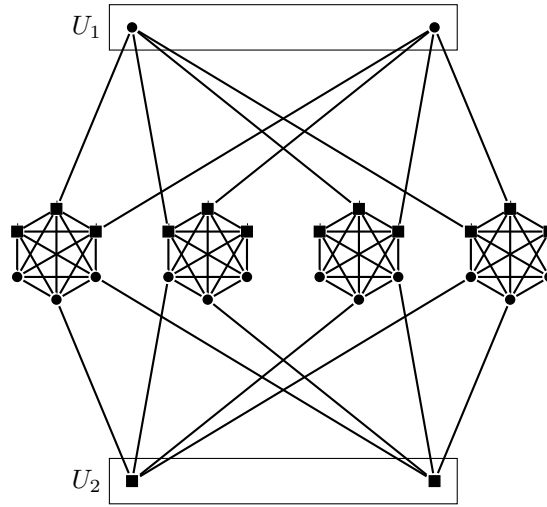
After applying the previous rule, every cluster C not yet analyzed has size $d+2 \leq |C| \leq 2d$ which, in the case of the MATCHING CUT problem, where $d = 1$, is empty. To deal with these clusters, given a d -cut (A, B) , we say that a vertex v is in its *natural assignment* if $v \cup U(v)$ is in the same side of the cut; otherwise the vertex is in its *unnatural assignment*. Similarly, a cluster is *unnaturally assigned* if it has an unnaturally assigned vertex, otherwise it is *naturally assigned*.

► **Observation 4.** *Let \mathcal{C} be the set of all simple clusters with at least $d+2$ and no more than $2d$ vertices, and (A, B) a partition of $V(G)$. If there are $d|U| + 1$ edges uv , $v \in C \in \mathcal{C}$ and $u \in U$, such that uv is crossing the partition, then (A, B) is not a d -cut.*

Proof. Since there are $d|U| + 1$ edges crossing the partition between \mathcal{C} and U , there must be at least one $u \in U$ with $d+1$ neighbors in the other set of the partition. ◀

► **Corollary 14.** *In any d -cut of G , there are at most $d|U|$ unnaturally assigned vertices.*

Our next lemma limits how many clusters in \mathcal{C} relate in a similar way to U ; we say that two simple clusters C_1, C_2 have the same *pattern* if they have the same size s and there is a total ordering of C_1 and another of C_2 such that, for every $i \in [s]$, $v_i^1 \in C_1$ and $v_i^2 \in C_2$ satisfy $U(v_i^1) = U(v_i^2)$. Essentially, clusters that have the same pattern have neighbors in exactly the same monochromatic sets of U and the same multiplicity in terms of how many of their vertices are adjacent to a same monochromatic set U_i . Note that the actual neighborhoods in the sets U_i 's do not matter in order for two clusters to have the same pattern. Figure 2 gives an example of a maximal set of unnaturally assigned clusters; that is, any other cluster with the same pattern as the one presented must be naturally assigned, otherwise some vertex of U will violate the d -cut property. As shown by the following Lemma, we may discard clusters that must be naturally assigned, as we can easily extend the kernel's d -cut, if it exists, to include them.



■ **Figure 2** Example for $d = 4$ of a maximal set of unnaturally assigned clusters. Squared (resp. circled) vertices would be assigned to A (resp. B).

► **Lemma 15.** *Let $\mathcal{C}^* \subseteq \mathcal{C}$ be a subfamily of simple clusters, all with the same pattern, with $|\mathcal{C}^*| > d|U| + 1$. Let C be some cluster of \mathcal{C}^* , and $G' = G - C$. Then G has a d -cut if and only if G' has a d -cut.*

Proof. Since by Rule 1 no subset of a small cluster is alone in a side of a partition and, consequently, U intersects both sides of the partition, if G has a d -cut, so does G' .

For the converse, let (A', B') be a d -cut of G' . First, by Corollary 14, we know that at least one of the clusters of $\mathcal{C}^* \setminus \{C\}$, say C_n , is naturally assigned. Since all the clusters in \mathcal{C}^* have the same pattern, this guarantees that *any* of the vertices of a naturally assigned cluster cannot have more than d neighbors in the other side of the partition.

Let (A, B) be the bipartition of $V(G)$ obtained from (A', B') such that $u \in C$ is in A (resp. B) if and only if $U(u) \subseteq A$ (resp. $U(u) \subseteq B$); that is, C is naturally assigned. Define $C_A = C \cap A$ and $C_B = C \cap B$. Because $|C| = |C_n|$ and both belong to \mathcal{C}^* , we know that for every $u \in C_A$, it holds that $|N(u) \cap C_B| \leq d$; moreover, note that $N(u) \cap (B \setminus C) = \emptyset$. A symmetric analysis applies to every $u \in C_B$. This implies that no vertex of C has additional neighbors in the other side of the partition outside of its own cluster and, therefore, (A, B) is a d -cut of G . ◀

The safeness of our last rule follows directly from Lemma 15.

► **Reduction Rule 8.** *If there is some pattern such that the number of simple clusters with that pattern is at least $d|U| + 2$, delete all but $d|U| + 1$ of them.*

► **Lemma 16.** *After exhaustive application of Rules 1 through 8, G has $\mathcal{O}(d|U|^{2d})$ small clusters and $\mathcal{O}(d^2|U|^{2d+1})$ vertices in these clusters.*

Proof. By Rule 7, no small cluster with less than $d + 2$ vertices remains in G . Now, for the remaining sizes, for each $d + 2 \leq s \leq 2d$, and each pattern of size s , by Rule 8 we know that the number of clusters with s vertices that have the same pattern is at most $d|U| + 1$. Since we have at most $|U|$ possibilities for each of the s vertices of a cluster, we end up with $\mathcal{O}(|U|^s)$ possible patterns for clusters of size s . Summing all of them up, we get that we have $\mathcal{O}(|U|^{2d})$ patterns in total, and since each one has at most $d|U| + 1$ clusters of size at most $2d$, we get that we have at most $\mathcal{O}(d^2|U|^{2d+1})$ vertices in those clusters. ◀

19:12 Finding Cuts of Bounded Degree

The exhaustive application of all the above rules and their accompanying lemmas are enough to show that indeed, there is a polynomial kernel for d -CUT when parameterized by distance to cluster.

► **Theorem 17.** *When parameterized by distance to cluster $\text{dc}(G)$, d -CUT admits a polynomial kernel with $\mathcal{O}(d^2 \cdot \text{dc}(G)^{2d+1})$ vertices that can be computed in $\mathcal{O}(d^4 \cdot \text{dc}(G)^{2d+1}(n+m))$ time.*

Proof. The algorithm begins by finding a set U such that $G - U$ is a cluster graph. Note that $|U| \leq 3\text{dc}(G)$ since a graph is a cluster graph if and only if it has no induced path on three vertices: while there is some P_3 in G , we know that at least one its vertices must be removed, but since we don't know which one, we remove all three; thus, U can be found in $\mathcal{O}(\text{dc}(G)(n+m))$ time. After the exhaustive application of Rules 1 through 8, by Lemma 13, $V(G) \setminus U$ has at most $\mathcal{O}(d^2 \cdot \text{dc}(G)^3)$ vertices in clusters of size at least $2d+1$. By Rule 7, G has no simple cluster of size at most $d+1$. Ambiguous clusters of size at most $2d$, again by Lemma 13, also comprise only $\mathcal{O}(d^2 \cdot \text{dc}(G)^2)$ vertices of G . Finally, for simple clusters of size between $d+2$ and $2d$, Lemmas 15 and 16 guarantee that there are $\mathcal{O}(d^2 \cdot \text{dc}(G)^{2d+1})$ vertices in small clusters and, consequently, this many vertices in G .

As to the running time, first, computing and maintaining $N^{2d}(U_i)$ takes $\mathcal{O}(d \cdot \text{dc}(G)n)$ time. Rule 1 is applied only at the beginning of the kernelization, and runs in $\mathcal{O}(2^{2d}d(n+m))$ time. Rules 2 and 3 can both be verified in $\mathcal{O}(d \cdot \text{dc}(G)^2(n+m))$ time, since we are just updating $N^{2d}(U_i)$ and performing merge operations. Both are performed only $\mathcal{O}(\text{dc}(G)^2)$ times, because we only have this many pairs of monochromatic parts. The straightforward application of Rule 4 would yield a running time of $\mathcal{O}(n^2)$. However, we can ignore edges that are interior to clusters and only maintain which vertices belong together; this effectively allows us to perform this rule in $\mathcal{O}(n)$ time, which, along with its $\mathcal{O}(n)$ possible applications, yields a total running time of $\mathcal{O}(n^2)$ for this rule. Note that, when outputting the instance itself, we must write the edges explicitly; this does not change the final complexity of the algorithm, as each of the $\mathcal{O}(\text{dc}(G)^{2d+1})$ clusters has $\mathcal{O}(d \cdot \text{dc}(G))$ vertices. Rule 5 is directly applied in $\mathcal{O}(n)$ time; indeed, all of its applications can be performed in a single pass. Rule 6 is also easily applied in $\mathcal{O}(n+m)$ time. Moreover, it is only applied $\mathcal{O}(\text{dc}(G))$ times, since, by Lemma 13, the number of fixed clusters is linear in $\text{dc}(G)$; furthermore, we may be able to reapply Rule 6 directly to the resulting cluster, at no additional complexity cost. The analysis for Rule 7 follows the same argument as for Rule 5. Finally, Rule 8 is the bottleneck of our kernel, since it must check each of the possible $\mathcal{O}(\text{dc}(G)^{2d})$ patterns, spending $\mathcal{O}(n)$ time for each of them. Each pattern is only inspected once because the number of clusters in a pattern can no longer achieve the necessary bound for the rule to be applied once the excessive clusters are removed. ◀

In the next theorems, we provide FPT algorithms for d -CUT parameterized by distance to cluster and distance to co-cluster, respectively. Both are based on dynamic programming, with the first being considerably simpler than the one given by Komusiewicz et al. [18] for $d=1$, which applies four reduction rules and encodes the problem in a 2-SAT formula. However, for $d=1$ our algorithm is slower, namely $\mathcal{O}^*(4^{\text{dc}(G)})$ compared to $\mathcal{O}^*(2^{\text{dc}(G)})$. Observe that the minimum distance to cluster and co-cluster sets can be computed in time $1.92^{\text{dc}(G)} \cdot \mathcal{O}(n^2)$ and $1.92^{\text{dc}(G)} \cdot \mathcal{O}(n^2)$, respectively [6]. Thus, in the proofs of Theorems 18 and 19, we can safely assume that we have these sets at hand.

► **Theorem 18** (\star). *For every integer $d \geq 1$, there is an algorithm that solves d -CUT in time $\mathcal{O}(4^d(d+1)^{\text{dc}(G)} 2^{\text{dc}(G)} \text{dc}(G)n^2)$.*

► **Theorem 19** (\star). *For every integer $d \geq 1$, there is an algorithm solving d -CUT in time $\mathcal{O}(32^d 2^{d\bar{d}(G)} (d+1)^{d\bar{d}(G)+d} (d\bar{d}(G)+d)n^2)$.*

Using Theorems 18 and 19, and the relation $\tau(G) \geq \max\{dc(G), d\bar{d}(G)\}$ [18], we obtain the following corollary.

► **Corollary 20.** *For every $d \geq 1$, d -CUT parameterized by vertex cover is in FPT.*

4 Concluding remarks

We presented a series of algorithms and complexity results; many questions, however, remain open. For instance, all of our algorithms have an exponential dependency on d on their running times. While we believe that such a dependency is an intrinsic property of d -CUT, we have no proof for this claim. Similarly, the existence of a *uniform* polynomial kernel parameterized by the distance to cluster, i.e., a kernel whose degree does not depend on d , remains an interesting open question.

Also in terms of running time, we expect the constants in the base of the exact exponential algorithm to be improvable. However, exploring small structures that yield non-marginal gains as branching rules, as done by Komusiewicz et al. [18] for $d = 1$ does not seem a viable approach, as the number of such structures appears to rapidly grow along with d .

The distance to cluster kernel is hindered by the existence of clusters of size between $d+2$ and $2d$, an obstacle that is not present in the MATCHING CUT problem. Aside from the extremal argument presented, we know of no way of dealing with them. We conjecture that it should be possible to reduce the total kernel size from $\mathcal{O}(d^2 dc(G)^{2d+1})$ to $\mathcal{O}(d^2 dc(G)^{2d})$, matching the size of the smallest known kernel for MATCHING CUT [18].

We also leave open to close the gap between the polynomial and NP-hard cases in terms of maximum degree. We showed that, if $\Delta(G) \leq d+2$ the problem is easily solvable in polynomial time, while for graphs with $\Delta(G) \geq 2d+2$, it is NP-hard. But what about the gap $d+3 \leq \Delta(G) \leq 2d+1$? After some effort, we were unable to settle any of these cases. In particular, we are interested in 2-CUT, which has a single open case: $\Delta(G) = 5$. After some weeks of computation, we found no graph with more than 18 vertices and maximum degree five that had no 2-cut, in agreement with the computational findings of Ban and Linial [2]. Interestingly, all graphs on 18 vertices without a 2-cut are either 5-regular or have a single pair of vertices of degree 4, which are actually adjacent. In both cases, the graph is maximal in the sense that we cannot add edges to it while maintaining the degree constraints. We recall the initial discussion about INTERNAL PARTITION; closing the gap between the known cases for d -CUT would yield significant advancements on the former problem.

Finally, the smallest d for which G admits a d -cut may be an interesting additional parameter to be considered when more traditional parameters, such as treewidth, fail to provide FPT algorithms by themselves. Unfortunately, by Theorem 1, computing this parameter is not even in XP, but, as we have shown, it can be computed in FPT time under many different parameterizations.

References

- 1 N. R. Aravind, Subrahmanyam Kalyanasundaram, and Anjeneya Swami Kare. On Structural Parameterizations of the Matching Cut Problem. In *Proc. of the 11th International Conference on Combinatorial Optimization and Applications (COCO)*, volume 10628 of *LNCS*, pages 475–482, 2017.

- 2 Amir Ban and Nati Linial. Internal partitions of regular graphs. *Journal of Graph Theory*, 83(1):5–18, 2016.
- 3 Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *Journal of Computer and System Sciences*, 75(8):423–434, 2009.
- 4 Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Cross-Composition: A New Technique for Kernelization Lower Bounds. In *Proc. of the 28th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 9 of *LIPICs*, pages 165–176, 2011.
- 5 Paul S. Bonsma. The complexity of the matching-cut problem for planar graphs and other graph classes. *Journal of Graph Theory*, 62(2):109–126, 2009.
- 6 Anudhyan Boral, Marek Cygan, Tomasz Kociumaka, and Marcin Pilipczuk. A fast branching algorithm for cluster vertex deletion. *Theory of Computing Systems*, 58(2):357–376, 2016.
- 7 Vasek Chvátal. Recognizing decomposable graphs. *Journal of Graph Theory*, 8(1):51–53, 1984.
- 8 Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and computation*, 85(1):12–75, 1990.
- 9 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.
- 10 Matt DeVos. http://www.openproblemgarden.org/op/friendly_partitions, 2009.
- 11 Reinhard Diestel. *Graph Theory*, volume 173. Springer-Verlag, 4th edition, 2010.
- 12 R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.
- 13 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2019.
- 14 Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct PCPs for NP. *Journal of Computer and System Sciences*, 77(1):91–106, 2011.
- 15 Ron L Graham. On primitive graphs and optimal vertex assignments. *Annals of the New York academy of sciences*, 175(1):170–186, 1970.
- 16 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k -SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.
- 17 Atsushi Kaneko. On decomposition of triangle-free graphs under degree constraints. *Journal of Graph Theory*, 27(1):7–9, 1998.
- 18 Christian Komusiewicz, Dieter Kratsch, and Van Bang Le. Matching Cut: Kernelization, Single-Exponential Time FPT, and Exact Exponential Algorithms. In *Proc. of the 13th International Symposium on Parameterized and Exact Computation (IPEC)*, volume 115 of *LIPICs*, pages 19:1–19:13, 2018.
- 19 Dieter Kratsch and Van Bang Le. Algorithms solving the matching cut problem. *Theoretical Computer Science*, 609:328–335, 2016.
- 20 Hoàng-Oanh Le and Van Bang Le. On the Complexity of Matching Cut in Graphs of Fixed Diameter. In *Proc. of the 27th International Symposium on Algorithms and Computation (ISAAC)*, volume 64 of *LIPICs*, pages 50:1–50:12, 2016.
- 21 Van Bang Le and Bert Randerath. On stable cutsets in line graphs. *Theoretical Computer Science*, 301(1-3):463–475, 2003.
- 22 Jie Ma and Tianchi Yang. Decomposing C_4 -free graphs under degree constraints. *Journal of Graph Theory*, 90(1):13–23, 2019.
- 23 Dániel Marx, Barry O’Sullivan, and Igor Razgon. Treewidth Reduction for Constrained Separation and Bipartization Problems. In *Proc. of the 27th International Symposium on Theoretical Aspects of Computer Science, (STACS)*, volume 5 of *LIPICs*, pages 561–572, 2010.
- 24 Augustine M Moshi. Matching cutsets in graphs. *Journal of Graph Theory*, 13(5):527–536, 1989.
- 25 Maurizio Patrignani and Maurizio Pizzonia. The complexity of the matching-cut problem. In *Proc. of the 27th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 2204 of *LNCS*, pages 284–295, 2001.

- 26 Khurram H. Shafique and Ronald D. Dutton. On satisfactory partitioning of graphs. *Congressus Numerantium*, pages 183–194, 2002.
- 27 Michael Stiebitz. Decomposing graphs under degree constraints. *Journal of Graph Theory*, 23(3):321–324, 1996.
- 28 Carsten Thomassen. Graph decomposition with constraints on the connectivity and minimum degree. *Journal of Graph Theory*, 7(2):165–167, 1983.
- 29 Chee-Keng Yap. Some Consequences of Non-Uniform Conditions on Uniform Classes. *Theoretical Computer Science*, 26:287–300, 1983.