Matching Cut: Kernelization, Single-Exponential Time FPT, and Exact Exponential Algorithms

Article *in* Discrete Applied Mathematics · December 2019

DOI: 10.1016/j.dam.2019.12.010

Christian Komusiewicz
Philipps University of Marburg

Van Bang Le
University of Rostock

Matching Cut: Kernelization, Single-Exponential Time FPT, and Exact Exponential Algorithms[☆]

Christian Komusiewicz

Fachbereich Mathematik und Informatik, Philipps-Universität Marburg, Marburg, Germany

Dieter Kratsch

Laboratoire de Génie Informatique, de Production et de Maintenance, Université de Lorraine, Metz, France

Van Bang Le

Universität Rostock, Institut für Informatik, Rostock, Germany

---

Abstract

In a graph, a matching cut is an edge cut that is a matching. Matching Cut, which is known to be NP-complete, is the problem of deciding whether or not a given graph G has a matching cut. In this paper we show that Matching Cut admits a quadratic-vertex kernel for the parameter distance to cluster and a linear-vertex kernel for the parameter distance to clique. We further provide an $O^*(2^{\mathrm{dc}(G)})$-time and an $O^*(2^{\mathrm{d\overline{c}}(G)})$-time FPT algorithm for Matching Cut, where $\mathrm{dc}(G)$ and $\mathrm{d\overline{c}}(G)$ are the distance to cluster and distance to co-cluster, respectively. We also improve the running time of the best known branching algorithm to solve Matching Cut from $O^*(1.4143^n)$ to $O^*(1.3071^n)$ where n is the number of vertices in G. Moreover, we point out that, unless NP $\subseteq$ coNP/poly, Matching Cut does not admit a polynomial kernel when parameterized simultaneously by treewidth, the number of edges crossing the cut, and the maximum degree of G.

Keywords: Graph problem, NP-hard problem, kernelization, branching algorithm

---

[☆]A preliminary version of this article appeared in *Proceedings of the 13th International Symposium on Parameterized and Exact Computation (IPEC '18)*, Helsinki, Finland, August 2018.

Email addresses: `komusiewicz@informatik.uni-marburg.de` (Christian Komusiewicz), `dieter.kratsch@univ-lorraine.fr` (Dieter Kratsch), `van-bang.le@uni-rostock.de` (Van Bang Le)

1. Introduction

In a graph $G = (V, E)$, a *cut* is a partition $V = A \dot\cup B$ of the vertex set into disjoint, nonempty sets A and B, written (A, B). The set of all edges in G having an endvertex in A and the other endvertex in B, also written (A, B), is called the *edge cut* of the cut (A, B). A *matching cut* is a possibly empty edge cut that is a matching. Note that, by our definition, a matching whose removal disconnects the graph need not be a matching cut.

Another way to define matching cuts is as follows ([16, 8]). A partition $V = A \dot\cup B$ of the vertex set of the graph $G = (V, E)$ into disjoint, nonempty sets A and B, is a matching cut if and only if each vertex in A has at most one neighbor in B and each vertex in B has at most one neighbor in A. Not every graph has a matching cut; the MATCHING CUT problem is the problem of deciding whether or not a given graph has a matching cut:

MATCHING CUT	
Instance:	A graph $G = (V, E)$.
Question:	Does G have a matching cut?

Farley and Proskurowski [13] studied matching cuts in graphs in the context of network applications. Patrignani and Pizzonia [25] pointed out an application of matching cuts in graph drawing. Graphs having no matching cut were first discussed by Graham in [16] under the name *indecomposable graphs*, and have been used by Araújo et al. [1] in the context of WDM (Wavelength Division Multiplexing) networks.

Previous Results and Related Work. Chvátal [8] showed that MATCHING CUT is NP-complete, even when restricted to graphs of maximum degree four, and polynomially solvable for graphs of maximum degree three. These results triggered a lot of research on the computational complexity of MATCHING CUT in graphs with additional structural assumptions [5, 7, 19, 21, 20, 24, 25]. In particular, the NP-hardness of MATCHING CUT has been further strengthened to planar graphs of maximum degree four [5] and bipartite graphs of maximum degree four [21]. As noted previously [19], the NP-hardness reduction by Chvátal [8] also shows that MATCHING CUT cannot be solved in $2^{o(n)}$ time, where n is the number of vertices of the input graph, if the Exponential Time Hypothesis (ETH) is true.

Exact exponential algorithms for MATCHING CUT on graphs without any restriction have been recently considered by Kratsch and Le [19] who provided the first exact branching algorithm for MATCHING CUT running in time $O^*(1.4143^n)$[1], and a single-exponential algorithm of running time $2^{\tau(G)}O(n^2)$, where $\tau(G)$ is the vertex cover number. We note that MATCHING CUT can be expressed in MSOL, see for example [5]; hence MATCHING CUT is fixed-parameter tractable when parameterized by $\mathrm{tw}(G)$, the treewidth of G. Aravind

[1]Throughout the paper we use the O^* notation which suppresses polynomial factors.

et al. [2] presented a tree-decomposition-based dynamic programming algorithm that solves Matching Cut in $O^*(12^{\mathrm{tw}(G)})$ time and fixed-parameter algorithms for further parameters describing the structure of the input graph. For example, Matching Cut can be solved in $O^*(2^{\mathrm{tc}(G)})$ time where $\mathrm{tc}(G) \le \tau(G)$ is the size of a smallest twin cover of G [2]. Recently, the dynamic programming algorithm for Matching Cut parameterized by treewidth was improved to a running time of $O^*(8^{\mathrm{tw}(G)})$ [15]. Finally, as noted by Gomes and Sau [15], the treewidth reduction technique [23] can be used to obtain a fixed-parameter algorithm for Matching Cut parameterized by the number of edges crossing the cut.

Our Contributions. We give the first nontrivial polynomial kernels for Matching Cut by showing that Matching Cut admits a quadratic-vertex kernel for the parameter distance to cluster and a linear-vertex kernel for the parameter distance to clique. Second, we show that Matching Cut can be solved by single-exponential algorithms running in time $2^{\mathrm{dc}(G)}O(n^2)$ and $2^{\mathrm{d\overline{c}}(G)}O(nm)$, respectively, where $\mathrm{dc}(G)$ is the distance to cluster, $\mathrm{d\overline{c}}(G)$ is the distance to co-cluster, and m is the number of edges of G. This improves upon the FPT algorithms for Matching Cut with running time $2^{\tau(G)}O(n^2)$ [19], where $\tau(G) \ge \max\{\mathrm{dc}(G), \mathrm{d\overline{c}}(G)\}$ is the vertex cover number of G which can be much larger than $\mathrm{dc}(G)$ and $\mathrm{d\overline{c}}(G)$. Similarly, this improves upon the FPT algorithm with running time $O^*(2^{\mathrm{tc}(G)})$ [2] since $\mathrm{tc}(G) \ge \mathrm{dc}(G)$. Third, we provide a Sat-based $O^*(1.3071^n)$-time randomized algorithm and an exact branching algorithm for Matching Cut that has time complexity $O^*(1.3803^n)$. Both improve upon the first exact branching algorithm for Matching Cut that has time complexity $O^*(1.4143^n)$ [19].

The polynomial kernel and the fixed-parameter algorithm for the parameter $\mathrm{dc}(G)$ have been subsequently generalized to the d-Cut problem where we ask for a cut (A, B) such that every $v \in A$ has at most d neighbors in B and vice versa [15]. For Matching Cut, these more general algorithms achieve a kernel with $O(\mathrm{dc}(G)^3)$ vertices and a running time of $4^{\mathrm{dc}(G)}O(\mathrm{dc}(G)^2 n^2)$, respectively. Hence, our algorithms still constitute the state-of-the-art for Matching Cut parameterized by $\mathrm{dc}(G)$.

Notation and Terminology. Let $G = (V, E)$ be a graph with vertex set $V(G) := V$ and edge set $E(G) := E$. We use $n := |V|$ and $m := |E|$ to denote the number of vertices and edges in the graph under consideration. A *stable set* (a *clique*) in G is a set of pairwise non-adjacent (adjacent) vertices. The neighborhood of a vertex v in G, denoted by $N_G(v)$, is the set of all vertices in G adjacent to v; if the context is clear, we simply write $N(v)$. Set $\deg(v) := |N(v)|$, the degree of the vertex v. For a subset $W \subseteq V$, $G[W]$ is the subgraph of G induced by W, and $G - W$ stands for $G[V \setminus W]$. We write $N_W(v)$ for $N(v) \cap W$ and call the vertices in $N(v) \cap W$ the *W-neighbors* of v. A graph is a *cluster graph* if it is a vertex disjoint union of cliques. The maximal cliques of a cluster graph are called *clusters*. A graph is a *co-cluster graph* if it is a complete multipartite graph or, equivalently, the complement graph of a cluster graph. Observe that a clique is a cluster graph and a co-cluster graph.

A *vertex cover* of G is a subset $C \subseteq V$ such that every edge of G has at least one endvertex in C, that is, $V \setminus C$ is a stable set in G. The vertex cover number of G, denoted by $\tau(G)$, is the smallest size of a vertex cover of G. More generally, given a graph property $\mathcal{P}$, a *distance to $\mathcal{P}$ set* of a graph G is a subset $U \subseteq V$ such that $G - U$ has the property $\mathcal{P}$. The *distance to $\mathcal{P}$* is the smallest size of a distance to $\mathcal{P}$ set. This number is called *distance to cluster*, denoted by $\mathrm{dc}(G)$, in case $\mathcal{P}$ is the set of cluster graphs, it is called *distance to co-cluster*, denoted by $\mathrm{d\overline{c}}(G)$, in case $\mathcal{P}$ is the set of co-cluster graphs, and it is called *distance to clique*, denoted by $\mathrm{dq}(G)$, in case $\mathcal{P}$ is the set of cliques. By the definition of cluster graphs and co-cluster graphs, we have $\tau(G) \geq \max\{\mathrm{dc}(G), \mathrm{d\overline{c}}(G)\}$ and $\mathrm{dq}(G) \geq \max\{\mathrm{dc}(G), \mathrm{d\overline{c}}(G)\}$ for any graph G.

Throughout the paper we use the concept of monochromatic vertex subsets and induced subgraphs. Let $G = (V, E)$ be a graph and $U \subseteq V$. We call U *monochromatic* in G if for every matching cut (A, B) of G, either $U \subseteq A$ or $U \subseteq B$; slightly abusing notation we shall sometimes also call $G[U]$ monochromatic in G. Note that a complete subgraph K_n is monochromatic if $n = 1$ or $n \geq 3$, and that a complete bipartite subgraph $K_{n,m}$ is monochromatic if $n \geq 3$ and $m \geq 2$ or vice versa. Moreover, disconnected graphs and graphs having a vertex of degree at most one admit a matching cut. Hence, we may assume that all graphs considered are connected and have minimum degree at least two.

Parameterized complexity deals with NP-hard problems whose instances come equipped with an additional integer parameter k. Its main objective is to design algorithms whose running time is $f(k)\cdot\mathrm{poly}(n)$ for some computable function f. Problems admitting such algorithms are called *fixed-parameter tractable*. Another major concept in parameterized complexity are polynomial (size) kernels. A *kernelization* algorithm, or simply a *kernel*, for a parameterized problem Q is an algorithm A that, given an instance (I, k) of Q, works in polynomial time and returns an equivalent instance (I', k') of Q such that $|I'| + k' \leq g(k)$ for some computable function g depending only on k; we say that the kernel has *size* $g(k)$. If g is a polynomial, then we say that the kernel is polynomial. For more information on parameterized algorithms we refer to the standard monographs [9, 11].

When an algorithm branches on the current instance of size n into r subproblems of sizes at most $n-t_1, n-t_2, \ldots, n-t_r$, then $(t_1, t_2, \ldots, t_r)$ is called the *branching vector* of this branching, and the unique positive root of $x^n - x^{n-t_1} - x^{n-t_2} - \cdots - x^{n-t_r} = 0$, denoted by $\tau(t_1, t_2, \ldots, t_r)$, is called its *branching number*. The running time of a branching algorithm is $O^*(\alpha^n)$, where $\alpha = \max_i \alpha_i$ and α_i is the branching number of branching rule i, and the maximum is taken over all branching rules. We refer to [14] for more details on exact branching algorithms.

2. A Polynomial Kernel for the Distance to Cluster

In this section we present a polynomial kernel for the parameter $\mathrm{dc}(G)$, the distance of G to a cluster graph. One reason for considering this parameter is that $\mathrm{dc}(G)$ is never larger than the vertex cover number $\tau(G)$ while also being arbitrarily smaller in some instances. Hence, achieving a polynomial kernel for $\mathrm{dc}(G)$ strengthens the previous fixed-parameter tractability result for $\tau(G)$ [19]. In addition, we motivate the study of kernelization for the parameter $\mathrm{dc}(G)$ by the following negative result.

Recall that there is an FPT algorithm for Matching Cut when parameterized by treewidth [2, 5]. Hence, a natural question is whether Matching Cut admits a polynomial kernel for this parameter. A further candidate parameter for a polynomial kernel is the minimum number k of edges crossing any matching cut; this could be considered the standard solution size parameter for Matching Cut. Finally, a common parameter in kernelizations is the maximum degree of the input graph. We rule out polynomial kernels for all three parameters.

Proposition 1. Matching Cut *does not admit polynomial kernel with respect to the sum of the treewidth of G, the minimum number of edges crossing any matching cut of G, and the maximum degree in G unless* $\mathrm{NP} \subseteq \mathrm{coNP/poly}$.

Proof. We show that Matching Cut cross-composes into itself (for the cross-composition framework, see [4]) via the following reduction.

Given $G_1, \ldots, G_t$ with the same number n of vertices, choose an arbitrary vertex $v_i \in V(G_i)$ for each $1 \leq i \leq t$. Let G be obtained from the disjoint union of $G_i, 1 \leq i \leq t$, by adding for each $1 \leq i < t$ two new vertices u_i and w_i and five edges u_iw_i, u_iv_i, w_iv_i, u_iv_{i+1}, and w_iv_{i+1}. Then G has a matching cut if and only if some G_i has a matching cut: First, if, for some i, (A_i, B_i) is a matching cut of G_i with $v_i \in A_i$, say, then $(V(G) \setminus B_i, B_i)$ clearly is a matching cut of G. Second, if (A, B) is a matching cut of G, then $\{v_i, v_{i+1}\}$, $1 \leq i < t$, is monochromatic because both vertices form triangles with u_i and w_i. Thus, $\{u_1, \ldots, u_{t-1}, w_1, \ldots, w_{t-1}, v_1, \ldots, v_t\}$ belongs to A or else to B. Assume $\{u_1, \ldots, u_{t-1}, w_1, \ldots, w_{t-1}, v_1, \ldots, v_t\} \subseteq A$, hence $A \cap V(G_i) \neq \emptyset$ for all i. Then, as $A \neq V(G)$, there exists some i such that $B \cap V(G_i) \neq \emptyset$, and $(A \cap V(G_i), B \cap V(G_i))$ clearly is a matching cut of G_i.

The treewidth, the number k of edges crossing the cut, and the maximum degree of G are clearly bounded by a polynomial of $\max_{1 \leq i \leq t} |V(G_i)| = n$. $\square$

Observe that this result also holds for the restriction of Matching Cut to planar graphs: in this case, we may assume that all instances G_i are planar and by fixing a planar embedding for each and choosing v_i to be a vertex on the outer face we obtain that the output instance is also planar.

Summarizing, we have excluded many natural candidate parameters for kernelization which motivates our study of kernelization for $\mathrm{dc}(G)$, the distance of G to a cluster graph. Observe that $\mathrm{dc}(G)$ is unrelated to the treewidth of G in the sense that there are graphs for which $\mathrm{dc}(G)$ is much smaller than $\mathrm{tw}(G)$

and vice versa. Let $U = \{u_1, \dots, u_{|U|}\}$ denote a distance to cluster set, that is, $G - U$ is a cluster graph. We may assume that $|U| \leq 3\text{dc}(G)$ since a 3-approximate distance to cluster set can be computed in time $O(\text{dc}(G)(n+m))$ based on the observation that a graph is a cluster graph if and only if it does not contain an induced path on three vertices. During the kernelization, we maintain a partition of U into $U_1, \dots, U_\ell$ such that each U_i is monochromatic. The initial partition contains one set for each vertex of U, that is, $U_i := \{u_i\}$, $1 \leq i \leq |U|$. We call the sets of the partition the *monochromatic parts* of U.

During the kernelization, we may *merge* two sets U_i and U_j, $i \neq j$, which is to remove U_i and U_j from the partition and to add $U_i \cup U_j$. We say that merging U_i and U_j is *safe* if $U_i \cup U_j$ is monochromatic in G.

The main idea of the kernelization is to find opportunities to merge monochromatic parts of U or to transform distinct clusters into a single cluster because we infer that they form a larger monochromatic set. Intuitively, an opportunity for merging arises when there are many vertices in $V \setminus U$ with at least two neighbors in U. The first step handles some clusters that have no such vertex.

Reduction Rule 1. *If $V \setminus U$ contains a degree-one vertex or a cluster C such that $(V \setminus C, C)$ is a matching cut, then STOP: "G has a matching cut".*

The rule is trivially safe. After its application, every cluster C contains either a vertex v with two neighbors in U or there is a vertex in $u \in U$ with two neighbors in C.

To identify clusters that form monochromatic sets together with some monochromatic parts of U, we introduce the following notation, see Figure 1 for an illustration. For each monochromatic part U_i of U, we let $N^2(U_i)$ denote the set of vertices $v \in V \setminus U$ such that at least one of the following holds:

- v has two neighbors in U_i,
- v is in a cluster of size at least three in $G - U$ that contains a vertex that has two neighbors in U_i, or
- v is in a cluster C in $G - U$ and some vertex in U_i has two neighbors in C.

Proposition 2. *$U_i \cup N^2(U_i)$ is monochromatic.*

Proof. In the first case, v has two neighbors in the monochromatic set U_i and thus $U_i \cup \{v\}$ is monochromatic. In the second case, the cluster C containing v is monochromatic and contains a vertex w such that $U_i \cup \{w\}$ is monochromatic. Hence, $U_i \cup C$ is monochromatic. In the third case, the cluster C contains two vertices x and y that form a triangle with some vertex from U_i and thus $U_i \cup \{x, y\}$ is monochromatic. If $|C| = 2$, then $x = v$ or $y = v$; if $|C| > 3$, then C is monochromatic. In both cases $U_i \cup C$ is monochromatic. $\square$

The next two rules identify monochromatic parts that can be merged because they belong to overlapping monochromatic sets.

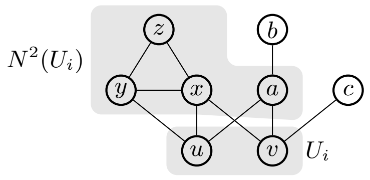

Figure 1: An example for the definition of $N^2(U_i)$ with $U_i = \{u, v\}$. We have $\{x, a\} \subseteq N^2(U_i)$ since x and a have two neighbors in U_i. Moreover, $\{y, z\} \subseteq N^2(U_i)$ since y and z are in a cluster of size at least three with x. Alternatively, $\{x, y, z\} \subseteq N^2(U_i)$ since $u \in U_i$ is adjacent to x and y. Finally, $b \notin N^2(U_i)$ even though $a \in N^2(U_i)$, since the cluster $\{a, b\}$ has only size two.

Reduction Rule 2. *If there is a vertex v that is contained in $N^2(U_i)$ and $N^2(U_j)$ for $i \neq j$, then merge U_i and U_j.*

Proof (of safeness). By Proposition 2, $\{v\} \cup U_i$ and $\{v\} \cup U_j$ are monochromatic in G. Thus, $U_i \cup U_j \cup \{v\}$ is monochromatic. $\square$

Reduction Rule 3. *If there are three vertices v_1, v_2, and v_3 in V that have two common neighbors $u \in U_i$ and $u' \in U_j$, $i \neq j$, then merge U_i and U_j.*

Proof (of safeness). The safeness follows from the fact that a $K_{n,m}$ is monochromatic for $n \geq 3$ and $m \geq 2$: Assume that u and u' are not in the same part of some matching cut (A, B). Then, at most one vertex of $\{v_1, v_2, v_3\}$ is in A and at most one is in B. This is absurd and, thus, $\{u, u'\}$ is monochromatic which makes $U_i \cup U_j$ monochromatic. $\square$

In the following, a cluster consisting of two vertices is an *edge cluster*, all other clusters are *nonedge clusters*. As we will show, after application of the above rules, we have essentially reached a situation in which there is a bounded number of nonedge clusters that are not contained in some $N^2(U_i)$, we call these clusters *ambiguous*. More precisely, we say that a vertex in $V \setminus U$ is *ambiguous* if it has neighbors in U_i and U_j where $i \neq j$. A cluster is *ambiguous* if it contains at least one ambiguous vertex. In contrast, we call a cluster *fixed* if it is contained in $N^2(U_i)$ for some U_i.

Proposition 3. *If G is reduced with respect to Rule 1, then every nonedge cluster in G is ambiguous or fixed.*

Proof. Since G is reduced with respect to Rule 1, every cluster C contains at least one vertex v that has two neighbors in U or there is a vertex u from some monochromatic part U_i that has two neighbors in C. In the latter case, C is contained in $N^2(U_i)$ and thus fixed. In the first case, if v has two neighbors in the same part U_i, then $C \subseteq N^2(U_i)$ and C is fixed. Otherwise, v is ambiguous which means that C is ambiguous. $\square$

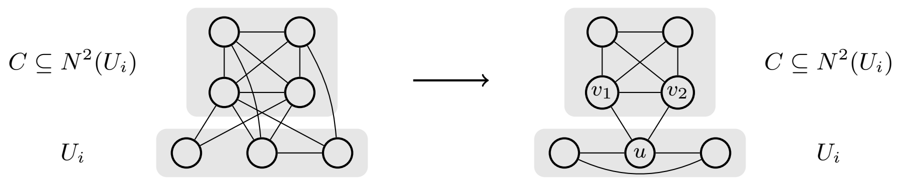

Figure 2: An example of the application of Rule 6.

Observe that according to this definition, a nonedge cluster may be ambiguous and fixed at the same time. We will decrease the number of fixed clusters with the following rule.

Reduction Rule 4. *If there are two clusters C_1 and C_2 that are contained in $N^2(U_i)$, then add all edges between C_1 and C_2.*

Proof (of safeness). Let G denote the graph to which the rule is applied and let G' denote the resulting graph. If G' has a matching cut, then so does G, because G is a subgraph of G' on the same vertex set.

For the converse, consider the following: $C_1 \cup C_2$ is monochromatic since C_1 and C_2 are contained in $N^2(U_i)$. Thus, if G has a matching cut, then so does G' because adding edges between vertices of a monochromatic set does not destroy the matching cut property. □

The next lemma follows from the pigeonhole principle and the fact that the number of monochromatic parts is at most $|U|$.

Lemma 1. *Let G be an instance of* Matching Cut *with cluster vertex deletion set U that is reduced with respect to Rule 4. Then G has $O(|U|)$ fixed clusters.*

The next rules will help in bounding the number of vertices in the clusters.

Reduction Rule 5. *If there is a cluster C with more than three vertices that contains a vertex v with no neighbors in U, then remove v.*

Proof (of safeness). Let G denote the original graph and let G' be the graph obtained from the application of the reduction rule. Since $|C| \geq 4$, C is monochromatic in G and $C \setminus \{v\}$ is monochromatic in G'. This and the fact that v has only neighbors in C immediately implies that G has a matching cut if and only if G' has a matching cut. □

After application of Rule 5, every vertex in a cluster of size at least four has a neighbor in U. The next rule removes unnecessary edges between monochromatic parts and clusters; an example application of the rule is shown in Figure 2.

Reduction Rule 6. *If there is a cluster C with at least three vertices and a monochromatic set U_i such that $C \subseteq N^2(U_i)$, then remove all edges between C and U_i from G, choose an arbitrary vertex $u \in U_i$ and two vertices $v_1, v_2 \in C$, and add two edges $\{u, v_1\}$ and $\{u, v_2\}$. If $|U_i| = 2$, then add an edge between $u' \in U_i \setminus \{u\}$ and $v_3 \in C \setminus \{v_1, v_2\}$. Finally, make U_i a clique.*

Proof (of safeness). Let G denote the original graph and let G' be the graph obtained from the application of the reduction rule. Since $C \subseteq N^2(U_i)$, we have, by Proposition 2, that $U_i \cup C$ is monochromatic. Thus, if G has a matching cut (A, B), then without loss of generality we have $U_i \cup C \subseteq A$. This implies that (A, B) is a matching cut of G' since G' is obtained from G by removing and adding certain edges between vertices in $U_i \cup C \subseteq A$.

Conversely, assume that G' has a matching cut. We show that $U_i \cup C$ is monochromatic in G'. First observe that C is monochromatic. If $|U_i| \neq 2$, then U_i is also monochromatic: if $|U_i| = 1$, then U_i is trivially monochromatic, otherwise U_i is a clique of size at least three and thus monochromatic. Consequently, $C \cup U_i$ is monochromatic as there is one vertex in U_i that forms a K_3 with two vertices of C. If $|U_i| = 2$, then one vertex u of U_i forms a K_3 with two vertices of C and thus $\{u\} \cup C$ is monochromatic. The other vertex u' of U_i has two neighbors in $\{u\} \cup C$ and thus $\{u, u'\} \cup C = U_i \cup C$ is monochromatic. Consequently, G can be obtained from G' by removing and adding certain edges between vertices of the monochromatic set $U_i \cup C$. As above, this implies that G has a matching cut. □

After application of these rules, the size of the instance is, with exception of the edge clusters, already bounded by a polynomial function of $|U|$ as we show in the following.

Lemma 2. *Let G be an instance of* Matching Cut *with cluster vertex deletion set U that is reduced with respect to Rules 1–6. Then G has*

- *$O(|U|^2)$ ambiguous vertices, and*
- *$O(|U|^2)$ nonedge clusters, each containing $O(|U|)$ vertices.*

Proof. First, we show the bound on the number of ambiguous vertices. Choose for each such vertex v two arbitrary neighbors $u \in U_i$ and $u' \in U_j$, $i \neq j$, and call u and u' the *representative neighbors* of v. Grouping these vertices according to their representative neighbors results in at most $\binom{|U|}{2}$ groups. Thus, if there are at least $3 \cdot \binom{|U|}{2}$ such vertices, then one of these groups, without loss of generality the group for $u \in U_i$ and $u' \in U_j$, has size at least three by the pigeonhole principle. Hence, there are three vertices in $V \setminus U$ that have $u \in U_i$ and $u' \in U_j$, $i \neq j$, as common neighbors. This contradicts the fact that Rule 3 is applied exhaustively. Hence, the number of these vertices is less than $3 \cdot \binom{|U|}{2} = O(|U|^2)$ as claimed.

Next, we bound the number of nonedge clusters. Every nonedge cluster is fixed or ambiguous by Proposition 3. The number of fixed nonedge clusters is $O(|U|)$ by Lemma 1. Moreover, the number of ambiguous clusters is bounded by the number of ambiguous vertices which is $O(|U|^2)$ by the first statement of the lemma.

It remains to show that each nonedge cluster C contains $O(|U|)$ vertices. Consider a cluster C of size at least four. Since G is reduced with respect to Rule 5, every vertex in C is the neighbor of some vertex in U. Moreover, if there are two vertices $u \in U_i$ and $u' \in U_j$ each with two neighbors in C, then either (i) $C \subseteq N^2(U_i)$ and $C \subseteq N^2(U_j)$ for some $i \neq j$ and Rule 2 would apply, or (ii) $U_i = U_j$ and Rule 6 would apply. Consequently, there is at most one vertex $u \in U$ that has at least two neighbors in C. Since G is reduced with respect to Rule 6, u has also at most two neighbors in C. All other vertices of U have at most one neighbor in C. Thus, the overall number of vertices in C is at most $2 + |\{u \in U : N(u) \cap C \neq \emptyset\}| = O(|U|)$. □

To obtain a first bound on the instance size, it remains to reduce the overall number of edge clusters. To this end, we consider for each edge cluster $\{u, v\}$ the neighborhoods of u and v in U. First, observe that, assuming G is reduced with respect to Rule 1, u and v have neighbors in U. Now we call an edge cluster $\{u, v\}$ *simple* if u has only neighbors in some U_i and v has only neighbors in some U_j (possibly $i = j$). Observe that the number of non-simple edge clusters is already bounded: Each such cluster is ambiguous because at least one of its vertices is ambiguous. By Lemma 2, there are $O(|U|^2)$ such vertices and thus $O(|U|^2)$ clusters containing them. To obtain a bound on the overall number of edge clusters, we show that all simple edge clusters can be removed.

Reduction Rule 7. *If there is a simple edge cluster $\{u, v\}$, then remove u and v from G.*

Proof (of safeness). We show that $G - \{u, v\}$ has a matching cut if and only if G has a matching cut.

First, assume that $G-\{u, v\}$ has matching cut. Let U_i be the monochromatic part that contains the neighbors of u in U and let U_j be the monochromatic part that contains the neighbors of v in U. We distinguish two cases with respect to U_i and U_j. First, consider the case that U_i and U_j are on the same side of the cut of $G - \{u, v\}$, without loss of generality $(U_i \cup U_j) \subseteq A$. Then, $(A \cup \{u, v\}, B)$ is a matching cut in G: u and v have only neighbors in A, every other vertex in A has at most one neighbor in B and every vertex in B has at most one neighbor in A.

Now consider the case that U_i and U_j are on different sides of the cut, without loss of generality assume $U_i \subseteq A$ and $U_j \subseteq B$. By assumption, we have $(N(u) \setminus \{v\}) \subseteq U_i$ and $(N(v) \setminus \{u\}) \subseteq U_j$. This implies that $(A \cup \{u\}, B \cup \{v\})$ is a matching cut in G: The only additional edge between $A \cup \{u\}$ and $B \cup \{v\}$ is $\{u, v\}$.

Conversely, assume G has a matching cut (A, B). Since Rule 1 does not apply, we have that A is not a subset of $\{u, v\}$ and, symmetrically B is not a subset of $\{u, v\}$. In other words, $A \setminus \{u, v\} \neq \emptyset$ and $B \setminus \{u, v\} \neq \emptyset$ and thus $(A \setminus \{u, v\}, B \setminus \{u, v\})$ is a matching cut in $G - \{u, v\}$. □

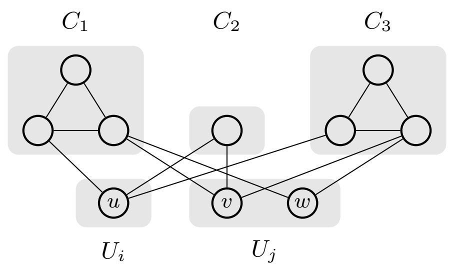

Figure 3: Rule 8 applies to the vertices u and v; an application will merge U_i and U_j. The value of $N(u, w)$ used in Lemma 3 is two: C_1 and C_3 are ambiguous and each of them contains two distinct vertices that are neighbors of u and w, respectively.

Thus, with the above rules we obtain a kernel with $O(\mathrm{dc}(G)^3)$ vertices: a reduced instance has $O(|U|^2) = O(\mathrm{dc}(G)^2)$ clusters, each containing $O(|U|) = O(\mathrm{dc}(G))$ vertices. To obtain a kernel with an overall quadratic number of vertices, we observe first that after exhaustive application of Rules 2 and 6, every vertex in a cluster C, where $|C| \geq 3$ and $C \subseteq N^2(U_i)$ for some i, has at most one neighbor in any monochromatic set U_j. Thus, it remains to further reduce the number of vertices in $V \setminus U$ with only one neighbor in any monochromatic set U_i. This is achieved by the following reduction rule which finds monochromatic parts of U that can be merged not because they have many common neighbors but, instead, because they have common neighbors in several nonedge clusters; a situation in which the rule applies is illustrated in Figure 3.

Reduction Rule 8. *If there are two vertices $u \in U_i$ and $u' \in U_j$, $i \neq j$, and three distinct nonedge clusters C_1, C_2, and C_3 such that u and u' have at least one neighbor in each of them, then merge U_i and U_j.*

Proof (of safeness). We show that $U_i \cup U_j$ is monochromatic; by definition this implies that merging U_i and U_j is safe. Let (A, B) be any matching cut of G. Each of the three clusters is monochromatic because they are nonedge clusters. Hence, we can assume without loss of generality, that A contains C_1 and C_2. Since u has neighbors in C_1 and in C_2, it has two neighbors in A and thus $U_i \subseteq A$. Similarly, $U_j \subseteq A$. Hence, $U_i \cup U_j$ is in the same part of the cut. This holds for all cuts, making $U_i \cup U_j$ monochromatic. $\square$

Lemma 3. *After exhaustive application of Rules 1–8, there are $O(|U|^2)$ vertices in $V \setminus U$ that are in nonedge clusters and have only one neighbor in U.*

Proof. First, by Lemma 1, there are $O(|U|)$ fixed nonedge clusters. By Lemma 2, these clusters contain $O(|U|)$ vertices each. Thus, the number of vertices in fixed nonedge clusters that have only one neighbor in U is $O(|U|^2)$.

Hence, it remains to bound the number of vertices that have only one neighbor in U and are contained in a nonfixed cluster C. By Proposition 3, these clusters are ambiguous. Each ambiguous cluster C contains an ambiguous vertex with neighbors in U_i and U_j, where $i \neq j$. Thus, for each vertex of C with only one neighbor $u \in U$, there is at least one further vertex $u' \in U$ such that u' has at least one neighbor in C, and u and u' are not from the same monochromatic part U_ℓ (because u is in at most one of U_i and U_j). Now, for each $u \in U$ and $u' \in U$ that are in distinct monochromatic parts of U, let $N(u, u')$ denote the number of vertex pairs $\{v, v'\}$ such that there is an ambiguous cluster C containing v and v', one of v and v' is adjacent to u, and the other is adjacent to u'. By the above discussion, any vertex in an ambiguous cluster C with exactly one neighbor in U increases the number $N(u, u')$ for at least one pair of vertices u and u'. By the pigeonhole principle, if there are more than $3 \cdot \binom{|U|}{2}$ vertices in ambiguous nonedge clusters that have only one neighbor in U, then there is some pair of vertices u and u' such that $N(u, u') \geq 3$. Since u and u' each have at most one neighbor in every ambiguous cluster, this means that there are three distinct clusters C_1, C_2, and C_3 which contain neighbors of u and u'. Because u and u' are from distinct monochromatic parts, Rule 8 applies, which contradicts the fact that G is reduced with respect to this rule. Consequently, the number of vertices in ambiguous nonedge clusters that have only one neighbor in U is $O(|U|^2)$. □

Theorem 1. Matching Cut *admits a problem kernel with* $O(\mathrm{dc}(G)^2)$ *vertices that can be computed in* $O(\mathrm{dc}(G)^2 \cdot (n^2 + nm))$ *time.*

Proof. Note that every instance contains $O(\mathrm{dc}(G))$ vertices in U because we may assume that $|U| \leq 3\mathrm{dc}(G)$. To obtain the kernel, we need to reduce the size of $V \setminus U$. To this end, we first apply exhaustively Rules 1–8. Afterwards, $V \setminus U$ has $O(\mathrm{dc}(G)^2)$ vertices: By Lemma 2, $V \setminus U$ has $O(\mathrm{dc}(G)^2)$ ambiguous vertices. Moreover, since G is reduced with respect to Rule 7, we have that every edge cluster contains an ambiguous vertex. Hence, the number of edge clusters, and therefore the number of vertices in edge clusters, is $O(\mathrm{dc}(G)^2)$. It remains to bound the number of vertices in nonedge clusters that are not ambiguous. Each of these vertices has only one neighbor in U because G is reduced with respect to Rule 6. By Lemma 3, $V \setminus U$ has $O(\mathrm{dc}(G)^2)$ vertices that are in nonedge clusters and have only one neighbor in U.

Finally, the number of vertices that have no neighbors in any set U_i is $O(1)$ for each cluster because G is reduced with respect to Rule 5, and thus $O(\mathrm{dc}(G)^2)$ overall.

It remains to bound the running time for the application of the rules. Rule 1 can be applied in time $O(n+m)$ and applies only once. For the remaining rules, we need to maintain the set $N^2(U_i)$ for each U_i. These sets can be computed in $O(\mathrm{dc}(G)(n+m))$ time. Afterwards, the applicability of each rule can be tested in $O(\mathrm{dc}(G)^2(n+m))$ with the bottleneck being Rule 8. Moreover, each rule can be applied in $O(n)$ time, with the exception of Rule 4 which may take $\Theta(n^2)$ time, because it may add $\Theta(n^2)$ many edges. To make this rule more efficient, we observe that during the kernelization it suffices to know which vertices belong to the same cluster, since each cluster induces a complete subgraph. Consequently, we may store the edges in each cluster only implicitly. This gives a running time bound of $O(n)$ also for Rule 4; we omit the technical details. To obtain the claimed bound on the running time it is thus sufficient to bound the number of applications of the rules by $O(n)$: All rules that merge monochromatic parts can be performed $O(\mathrm{dc}(G)) = O(n)$ times overall. For the remaining rules, we have that Rule 4 can be performed $O(n)$ times, because it decreases the number of clusters in $G - U$ by one, Rules 5 and 7 can be performed $O(n)$ times, because they remove at least one vertex from G. Rule 6 can be performed $O(n)$ times as well: First, Rule 6 is performed at most n on the at most n initial clusters. Afterwards, any application of Rule 6 is caused by a previous application of Rule 4, which merges two clusters, or by an application of one of the rules that merge monochromatic parts. By applying Rule 4 exhaustively before applying Rule 6, we have that every monochromatic part U_i has at most one cluster of size at least three that is contained in $N^2(U_i)$. Hence, there is an injective mapping from noninitial applications of Rule 6 to the $O(n)$ many applications of some rule that merges clusters or monochromatic parts. Consequently, Rule 6 is applied $O(n)$ times in total. $\square$

It is worth noting that if $G - U$ consists of only one cluster, that is, $G - U$ is a clique C, then Lemma 2 shows that C can be reduced to contain $O(|U|)$ many vertices.

Corollary 1. Matching Cut *admits a linear kernel when parameterized by* $\mathrm{dq}(G)$*, the distance to clique.*

3. Single-exponential FPT Algorithms

In this section, we consider Matching Cut parameterized by the distance to cluster and by the distance to co-cluster. Recall that co-cluster graphs are precisely the complete multipartite graphs. We show that Matching Cut can be solved in time $2^{\mathrm{dc}(G)}O(n^2)$ and in time $2^{\mathrm{d\overline{c}}(G)}O(nm)$. While Gomes and Sau [15] presented a direct dynamic programming algorithm for the more general d-Cut problem, this algorithm has a worse running time of $4^{\mathrm{dc}(G)}O(\mathrm{dc}(G)^2 n^2)$ for Matching Cut. Recall that we may assume that all graphs considered are connected, have minimum degree at least two and that a clique Q is monochromatic if $|Q| \neq 2$. Finally, observe that minimum distance to cluster sets and minimum distance to co-cluster sets can be computed in $1.92^{\mathrm{dc}(G)} \cdot O(n^2)$ time and $1.92^{\mathrm{d\overline{c}}(G)} \cdot O(n^2)$ time, respectively [6].

3.1. Distance to Cluster

Next, we provide an FPT algorithm solving Matching Cut running in single-exponential time $2^{\mathrm{dc}(G)}O(n^2)$.

Lemma 4. *Let $U \subset V(G)$ such that $G - U$ is a cluster graph. Given a partition (A, B) of U, it can be decided in time $O(n^2)$ if G has a matching cut (X, Y) such that $A \subseteq X$ and $B \subseteq Y$.*

Proof. We first consider the case where A or B is empty, say $B = \emptyset$. Thus, $A = U$ and we are searching for a matching cut (X, Y) such that $U \subseteq X$. If $G - U$ has some cluster Q such that $(Q, V(G) \setminus Q)$ is a matching cut, then we are done. Otherwise, consider some matching cut (X, Y) such that $U \subseteq X$. For each cluster Q of $G - U$ we have $Q \subseteq X$ or $Q \subseteq Y$: For the clusters of size at least three and those of size one this is true because they are monochromatic in G. For each cluster $\{u, v\}$ of size two observe the following. Since $(V(G) \setminus \{u, v\}, \{u, v\})$ is not a matching cut, either

- u and v have a common neighbor in U, or
- $|N(u) \cap U| \geq 2$ or $|N(v) \cap U| \geq 2$.

In the first case, $\{u, v\}$ is monochromatic and thus we have $\{u, v\} \subseteq X$. In the second case u and v are in the same part of the cut as U: One of u and v, say u has two neighbors in U, thus $u \in X$. Now v has two neighbors in X, one neighbor is u and the other neighbor is some vertex in U. Consequently, $\{u, v\} \subseteq X$ in this case as well. Summarizing, $U \subseteq X$ and for each cluster Q of $G - U$ we have $Q \subseteq X$ or $Q \subseteq Y$. Since there is no matching cut between U and a cluster Q of $G - U$, this implies $Q \subseteq X$. Hence, $U \cup (V(G) \setminus U) \subseteq X$ and thus $Y = \emptyset$. Therefore, G has no matching cut (X, Y) such that $U \subseteq X$.

Now assume that A and B are nonempty. We let $F := V(G) \setminus (A \cup B)$ denote the *free* vertices, the vertices that we have not assigned to A or B. The algorithm first applies Rules 9–12 given in [19] to recognize situations where there is no matching cut or to identify vertices that can be moved from F to A or B; the correctness of these rules is easy to see, some of them are illustrated in Figure 4.

Reduction Rule 9. *If an A-vertex has two B-neighbors or a B-vertex has two A-neighbors, then STOP: "G has no matching cut separating A, B".*

Reduction Rule 10. *(1) If $v \in F$, $|N(v) \cap A| \geq 2$, and $|N(v) \cap B| \geq 2$, then STOP: "G has no matching cut separating A, B". (2) If $v \in F$ and $|N(v) \cap A| \geq 2$, then $A := A \cup \{v\}$. (3) If $v \in F$ and $|N(v) \cap B| \geq 2$, then $B := B \cup \{v\}$.*

Reduction Rule 11. *(1) If $v \in A$ has two adjacent F-neighbors w_1, w_2, then $A := A \cup \{w_1, w_2\}$. (2) If $v \in B$ has two adjacent F-neighbors w_3, w_4, then $B := B \cup \{w_3, w_4\}$.*

Reduction Rule 12. *(1) If there is an edge xy in G such that $x \in A$, $y \in B$, and $N(x) \cap N(y) \cap F \neq \emptyset$, then STOP: "$G$ has no matching cut separating A, B".*
(2) If there is an edge xy in G such that $x \in A$ and $y \in B$, then add $N(x) \cap F$ to A, and add $N(y) \cap F$ to B.

If none of the Rules 9–12 can be applied, then

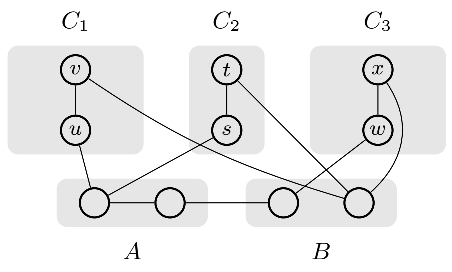

Figure 4: The application of the reduction rules and a resulting 2-Sat formula for a partition (A, B) of U. The vertex w will be added to B by Rule 12 (2) since its neighbor in B is incident with an edge between A and B. Afterwards, the vertex x will be added to B by Rule 10 (3) since it now has two neighbors in B. After these reduction rules, the set F contains four vertices s, t, u, v all of which belong to R. In addition to the clauses ensuring that each vertex goes to A or B, we add clauses $(\neg u_B \vee \neg v_A)$, $(\neg s_B \vee \neg t_A)$, $(\neg u_B \vee \neg s_B)$, and $(\neg v_A \vee \neg t_A)$. The satisfying assignment implied by $u_A = v_B = s_A = t_B =$ `true` corresponds to a matching cut with $\{u, s\} \subseteq A$ and $\{v, t\} \subseteq B$.

- the A, B-edges of G form a matching cut in $G[A \cup B] = G - F$ due to Rule 9,
- every F-vertex is adjacent to at most one A- and at most one B-vertex due to Rule 10,
- the F-neighbors of any A-vertex and the F-neighbors of any B-vertex form an independent set due to Rule 11, and
- every A-vertex adjacent to a B-vertex has no F-neighbor and every B-vertex adjacent to an A-vertex has no F-neighbor due to Rule 12.

Note that $G[F]$ is a cluster graph. Let $\mathcal{Q}$ be the set of all monochromatic clusters in $G[F]$ and let $R := F \setminus \bigcup_{Q \in \mathcal{Q}} V(Q)$. That is, each member in $\mathcal{Q}$ is a cluster with one vertex or a cluster with at least three vertices, and each vertex in R belongs to a 2-vertex cluster in $G[F]$. Now, create a boolean formula ϕ as follows:

- For each $Q \in \mathcal{Q}$ we have two boolean variables Q_A and Q_B (indicating all vertices of Q should be added to A, respectively, to B).
- For each vertex $u \in R$ we have two boolean variables u_A, u_B (indicating u should be added to A, respectively, to B).

The clauses of ϕ are as follows:

(c1) For each $Q \in \mathcal{Q}$: $(Q_A \vee Q_B)$, $(\neg Q_A \vee \neg Q_B)$. These clauses ensure that Q will be moved to A or else to B.

(c2) For each vertex $u \in R$: $(u_A \vee u_B)$, $(\neg u_A \vee \neg u_B)$. These clauses ensure that u will be moved to A or else to B.

(c3) For each two adjacent vertices $u, v \in R$:

(c3.1) If u has neighbors in A and B, if v has neighbors in A and B, if u and v have neighbors in A, or if u and v have neighbors in B: $(u_A \leftrightarrow v_A)$, $(u_B \leftrightarrow v_B)$. Each of these nonstandard clauses corresponds to adding two standard 2-Sat clauses. For example, $(u_A \leftrightarrow v_A)$ is equivalent to $(u_A \vee \neg v_A) \wedge (\neg u_A \vee v_A)$. They ensure that either u and v are both moved to A, or both are moved to B.

(c3.2) If $|N(u) \cap A| = |N(v) \cap B| = 1$ and $N(u) \cap B = N(v) \cap A = \emptyset$: $(\neg u_B \vee \neg v_A)$. This clause ensures that in case u goes to B, v must also go to B, and in case v goes to A, u must also go to A.

(c3.3) If $|N(u) \cap B| = |N(v) \cap A| = 1$ and $N(u) \cap A = N(v) \cap B = \emptyset$: $(\neg u_A \vee \neg v_B)$. This clause ensures that in case u goes to A, v must also go to A, and in case v goes to B, u must also go to B.

(c4) For $z, z' \in \mathcal{Q} \cup R$:

(c4.1) If z, z' have a common neighbor in A: $(\neg z_B \vee \neg z'_B)$. This clause ensures that in this case, z or z' must go to A.

(c4.2) If z, z' have a common neighbor in B: $(\neg z_A \vee \neg z'_A)$. This clause ensures that in this case, z or z' must go to B.

Then ϕ is the conjunction of all these clauses over all $Q \in \mathcal{Q}$ and all $u \in R$, see Figure 4 for some example clauses. We now show the equivalence of ϕ and the problem of finding a matching cut with A on one side and B on the other.

Claim: G has a matching cut (X, Y) with $A \subseteq X$ and $B \subseteq Y$ if and only if ϕ is satisfiable.

Suppose first that (X, Y) is a matching cut of G such that $A \subseteq X$ and $B \subseteq Y$. Then, clearly, $Q \subseteq X$ or $Q \subseteq Y$ for any $Q \in \mathcal{Q}$. Define an assignment for ϕ as follows.

- for each $v \in R$ set $v_A = \texttt{true}$, $v_B = \texttt{false}$ if $v \in X$ and $v_A = \texttt{false}$, $v_B = \texttt{true}$ if $v \in Y$,
- for each $Q \in \mathcal{Q}$ set $Q_A = \texttt{true}$, $Q_B = \texttt{false}$ if $Q \subseteq X$ and $Q_A = \texttt{false}$, $Q_B = \texttt{true}$ if $Q \subseteq Y$.

It is a routine to check that ϕ is satisfied by this assignment.

Conversely, suppose that there is a satisfying assignment for ϕ. Then set

$$X := A \cup \{v \in R : v_A \text{ is } \texttt{true}\} \cup \bigcup_{Q \in \mathcal{Q},\, \mathcal{Q}_A = \texttt{true}} Q,$$

$$Y := B \cup \{v \in R : v_B \text{ is } \texttt{true}\} \cup \bigcup_{Q \in \mathcal{Q},\, \mathcal{Q}_B = \texttt{true}} Q.$$

Observe first that, as the clauses (c1), (c2) are satisfied, $V(G) = X \cup Y$ and $X \cap Y = \emptyset$. Next suppose, by contradiction, that $u \in X$ has two neighbors

v, w in Y. If $u \in A$, then $v, w \in F$ (as Rules 9 and 12 are not applicable) and v, w are non-adjacent (as Rule 11 is not applicable). But this contradicts the assumption that the clauses (c4.1) are satisfied. If $u \in F$, then at least one of v, w is in F (as Rule 10 is not applicable). If both v, w are in F, then u, v, w belong to a clique $Q \in \mathcal{Q}$ (as $G[F]$ is a cluster graph). But this contradicts the assumption that the clauses (c1) are satisfied. So, let $v \in F$ and $w \in B$, say. Since the clauses (c4.1) are satisfied and $u \in X, v \in Y$, it holds that $u, v \in R$. Since the clauses (c3.1) are satisfied, $N(u) \cap A = \emptyset$ and $N(v) \cap B = \emptyset$. Since $\deg(v) \geq 2$, v has a neighbor in A. But this contradicts the assumption that the clauses (c3.3) are satisfied. Thus, no vertex in X can have two neighbors in Y, and similarly, no vertex in Y can have two neighbors in X, which proves the claim.

The length of the formula ϕ is $O(n^2)$. Since 2-SAT can be solved in linear time (cf. [3, 10, 12]), the above discussion yields an $O(n^2)$-time algorithm for deciding if G has a matching cut (X, Y) such that $A \subseteq X$ and $B \subseteq Y$. $\square$

Running the algorithm of Lemma 4 for all partitions (A, B) of U, where U is a minimum distance to cluster set of the input graph G, one obtains

Theorem 2. MATCHING CUT *can be solved in time* $2^{\mathrm{dc}(G)}O(n^2)$.

3.2. Distance to Co-cluster

We now provide an FPT algorithm solving MATCHING CUT running in single-exponential time $2^{\mathrm{d\overline{c}}(G)}O(nm)$.

Lemma 5. *Let* $U \subset V(G)$ *such that* $F = V(G) \setminus U$ *induces a co-cluster graph. Given a partition* (A, B) *of* $G[U]$, *it can be decided in time* $O(nm)$ *if* G *has a matching cut* (X, Y) *such that* $A \subseteq X$ *and* $B \subseteq Y$.

PROOF. Let $F_1, \ldots, F_t$ be the maximal independent sets of $G[F]$ (thus, $F_1, \ldots, F_t$ are the cliques in $\overline{G[F]}$). If $t = 1$ (thus, F is an independent set) or $|F_i| = 1$ for all $1 \leq i \leq t$ (thus, F is a clique), then $G[F]$ is a cluster graph, and we proceed in time $O(n^2)$ similarly as in the case of distance to cluster. So, let $t \geq 2$ and assume $|F_1| \leq |F_2| \leq \cdots \leq |F_t|$ and $|F_t| \geq 2$. We distinguish three cases.

CASE 1. $t \geq 3$, or $t = 2$ and $|F_1| \geq 2, |F_2| \geq 3$.
In this case, F is monochromatic. Thus, G has a matching cut (X, Y) such that $A \subseteq X$ and $B \subseteq Y$ if and only if $X = A \cup F$ and $Y = B$, or $X = A$ and $Y = B \cup F$. As testing if (X, Y) is a matching cut can be done in linear time $O(n + m)$, Case 1 is settled.

CASE 2. $t = 2$ and $|F_1| = |F_2| = 2$.
In this case $G[F]$ is a 4-cycle, and we have six possible extensions of (A, B) to a matching cut (X, Y). Thus, Case 2 can be settled in linear time.

CASE 3. $t = 2$ and $|F_1| = 1$.
Let $F_1 = \{u\}$, $F_2 = \{v_1, \ldots, v_r\}$ for some $r \geq 2$. Thus, $G[F]$ is a star with r edges uv_i, $1 \leq i \leq r$. In this case, G has a matching cut (X, Y) such that $A \subseteq X$ and $B \subseteq Y$ if and only if

- $X = A \cup F$ and $Y = B$, or
- $X = A$ and $Y = B \cup F$, or
- $X = A \cup \{v_i\}$ and $Y = B \cup F \setminus \{v_i\}$ for some $1 \leq i \leq r$, or
- $X = A \cup F \setminus \{v_i\}$ and $Y = B \cup \{v_i\}$ for some $1 \leq i \leq r$.

Thus, we can decide in time $O(r)O(n + m) = O(nm)$ if one of these $2r + 2$ partitions (X, Y) is a matching cut extending (A, B). □

Running the algorithm of Lemma 5 for all partitions (A, B) of $G[U]$, where U is a minimum distance to co-cluster set of the input graph G, one obtains

Theorem 3. Matching Cut *can be solved in time* $2^{\mathrm{d}\overline{\mathrm{c}}(G)}O(nm)$.

4. Improved Exact Exponential Algorithms

In this section, we give two exact exponential algorithms for Matching Cut both improve the best known running time of $O^*(1.4143^n)$. The first one is Sat-based and runs in time $O^*(1.3280^n)$ or in randomized time $O^*(1.3071^n)$.[2] The second one is a branching algorithm without any Sat-subroutine and runs in time $O^*(1.3803^n)$. Though the running time of the branching algorithm is worse than for the Sat-based algorithm, the used branching rules and the idea of the termination lemma in the analysis are interesting on their own. Moreover, it is interesting to know how fast a Sat-free algorithm can be compared to a Sat-based algorithm for Matching Cut. In fact, very recently, a similar branching algorithm for Matching Cut restricted to graphs with minimum degree at least 469 which has running time $O^*(1.0099^n)$ has been obtained in [18]. Given the current knowledge, such a running time for solving Matching Cut cannot be achieved by reducing to 3-Sat.

4.1. Sat-based Algorithms

Deciding if a graph $G = (V, E)$ has a matching cut can be reduced in polynomial time to $O(n^2)$ 3-Sat instances each of which has $n = |V|$ variables. The reduction is as follows. Let $G = (V, E)$ be a connected graph with minimum degree at least two. Construct a 3-Cnf formula as follows. For each vertex $v \in V$ we create a Boolean variable x_v. For each two neighbors u and w of v, we introduce a clause $\neg(x_v \wedge (\neg x_u \wedge \neg x_w)) \equiv (\neg x_v \vee x_u \vee x_w)$, meaning that it is not allowed that v and its two neighbors u and w belong to different parts of a matching cut of G. Now, fix two vertices $a, b \in V$ and let

$$\phi(a, b) = (x_a) \wedge (\neg x_b) \wedge \bigwedge_{v \in V} \bigwedge_{u,w \in N(v)} (\neg x_v \vee x_u \vee x_w).$$

[2]This algorithm was suggested by a reviewer; we are deeply grateful for her or his suggestion.

Obviously, $\phi(a,b)$ can be constructed in time $O\left(\sum_{v\in V}\binom{\deg(v)}{2}\right) = O(n^3)$. Furthermore, G has a matching cut separating a and b if and only if the formula $\phi(a,b)$ is satisfiable: If (A,B) is a matching cut of G with $a \in A$ and $b \in B$, then $\phi(a,b)$ is satisfied by setting $x_v = \texttt{true}$ for $v \in A$ and $\texttt{false}$ for $v \in B$. Conversely, if there is a satisfying assignment for $\phi(a,b)$, then (A,B) with $A = \{v \in V \mid x_v = \texttt{true}\}$ and $B = \{v \in V \mid x_v = \texttt{false}\}$ is a matching cut of G separating a and b.

Thus, by using the fastest known deterministic algorithm [22] and the fastest known randomized algorithm for 3-Sat [26, 17], we obtain

Theorem 4. *There is a (deterministic) $O^*(1.328^n)$-time* Sat*-based algorithm and a randomized $O^*(1.3071^n)$-time* Sat*-based algorithm for* Matching Cut*.*

Note that the same reduction applies to d-Cut, that is, any instance $G = (V,E)$ of d-Cut can be reduced in polynomial time to $O(|V|^2)$ instances of $(d+2)$-Sat such that G has a d-cut if and only if at least one of these $(d+2)$-Cnf formulas is satisfiable. Since k-Sat can be solved in $2^{(1-\frac{c}{k-1})n}$ time for some constant $c > 1$ [26], this implies that d-Cut can be solved in time $O^*(2^{(1-\frac{c}{d+1})n})$ for some constant $c > 1$.

4.2. A Sat*-free Branching Algorithm*

Our algorithm takes as input a graph $G = (V,E)$ and decides whether or not there is an edge set $M \subseteq E$ such that M is a matching cut of G. As above, we may assume that G is connected and has minimum degree at least two. The idea is to compute a partition of the vertex set into subsets A and B such that A and B are nonempty unions of components of $G - M$ and all M-edges have one endvertex in A and the other in B. Our algorithm consists of reduction rules and branching rules that label the vertices of the input graph by either A or B but never change the graph G. Finally we provide a termination lemma stating that if neither a reduction rule nor a branching rule can be applied, then there is a matching cut in the graph G, respecting the current partial partition into A and B.

The branching algorithm below will be executed for all possible pairs $a, b \in V$, hence $O(n^2)$ times. To do this set $A := \{a\}$, $B := \{b\}$, and $F := V \setminus \{a,b\}$ and call the branching algorithm. At each stage of the algorithm, A and/or B will be extended or it will be determined that there is no matching cut that separates A from B.

We describe our algorithm by a list of reduction and branching rules given in preference order, that is, in an execution of the algorithm on any instance of a subproblem one always applies the first rule applicable to the instance, which could be a reduction or a branching rule. A reduction rule produces one subproblem while a branching rule results in at least two subproblems, with different extensions of A and B. Note that G has a matching cut that separates A from B if and only if in at least one recursive branch, extensions A' of A and B' of B are obtained such that G has a matching cut that separates A' from

B'. Typically a rule assigns one or more free vertices, vertices of F, either to A or to B and removes them from F, that is, we always have $F := V \setminus (A \cup B)$.

First our algorithm applies Reduction Rules 9–12 mentioned in Section 3 to the current instance (in the order of the rules). In addition, we need two new reduction rules.

Reduction Rule 13. *If there are vertices $u, v \in F$ such that $N(u) = N(v) = \{x, y\}$ with $x \in A, y \in B$, then $A := A \cup \{u\}$, $B := B \cup \{v\}$.*

Proof (of safeness). We show that G has a matching cut separating A and B if and only if G has a matching cut separating $A \cup \{u\}$ and $B \cup \{v\}$.

First, assume that (X, Y) is a matching cut of G with $A \subseteq X$ and $B \subseteq Y$. If $u \in X$, then v must belong to Y, hence (X, Y) is a matching cut separating $A \cup \{u\}$ and $B \cup \{v\}$. If $u \in Y$, then v must belong to X. In this case, as $N(u) = N(v) = \{x, y\}$, the cut $(X \setminus \{v\} \cup \{u\}, Y \setminus \{u\} \cup \{v\})$ is a matching cut. This matching cut clearly separates $A \cup \{u\}$ and $B \cup \{v\}$.

The other direction can be seen as follows: any matching cut of G separating $A \cup \{u\}$ and $B \cup \{v\}$ separates A and B, too. □

Reduction Rule 14. *(1) If there are vertices $u, v \in F$ such that $N(u) = N(v) = \{x, y\}$ with $x \in A, y \in F$, then $A := A \cup \{u\}$.*
(2) If there are vertices $u, v \in F$ such that $N(u) = N(v) = \{x, y\}$ with $x \in F, y \in B$, then $B := B \cup \{v\}$.

Proof (of safeness). The proof is similar to the safeness proof for Reduction Rule 13. We show the proof for part (1), the proof for part (2) is symmetric.

First, assume that (X, Y) is a matching cut of G with $A \subseteq X$ and $B \subseteq Y$. If $u \in X$, then (X, Y) is clearly a matching cut separating $A \cup \{u\}$ and B. If $u \in Y$, then v must belong to X. In this case, as $N(u) = N(v) = \{x, y\}$, the cut $(X \setminus \{v\} \cup \{u\}, Y \setminus \{u\} \cup \{v\})$ is a matching cut which separates $A \cup \{u\}$ and B.

The other direction can be seen as follows: any matching cut of G separating $A \cup \{u\}$ and B separates A and B, too. □

Our algorithm consists of six branching rules dealing with small *configurations*. These are connected (not necessarily induced) subgraphs with at most eight vertices, some of them already belonging to A or B. The six configurations (B1)–(B6) are shown in Figure 5; each configuration corresponds to one branching rule. The Branching Rules (B1) and (B2) are new, the last four have been used in [19]. Moreover, compared to [19] we do a better branching on configuration (B5).

To determine the branching vectors which correspond to our branching rules, we set the size of an instance (G, A, B) as its number of free vertices, that is, $|V(G)| - |A| - |B|$.

Branching Rule (B1). *We branch into two subproblems. First, add v_1 to B. Then v_2 has to be added to B and v_3 has to be added to A. Second, add v_1 to A. Then v_2 has to be added to A too. Hence the branching vector is $(3, 2)$.*

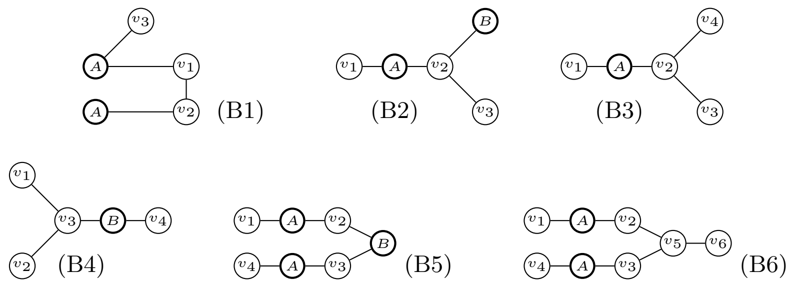

Figure 5: Branching configurations: Vertices with label A and B belong to A, respectively, to B; vertices with labels v_i are in F.

Branching Rule (B2). *We branch into two subproblems. First, add v_2 to B. Then v_1 has to be added to A and v_3 has to be added to B. Second, add v_2 to A. Then v_3 has to be added to A too. Hence the branching vector is* $(3, 2)$.

Branching Rule (B3). *We branch into two subproblems. First, add v_2 to B. This implies that v_1 has to be added to A, and that v_3 and v_4 have to be added to B. Second, add v_2 to A. Hence the branching vector is* $(4, 1)$.

Branching Rule (B4). *On this configuration we branch into two subproblems, similar to (B3). First, add v_3 to A. This implies that v_4 has to be added to B, and that v_1 and v_2 have to be added to A. Second, add v_3 to B. Hence the branching vector is* $(4, 1)$.

Branching Rule (B5). *We branch into two subproblems. First, add v_2 to A. This implies that v_3 has to be added to B and then v_4 has to be added to A. Second, add v_2 to B. Then v_1 must be added to A. Hence the branching vector is* $(3, 2)$.

Branching Rule (B6). *We branch into four subproblems. First, add v_5 to A. This implies that v_2 and v_3 have to be added to A. Next, add v_5 to B. There are three choices to label v_2 and v_3: AB, BA, BB. In the first two choices, v_6 has to be added to B and v_1 or v_4 has to be added to A. In the last choice, v_1 and v_4 have to be added to A. Hence the branching vector is* $(3, 5, 5, 5)$.

The branching numbers of the branching vectors of our algorithm are 1.3803 (Branching Rules (B3) and (B4)), 1.3734 (Branching Rule (B6)) and 1.3248 (Branching Rules (B1), (B2), and (B5)). Consequently, the running time of our algorithm is $O^*(1.3803^n)$.

Termination Lemma. Now, the question is: What happens in case any of the reduction and branching rules is not longer applicable to the current instance (G, A, B)?

Note that $(A, B \cup F)$ is already a matching cut of G, or else the set A' of vertices in A having at least two neighbors in F is nonempty. We will see that if

$N(A') \cap F$ contains a vertex of degree at least three, then there exists a configuration (B2) or (B3), and the branching process will continue. In particular, if none of the branching rules is applicable, then every vertex in $N(A') \cap F$ must have degree two. In this case, we can finally show that G has a matching cut (X, Y) such that $A \subseteq X$ and $B \subseteq Y$. We formalize this situation as follows.

Definition 1. *Let (G, A, B) be an instance of the algorithm. Let $A' = \{a \in A : |N(a) \cap F| \geq 2\}$. A is called final if every vertex in $N(A') \cap F$ has degree two.*

Lemma 6. *If A is not final and none of the reduction rules is applicable to (G, A, B), then (G, A, B) has a configuration (B2) or (B3).*

Proof. If A is not final, then there exists a vertex $v \in N(A') \cap F$ with degree at least three. (Recall that G has minimum degree at least two.) As Reduction Rule 10 is not applicable, v has exactly one neighbor in A. Thus, let $a \in A'$ be the neighbor of v in A. Let $v_1, v_2 \in F \cup B$ be two other neighbors of v. Since Reduction Rule 10 is not applicable at most one of v_1 and v_2 may belong to B. Moreover, since Reduction Rules 11 and 12 are not applicable, a is adjacent to none of v_1 and v_2. Thus, by definition of A', a has another neighbor $u \in F \setminus \{v, v_1, v_2\}$. Now, if both v_1, v_2 are not in B, then a, u, v, v_1, v_2 form configuration (B3). Otherwise, we may assume $v_1 \in B$ without loss of generality which implies that a, u, v, v_1, v_2 form configuration (B2). □

Lemma 7. *Let (G, A, B) be an instance such that all reduction and branching rules are not applicable. Then G has a matching cut (X, Y) such that $A \subseteq X$ and $B \subseteq Y$ (and such a matching cut can be computed in polynomial time).*

Proof. It follows from Lemma 6 that A is final. Hence every vertex in $F' := N(A') \cap F$ has degree two.

Write $F'' = \{u \in F \setminus F' : u \text{ has two neighbors in } F'\}$, and set

$$X = A \cup F' \cup F'' \text{ and } Y = V(G) \setminus X.$$

Note that F' is a stable set (since Reduction Rule 11 and Branching Rule (B1) are not applicable). See Figure 6.

We claim that the edge set (X, Y) is a matching cut.

First we show that every vertex in X has at most one neighbor in Y. This can be seen as follows:

- Every vertex $x \in A$ has at most one neighbor in Y. This is because Reduction Rules 9 and 12 are not applicable, and, by definition of X, x has at most one neighbor in $F \setminus (F' \cup F'')$.

- Every vertex $v \in F'$ has at most one neighbor in Y, as $\deg(v) = 2$ and v has a neighbor in $A' \subseteq A$.

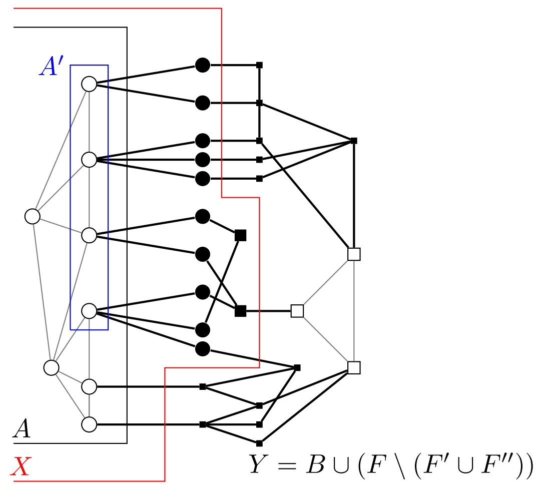

Figure 6: When A is final: F'-vertices: black circles; F''-vertices: big black squares, B-vertices: white squares.

- Every vertex $v \in F''$ has at most one neighbor in Y. By contradiction, assume that v has two neighbors in Y. Then, as Reduction Rule 10 is not applicable, v has a neighbor $u \in Y \setminus B$. Let v_1, v_2 be two neighbors of v in F' and let $a_1, a_2 \in A'$ be the neighbor of v_1 and v_2, respectively. Note that $a_1 \neq a_2$, as otherwise Reduction Rule 14 would be applicable on a_1, v_1, v_2 and v. Now, let $v_i' \in F' \setminus \{v_i\}$ be another neighbor of a_i in F', $i = 1, 2$. Then $a_1, a_2, v_1, v_1', v_2, v_2', v$, and u form configuration (B6), a contradiction. Thus, v has at most one neighbor in Y, as claimed.

Next we show that every vertex in Y has at most one neighbor in X. This can be seen as follows:

- Assume to the contrary that $y \in B$ has two neighbors $u_1, u_2 \in X$. Then, as Reduction Rules 9 and 12 are not applicable, $u_1, u_2 \in F' \cup F''$. First, suppose $u_1, u_2 \in F'$, and let $u_i \in N(a_i)$ for some $a_i \in A'$, $i = 1, 2$. As Reduction Rule 13 is not applicable, $a_1 \neq a_2$. Then $a_1, u_1, u_1', a_2, u_2, u_2'$, and y form configuration (B5), where u_i' is a neighbor of a_i in $F' \setminus \{u_i\}$, a contradiction. So let $u_1 \in F''$, say. Let v_1, v_2 be two neighbors of u_1 in F'. Since Reduction Rule 11 is not applicable, $u_2 \neq v_1, v_2$. But then v_1, v_2, u_1, y, and u_2 form configuration (B4), a contradiction again. Thus, no vertex in B has two neighbors in X.

- Assume to the contrary that $v \in F \setminus (F' \cup F'')$ has two neighbors in X. Consider first the case that a neighbor u of v is in F''. Let v_1, v_2 be two neighbors of u in F' and let $a_i \in A'$ be adjacent to v_i, $i = 1, 2$. Now, as Reduction Rule 14 is not applicable, $a_1 \neq a_2$. Hence $a_1, a_2, v_1, v_1', v_2, v_2', u$, and v form configuration (B6), a contradiction again. Thus, v cannot have a neighbor in F''. Moreover, as $v \notin F''$, v has at most one neighbor in F', and, as Reduction Rule 10 is not applicable, v has at most one neighbor in A. Thus, a neighbor u_1 of v must be in F' and another neighbor u_2 of v must be in A. Then a, u_1, u_1', u_2, v form a configuration (B1), where $a \in A'$ is the neighbor of u_1 (note that $a \neq u_2$ because Reduction Rule 11 is not applicable) and u_1' is another neighbor of a in F', a contradiction. Thus, no vertex in $Y \setminus B$ has two neighbors in X.

We have seen that (X, Y) is a matching cut. □

Theorem 5. *There is a branching algorithm that solves* Matching Cut *in time* $O^*(1.3803^n)$.

5. Conclusions

We provided three algorithms for Matching Cut: a Sat-based exponential algorithm of running time $O^*(1.3071^n)$, a fixed-parameter algorithm of running time $2^{\mathrm{dc}(G)}O(n^2)$ where $\mathrm{dc}(G)$ is the distance to cluster number, and a fixed-parameter algorithm of running time $2^{\mathrm{d\bar{c}}(G)}O(nm)$ where $\mathrm{d\bar{c}}(G)$ is the distance to co-cluster number. Our results improved the $O^*(1.4143^n)$-time exact algorithm and the $2^{\tau(G)}O(n^2)$-time and $O^*(2^{\mathrm{tc}(G)})$-time algorithms previously given in [19] and [2], where $\tau(G) \geq \max\{\mathrm{dc}(G), \mathrm{d\bar{c}}(G)\}$ is the vertex cover number of G and $\mathrm{tc}(G) \geq \max\{\mathrm{dc}(G), \mathrm{d\bar{c}}(G)\}$ is the twin cover number of G. Moreover, we found a quadratic vertex-kernel for Matching Cut for the distance to cluster, and a linear vertex-kernel for the distance to clique.

There are many possible directions for future research. Does Matching Cut admit a linear vertex-kernel for the distance to cluster? Even for the parameter vertex cover number $\tau(G)$, a linear vertex-kernel is open. Moreover, it is open whether the problem admits a polynomial kernel for the distance to forests (known as feedback vertex set number) or the distance to cographs. It would also be interesting to study exact exponential algorithms for the problem of enumerating all matching cuts of a graph and to obtain bounds on their number in general graphs. It is natural to ask whether Matching Cut can be solved faster than via the Sat-based $O^*(1.3071^n)$-time algorithm.

Finally, we recall that the polynomial kernel for the parameter $\mathrm{dc}(G)$ has been generalized from Matching Cut to d-Cut where each vertex in A may have at most d neighbors in B and vice versa [15]. A further study of d-Cut seems to be a promising topic for future research.

References

[1] Araújo, J., Cohen, N., Giroire, F., Havet, F., 2012. Good edge-labelling of graphs. Discr. Appl. Math. 160, 2502–2513.

[2] Aravind, N.R., Kalyanasundaram, S., Kare, A.S., 2017. On structural parameterizations of the matching cut problem, in: Proceedings of the 11th

International Conference on Combinatorial Optimization and Applications (COCOA '17), Springer. pp. 475–482.

[3] Aspvall, B., Plass, M.F., Tarjan, R.E., 1979. A linear-time algorithm for testing the truth of certain quantified boolean formulas. Inf. Process. Lett. 8, 121–123.

[4] Bodlaender, H.L., Jansen, B.M.P., Kratsch, S., 2014. Kernelization lower bounds by cross-composition. SIAM J. Discrete Math. 28, 277–305.

[5] Bonsma, P.S., 2009. The complexity of the matching-cut problem for planar graphs and other graph classes. J. Graph Theory 62, 109–126.

[6] Boral, A., Cygan, M., Kociumaka, T., Pilipczuk, M., 2016. A fast branching algorithm for cluster vertex deletion. Theory Comput. Syst. 58, 357–376.

[7] Borowiecki, M., Jesse-Józefczyk, K., 2008. Matching cutsets in graphs of diameter 2. Theor. Comput. Sci. 407, 574–582.

[8] Chvátal, V., 1984. Recognizing decomposable graphs. J. Graph Theory 8, 51–53.

[9] Cygan, M., Fomin, F.V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S., 2015. Parameterized Algorithms. Springer.

[10] Davis, M., Putnam, H., 1960. A computing procedure for quantification theory. J. ACM 7, 201–215.

[11] Downey, R.G., Fellows, M.R., 2013. Fundamentals of Parameterized Complexity. Texts in Computer Science, Springer.

[12] Even, S., Itai, A., Shamir, A., 1976. On the complexity of timetable and multicommodity flow problems. SIAM J. Comput. 5, 691–703.

[13] Farley, A.M., Proskurowski, A., 1982. Networks immune to isolated line failures. Networks 12, 393–403.

[14] Fomin, F.V., Kratsch, D., 2010. Exact Exponential Algorithms. Springer.

[15] Gomes, G.C.M., Sau, I., 2019. Finding cuts of bounded degree: complexity, FPT and exact algorithms, and kernelization, in: Proceedings of the 14th International Symposium on Parameterized and Exact Computation (IPEC 2019), Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik. Accepted for publication.

[16] Graham, R.L., 1970. On primitive graphs and optimal vertex assignments. Ann. N. Y. Acad. Sci. 175, 170–186.

[17] Hertli, T., 2014. 3-SAT faster and simpler—unique-SAT bounds for PPSZ hold in general. SIAM J. Comput. 43, 718–729.

[18] Hsieh, S., Le, H., Le, V.B., Peng, S., 2019. Matching cut in graphs with large minimum degree, in: Proceedings of the 25th International Conference on Computing and Combinatorics (COCOON '19), Springer. pp. 301–312.

[19] Kratsch, D., Le, V.B., 2016. Algorithms solving the matching cut problem. Theor. Comput. Sci. 609, 328–335.

[20] Le, H., Le, V.B., 2019. A complexity dichotomy for matching cut in (bipartite) graphs of fixed diameter. Theor. Comput. Sci. 770, 69–78.

[21] Le, V.B., Randerath, B., 2003. On stable cutsets in line graphs. Theor. Comput. Sci. 301, 463–475.

[22] Liu, S., 2018. Chain, generalization of covering code, and deterministic algorithm for k-SAT, in: Proceedings of the 45th International Colloquium on Automata, Languages, and Programming (ICALP '18), Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik. pp. 88:1–88:13.

[23] Marx, D., O'Sullivan, B., Razgon, I., 2013. Finding small separators in linear time via treewidth reduction. ACM T. Algorithms 9, 30:1–30:35.

[24] Moshi, A.M., 1989. Matching cutsets in graphs. J. Graph Theory 13, 527–536.

[25] Patrignani, M., Pizzonia, M., 2001. The complexity of the matching-cut problem, in: Proceedings of the 27th International Workshop on Graph-Theoretic Concepts in Computer Science (WG '01), Springer. pp. 284–295.

[26] Paturi, R., Pudlák, P., Saks, M.E., Zane, F., 2005. An improved exponential-time algorithm for k-SAT. J. ACM 52, 337–364.