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

**Christian Komusiewicz**
Fachbereich Mathematik und Informatik, Philipps-Universität Marburg, Marburg, Germany
komusiewicz@informatik.uni-marburg.de
https://orcid.org/0000-0003-0829-7032

**Dieter Kratsch**
Laboratoire de Génie Informatique, de Production et de Maintenance, Université de Lorraine, Metz, France
dieter.kratsch@univ-lorraine.fr

**Van Bang Le**
Universität Rostock, Institut für Informatik, Rostock, Germany
van-bang.le@uni-rostock.de

## 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 $O^*(1.3803^n)$. Moreover, we point out that, unless NP $\subseteq$ coNP/poly, Matching Cut does not admit a polynomial kernel when parameterized by treewidth.

**2012 ACM Subject Classification** Mathematics of computing → Graph theory, Theory of computation → Graph algorithms analysis, Theory of computation → Parameterized complexity and exact algorithms

**Keywords and phrases** matching cut, decomposable graph, graph algorithm

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2018.19

## 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 an (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 ([13, 7]). 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 [11] studied matching cuts in graphs in the context of network applications. Patrignani and Pizzonia [18] pointed out an application of matching cuts in graph drawing. Graphs having no matching cut were first discussed by Graham in [13] under the name *indecomposable graphs*, and have been recently used by Araújo et al. [1] in the context of WDM (Wavelength Division Multiplexing) networks.

**Previous results.** Chvátal [7] 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 [4, 6, 14, 16, 15, 17, 18]. In particular, the NP-hardness of Matching Cut has been further strengthened to planar graphs of maximum degree four [4] and bipartite graphs of maximum degree four [16]. As noted previously [14], the NP-hardness reduction by Chvátal [7] 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 [14] 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 [4]; hence Matching Cut is fixed-parameter tractable when parameterized by $\mathrm{tw}(G)$, the treewidth of $G$. Very recently, Aravind 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 $\mathrm{tc}(G) \leq \tau(G)$ is the size of a smallest twin cover of $G$ [2].

**Our contributions.** We give the first 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 and $\mathrm{d\overline{c}}(G)$ is the distance to co-cluster. This improves upon the FPT algorithms for Matching Cut with running time $2^{\tau(G)}O(n^2)$ [14], where $\tau(G) \geq \max\{\mathrm{dc}(G), \mathrm{d\overline{c}}(G)\}$ 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 $O^*(2^{\mathrm{tc}(G)})$ [2] since $\mathrm{tc}(G) \geq \mathrm{dc}(G)$. Third, we provide an exact branching algorithm for Matching Cut that has time complexity $O^*(1.3803^n)$. This result improves upon the first exact branching algorithm for Matching Cut that has time complexity $O^*(1.4143^n)$ [14].

**Notation and terminology.** Let $G = (V, E)$ be a graph with vertex set $V(G) := V$ and edge set $E(G) := E$. We assume that a (input) graph has $n$ vertices and $m$ edges. 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 $G[V \setminus W]$.

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

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 $C$, i.e., $V \setminus C$ 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 $\mathrm{d\bar{c}}(G)$, 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 $\tau(G) \geq \max\{\mathrm{dc}(G), \mathrm{d\bar{c}}(G)\}$ and $\mathrm{dq}(G) \geq \max\{\mathrm{dc}(G), \mathrm{d\bar{c}}(G)\}$ 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 $U \subseteq V$. Then 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 $G$. Being monochromatic is hereditary: if $G[U]$ is monochromatic, then so is $G[U']$ for every $U' \subseteq U$. 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.

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 $\alpha_i$ is the branching number of branching rule $i$, and the maximum is taken over all branching rules. We refer to [12] for more details on exact branching algorithms. For the basic notions of parameterized complexity we refer to [8].

## 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. First, however, we motivate the study of the parameter $\mathrm{dc}(G)$ by a negative result. Recall that there is an FPT algorithm for Matching Cut when parameterized by treewidth [2, 4]. 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 NP $\subseteq$ coNP/poly.*

The proof of Proposition 1 uses cross-composition and is deferred to the full version of this work.

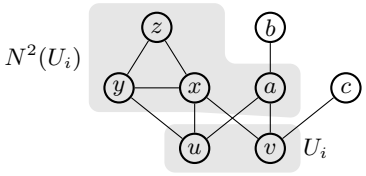


**Figure 1** An example for the definition of $N^2(U_i)$ with $U_i = \{u, v\}$. We have $\{x, a\} \subseteq N^2(U_i)$ since $x$ and $a$ have two neighbors in $U_i$. Moreover, $\{y, z\} \subseteq N^2(U_i)$ since $y$ and $z$ are in a cluster of size at least three with $x$. Alternatively, $\{x, y, z\} \subseteq N^2(U_i)$ 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.

This excludes many natural candidate parameters for kernelization and motivates our study of kernelization for $\mathrm{dc}(G)$, the distance of $G$ to a cluster graph. Let $U = \{u_1, \ldots, u_{|U|}\}$ denote a vertex set such that $G-U$ is a cluster graph. We may assume that $|U| \leq 3\mathrm{dc}(G)$ since a 3-approximation of Cluster Vertex Deletion can be computed in time $O(\mathrm{dc}(G)(n+m))$ 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 $U_1, \ldots, U_\ell$ 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 output "yes".*

The correctness of the rule is obvious. 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. 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 1.** *$U_i \cup N^2(U_i)$ 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 and thus $U_i \cup C$ is monochromatic. ◀

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

▶ **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 1, $\{v\} \cup U_i$ and $\{v\} \cup U_j$ are monochromatic in $G$. Thus, $U_i \cup U_j \cup \{v\}$ is monochromatic. ◀

▶ **Reduction Rule 3.** *If there are three vertices $v_1$, $v_2$, $v_3$ 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 proof is based on 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 a 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. ◀

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

**Proof.** Since $G$ is reduced with respect to Reduction 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. ◀

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 these clusters.*

**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 following is obvious by 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 Reduction 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 $C - v$ is monochromatic in $G'$. This and the fact that $v$ has only neighbors in $C$ immediately implies that $G$ and $G'$ are equivalent. ◀

After application of Rule 5, every vertex in a cluster of size at least three has a neighbor in $U$. The next rule removes unnecessary edges between monochromatic parts and clusters.

▶ **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 $v_3 \in C \setminus \{v_1, v_2\}$. If $|U_i| > 2$, then 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 1, 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. In $G'$, $U_i$ is monochromatic and thus the cluster $C$ is contained in $N^2(U_i)$. By Proposition 1, $U_i \cup C$ is monochromatic in $G'$. As above, since $G$ can be obtained from $G'$ by removing and adding certain edges between vertices in $U_i \cup C$, $G$ also 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 Reduction Rules 1–6. Then $G$ has*

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

The proof of Lemma 2 is deferred to the full version of this work.

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 Reduction Rule 1, $u$ and $v$ have neighbors in $U$. Now we call an edge cluster $\{u, v\}$ *simple* if $u$ has only neighbors in $U_i$ and $v$ has only neighbors in $U_j$ (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 $G$.*

The proof of the safeness of this rule is deferred to the full version of this work.

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 applications

of Reduction Rule 6, every vertex in cluster $C$, where $|C| \geq 3$ and $C \subseteq N^2(U_i)$ for some $i$, has at most one neighbor in $U_i$.

It remains to bound the number of vertices in $V \setminus U$ with only one neighbor in $U$. First, we find 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.

▶ **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 $C_1, C_2, C_3$ 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. ◀

▶ **Lemma 3.** *After exhaustive application of Reduction 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 2, 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 other 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 $U_\ell$ (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 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 $C_1$, $C_2$, $C_3$ which contain neighbors of $u$ and $u'$. Because $u$ and $u'$ are from distinct monochromatic parts, Reduction 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 4.** Matching Cut *admits a problem kernel with $O(\mathrm{dc}(G)^2)$ vertices that can be computed in $O(\mathrm{dc}(G)^3 \cdot (n^2 + nm))$ time.*

**Proof.** Note that every instance contains $O(\mathrm{dc}(G))$ vertices in $U$ because we may assume that $|U| \leq 3\mathrm{dc}(G)$. Furthermore, the number of special vertices is also $O(\mathrm{dc}(G))$ since there is a constant number of them for every monochromatic part. 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 Reduction 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 can however store the edges in each cluster only implicitly, giving a running time bound of $O(n)$ also for this rule; we omit the details. Thus, to obtain the claimed bound on the running time it is sufficient to bound the number of applications of the rules by $O(\mathrm{dc}(G) \cdot n)$: All rules that merge monochromatic parts can be performed $O(\mathrm{dc}(G))$ 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 $G$. ◀

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

▶ **Corollary 5.** 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 $2^{\mathrm{d\bar{c}}(G)}O(nm)$. 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 $O(1.92^{\mathrm{dc}(G)} \cdot n^2)$ time and $O(1.92^{\mathrm{d\bar{c}}(G)} \cdot n^2)$ time, respectively [5].

**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 6.** *Let $U \subset V(G)$ such that $F = V(G) \setminus U$ induces a cluster graph. Given a partition $(A, B)$ of $G[U]$, 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 $A$ 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 $G[F]$ has some connected component $Q$ 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 connected component $Q$ of $G[F]$ we have $Q \subseteq X$ or $Q \subseteq Y$: This is obviously true for the connected components of size at least three and for those of size one because they are monochromatic in $G$. For each

connected component $\{u, v\}$ 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, in the second case $u$ and $v$ are in the same part of the cut as $U$, and thus we have $\{u, v\} \subseteq X$. Summarizing, $U \subseteq X$ and for each connected component $Q$ of $G[F]$ we have $Q \subseteq X$ or $Q \subseteq Y$. Since there is no matching cut between $U$ and a connected component $Q$ of $G[F]$, this implies $Q \subseteq X$. Hence, $F \cup U \subseteq X$ 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. Then the algorithm first applies Reduction Rules (R1) – (R4) given in [14]; the correctness of these rules is easy to see.

**(R1)** If an $A$-vertex has two $B$-neighbors, or a $B$-vertex has two $A$-neighbors then STOP: "$G$ has no matching cut separating $A$, $B$".

**(R2)** 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 $A$, $B$".
If $v \in F$ and $|N(v) \cap A| \geq 2$ then $A := A \cup \{v\}$.
If $v \in F$ and $|N(v) \cap B| \geq 2$ then $B := B \cup \{v\}$.

**(R3)** If $v \in A$ has two adjacent $F$-neighbors $w_1, w_2$ then $A := A \cup \{w_1, w_2\}$.
If $v \in B$ has two adjacent $F$-neighbors $w_3, w_4$ then $B := B \cup \{w_3, w_4\}$.

**(R4)** If there is an edge $xy$ in $G$ such that $x \in A$ and $y \in B$ and $N(x) \cap N(y) \cap F \neq \emptyset$ then STOP: "$G$ has no matching cut separating $A$, $B$".
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 these reduction rules can be applied then

- the $A, B$-edges of $G$ form a matching cut in $G[A \cup B] = G - F$ due to (R1),
- every $F$-vertex is adjacent to at most one $A$- and at most one $B$-vertex due to (R2),
- the $F$-neighbors of any $A$-vertex and the $F$-neighbors of any $B$-vertex form an independent set due to (R3), 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.

Clearly these properties hold for the instance $(G, A, B)$ if none of the Rules (R1) – (R4) can be applied. Note that $G[F]$ is a cluster graph. Let $\mathcal{Q}$ be the set of all monochromatic connected components in $G[F]$ and let $R := F \setminus \bigcup_{Q \in \mathcal{Q}} V(Q)$. That is, each member in $\mathcal{Q}$ is a trivial clique or a clique with at least three vertices, and each vertex in $R$ belongs to a 2-vertex connected component in $G[F]$. Now, create a boolean formula $\phi$ 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 $\phi$ 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 $B$: $(u_A \leftrightarrow v_A)$, $(u_B \leftrightarrow v_B)$. These clauses ensure that either both $u$ and $v$ must be moved to $A$, or both must be 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 $\phi$ is the conjunction of all these clauses over all $Q \in \mathcal{Q}$ and all $u \in R$.

▶ **Proposition 3.** *$G$ has a matching cut $(X, Y)$ with $A \subseteq X$ and $B \subseteq Y$ if and only if $\phi$ is satisfiable.*

The proof of Proposition 3 is deferred to the full version of this work.

Obviously, the length of the formula $\phi$ is $O(n^2)$. Since 2-Sat can be solved in linear time (cf. [3, 9, 10]), 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 $B \subseteq Y$. ◀

Running the algorithm of Lemma 6 for all partitions $(A, B)$ of $G[U]$, where $U$ is a minimum distance to cluster set of the input graph $G$, one obtains

▶ **Theorem 7.** Matching Cut *can be solved in time $2^{\mathrm{dc}(G)}O(n^2)$.*

**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 8.** *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 $B \subseteq Y$.*

The proof of Lemma 8 is deferred to the full version of this work.

Running the algorithm of Lemma 8 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 9.** Matching Cut *can be solved in time $2^{\mathrm{d\overline{c}}(G)}O(nm)$.*

## 4 An Improved Exact Exponential 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, i.e., 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 instance/subproblem while a branching rule results in at least two instances/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 (R1) – (R4) mentioned in Section 3 to the current instance (in the order of the rules). In addition, we need two new reduction rules.

**(R5)** If there are vertices $u, v \in F$ such that $N(u) = N(v) = \{x, y\}$ with $x \in A, y \in B$, then $A := A \cup \{u\}$, $B := B \cup \{v\}$.

▶ **Proposition 4.** *Reduction Rule (R5) is safe: $G$ has a matching cut separating $A$ and $B$ if and only if $G$ has a matching cut separating $A \cup \{u\}$ and $B \cup \{v\}$.*

The proof of Proposition 4 is deferred to the full version of this work.

**(R6)** If there are vertices $u, v \in F$ such that $N(u) = N(v) = \{x, y\}$ with $x \in A, y \in F$ then $A := A \cup \{u\}$.

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

▶ **Proposition 5.** *Reduction Rule (R6) is safe:*

- **(i)** *Let $x \in A$ and $y \in F$. Then $G$ has a matching cut separating $A$ and $B$ if and only if $G$ has a matching cut separating $A \cup \{u\}$ and $B$.*
- **(ii)** *Let $x \in F$ and $y \in B$. Then $G$ has a matching cut separating $A$ and $B$ if and only if $G$ has a matching cut separating $A$ and $B \cup \{v\}$.*

The proof of Proposition 5 is deferred to the full version of this work.

Our algorithm consists of six branching rules dealing with small configurations (connected subgraphs with at most eight vertices some of them may already belong to $A$ or $B$.) See Fig. 2. The Branching Rules (B1) and (B2) are new, the last four have been used in [14]. Moreover, compared to [14] 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, i.e., $|V(G)| - |A| - |B|$.

**(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 $(3, 2)$.

**(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 $(3, 2)$.

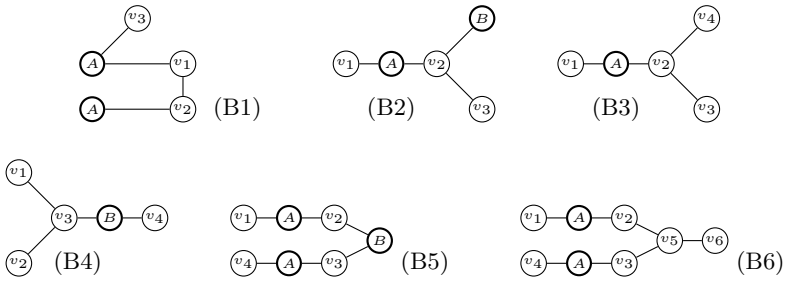


**Figure 2** Branching configurations: Vertices with label $A$ and $B$ belong to $A$, respectively, to $B$; vertices with labels $v_i$ are in $F$.

**(B3)** 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 $(4, 1)$.

**(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 $(4, 1)$.

**(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 $(3, 2)$.

**(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 (B3, B4), 1.3734 (B6) and 1.3248 (B1, B2, B5). Consequently, the running time of our algorithm is $O^*(1.3803^n)$.

It remains to show that if none of the reduction and branching rules is applicable to $(G, A, B)$ then $G$ has a matching cut $(X, Y)$ with $A \subseteq X$ and $B \subseteq Y$. The proof of this fact is deferred to the full version of this work. In summary, we obtain the following result.

▶ **Theorem 10.** Matching Cut *can be solved in time* $O^*(1.3803^n)$.

## 5 Conclusions

We provided three algorithms for Matching Cut: an exact exponential algorithm of running time $O^*(1.3803^n)$, 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 algorithm previously given in [14], where $\tau(G) \geq \max\{\mathrm{dc}(G), \mathrm{d\bar{c}}(G)\}$ is the vertex cover number. 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 a linear vertex-kernel for the parameter vertex cover number $\tau(G)$ is open. Moreover, it is open whether the problem admits a polynomial kernel for the feedback vertex set number of $G$. Finally, it is natural to ask whether the running time of our $O^*(1.3803^n)$ branching algorithm can be improved.

## References


1. Júlio Araújo, Nathann Cohen, Frédéric Giroire, and Frédéric Havet. Good edge-labelling of graphs. *Discr. Appl. Math.*, 160(18):2502–2513, 2012.
2. N. R. Aravind, Subrahmanyam Kalyanasundaram, and Anjeneya Swami Kare. On Structural Parameterizations of the Matching Cut Problem. In *Proceedings of the 11th International Conference on Combinatorial Optimization and Applications (COCOA '17)*, volume 10628 of *LNCS*, pages 475–482. Springer, 2017.
3. Bengt Aspvall, Michael F. Plass, and Robert Endre Tarjan. A Linear-Time Algorithm for Testing the Truth of Certain Quantified Boolean Formulas. *Inf. Process. Lett.*, 8(3):121–123, 1979.
4. Paul S. Bonsma. The complexity of the matching-cut problem for planar graphs and other graph classes. *J. Graph Theory*, 62(2):109–126, 2009.
5. Anudhyan Boral, Marek Cygan, Tomasz Kociumaka, and Marcin Pilipczuk. A Fast Branching Algorithm for Cluster Vertex Deletion. *Theory Comput. Syst.*, 58(2):357–376, 2016.
6. Mieczyslaw Borowiecki and Katarzyna Jesse-Józefczyk. Matching cutsets in graphs of diameter 2. *Theor. Comput. Sci.*, 407(1-3):574–582, 2008.
7. Vasek Chvátal. Recognizing decomposable graphs. *J. Graph Theory*, 8(1):51–53, 1984.
8. Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.
9. Martin Davis and Hilary Putnam. A Computing Procedure for Quantification Theory. *J. ACM*, 7(3):201–215, 1960.
10. Shimon Even, Alon Itai, and Adi Shamir. On the Complexity of Timetable and Multicommodity Flow Problems. *SIAM J. Comput.*, 5(4):691–703, 1976.
11. Arthur M. Farley and Andrzej Proskurowski. Networks immune to isolated line failures. *Networks*, 12(4):393–403, 1982.
12. Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Springer, 2010.
13. Ron L. Graham. On primitive graphs and optimal vertex assignments. *Ann. N. Y. Acad. Sci.*, 175(1):170–186, 1970.
14. Dieter Kratsch and Van Bang Le. Algorithms solving the Matching Cut problem. *Theor. Comput. Sci.*, 609:328–335, 2016.
15. Hoàng-Oanh Le and Van Bang Le. On the Complexity of Matching Cut in Graphs of Fixed Diameter. In *Proceedings of the 27th International Symposium on Algorithms and Computation (ISAAC '16)*, volume 64 of *LIPIcs*, pages 50:1–50:12. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.
16. Van Bang Le and Bert Randerath. On stable cutsets in line graphs. *Theor. Comput. Sci.*, 301(1-3):463–475, 2003.
17. Augustine M. Moshi. Matching cutsets in graphs. *J. Graph Theory*, 13(5):527–536, 1989.
18. Maurizio Patrignani and Maurizio Pizzonia. The Complexity of the Matching-Cut Problem. In *Proceedings of the 27th International Workshop on Graph-Theoretic Concepts in Computer Science (WG '01)*, volume 2204 of *LNCS*, pages 284–295. Springer, 2001.