The Perfect Matching Cut Problem Revisited

Van Bang Le[1] and Jan Arne Telle[2]

[1] Institut für Informatik, Universität Rostock, Rostock, Germany
van-bang.le@uni-rostock.de
[2] Department of Informatics, University of Bergen, N-5020 Bergen, Norway
Jan.Arne.Telle@uib.no

Abstract. In a graph, a perfect matching cut is an edge cut that is a perfect matching. PERFECT MATCHING CUT is the problem of deciding whether a given graph has a perfect matching cut, and is known to be NP-complete. We revisit the problem and show that PERFECT MATCHING CUT remains NP-complete when restricted to bipartite graphs of maximum degree 3 and arbitrarily large girth. Complementing this hardness result, we give two graph classes in which PERFECT MATCHING CUT is polynomial time solvable. The first one includes claw-free graphs and graphs without an induced path on five vertices, the second one properly contains all chordal graphs. Assuming the Exponential Time Hypothesis, we show there is no $O^*(2^{o(n)})$-time algorithm for PERFECT MATCHING CUT even when restricted to n-vertex bipartite graphs, and also show that PERFECT MATCHING CUT can be solved in $O^*(1.2721^n)$ time by means of an exact branching algorithm.

Keywords: Matching cut · Perfect matching cut · Computational complexity · Exact branching algorithm · Graph algorithm.

1 Introduction

In a graph $G = (V, E)$, a *cut* is a partition $V = X \cup Y$ of the vertex set into disjoint, non-empty sets X and Y. The set of all edges in G having an endvertex in X and the other endvertex in Y, written $E(X, Y)$, is called the *edge cut* of the cut (X, Y). A *matching cut* is an edge cut that is a (possibly empty) matching. Another way to define matching cuts is as follows; see [8,12]: a cut (X, Y) is a matching cut if and only if each vertex in X has at most one neighbor in Y and each vertex in Y has at most one neighbor in X. MATCHING CUT is the problem of deciding if a given graph admits a matching cut and this problem has received much attention lately; see [7,10] for recent results.

An interesting special case, where the edge cut $E(X, Y)$ is a *perfect matching*, was considered in [13]. The authors proved that PERFECT MATCHING CUT, the problem of deciding if a given graph admits an edge cut that is a perfect matching, is NP-complete. A perfect matching cut (X, Y) can be described as a (σ, ρ) 2-partitioning problem [21], as every vertex in X must have exactly one neighbor in Y and every vertex in Y must have exactly one neighbor in X. By results of [6,21,22] it can therefore be solved in FPT time when parameterized by treewidth or cliquewidth (to mention only the two most famous width parameters) and in XP time when parameterized by mim-width (maximum induced matching-width) of a given decomposition of the graph. For several classes of graphs, like interval and permutation, a decomposition of bounded mim-width can be computed in polynomial-time [3], thus the problem is polynomial on such classes.

In this paper, we revisit the PERFECT MATCHING CUT problem. Our results are:

- While MATCHING CUT is polynomial time solvable when restricted to graphs of maximum degree 3 and its computational complexity is still open for graphs with large girth, we prove that PERFECT MATCHING CUT is NP-complete in the class of bipartite graphs having maximum degree 3 and arbitrary large girth. Further, we show that PERFECT MATCHING CUT cannot be solved in $O^*(2^{o(n)})$ time for n-vertex bipartite graphs and cannot be solved in $O^*(2^{o(\sqrt{n})})$ time for bipartite graphs having maximum degree 3 and arbitrary girth.
- We provide the first exact algorithm to solve PERFECT MATCHING CUT on n-vertex graphs, of runtime $O^*(1.2721^n)$. Note that the fastest algorithm for MATCHING CUT has runtime $O^*(1.3280^n)$ and is based on the current-fastest algorithm for 3-SAT [17].
- We give two graph classes of unbounded mim-width in which PERFECT MATCHING CUT is solvable in polynomial time. The first class contains all claw-free graphs and graphs without an induced path on 5 vertices, the second class contains all chordal graphs.

Related work. The computational complexity of MATCHING CUT was first considered by Chvátal in [8], who proved that MATCHING CUT is NP-complete for graphs with maximum degree 4 and polynomial time solvable for graphs with maximum degree at most 3. Hardness results were obtained for further restricted graph classes such as bipartite graphs, planar graphs and graphs of bounded diameter (see [4,19,20]). Further graph classes in which MATCHING CUT is polynomial time solvable were identified, such as graphs of bounded tree-width, claw-free, hole-free and Ore-graphs (see [4,7,20]). FPT algorithms and kernelization for MATCHING CUT with respect to various parameters has been discussed in [1,2,10,11,17,18]. The current-best exact algorithm solving MATCHING CUT has a running time of $O^*(1.3280^n)$ where n is the vertex number of the input graph [17]. Faster exact algorithms can be obtained for the case when the minimum degree is large [7]. The recent paper [10] addresses enumeration aspects of matching cuts.

Very recently, a related notion has been discussed in [5]. In this paper, the authors consider perfect matchings $M \subseteq E$ of a graph $G = (V, E)$ such that $G \setminus M = (V, E \setminus M)$ is disconnected, which they call perfect matching-cuts. To avoid confusion, we call such a perfect matching a *disconnected perfect matching*. Note that, by definition, every perfect matching cut is a disconnected perfect matching but a disconnected perfect matching need not be a perfect matching cut. Indeed, all perfect matchings of the cycle on $4k + 2$ vertices are disconnected perfect matchings and none of them is a perfect matching cut. In [5], the authors showed, among others, that recognizing graphs having a disconnected perfect matching is NP-complete even when restricted to graphs with maximum 4, and left open the case of maximum degree 3. It is not clear whether our hardness result on degree-3 graphs can be modified to obtain a hardness result of recognizing degree-3 graphs having a disconnected perfect matching.

Notation and terminology. Let $G = (V, E)$ be a graph with vertex set $V(G) = V$ and edge set $E(G) = E$. 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 $N(v)$. Let $\deg(v) := |N(v)|$ be the degree of the vertex v, and $N[v] := N(v) \cup \{v\}$ be the closed neighborhood of v. For a subset $F \subseteq V$, $G[F]$ is the subgraph of G induced by F, and

$G - F$ stands for $G[V \setminus F]$. We write $N_F(v)$ and $N_F[v]$ for $N(v) \cap F$ and $N[v] \cap F$, respectively, and call the vertices in $N(v) \cap F$ the *F-neighbors* of v. The *girth* of G is the length of a shortest cycle in G, assuming G contains a cycle. The path on n vertices is denoted by P_n, the complete bipartite graph with one color class of size p and the other of size q is denoted by $K_{p,q}$; $K_{1,3}$ is also called a *claw*.

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 factor*. The running time of a branching algorithm is $O^*(\alpha^n)$, where $\alpha = \max_i \alpha_i$ and α_i is the branching factor of branching rule i, and the maximum is taken over all branching rules. Throughout the paper we use the O^* notation which suppresses polynomial factors. We refer to [9] for more details on exact branching algorithms.

Algorithmic lower bounds in this paper are conditional, based on the Exponential Time Hypothesis (ETH) [14]. The ETH states that there is no $O^*(2^{o(n)})$-time algorithm for 3-SAT where n is the variable number of the input 3-CNF formula. It is known that the hard case for 3-SAT already consists of formulas with $O(n)$ clauses [15]. Thus, assuming ETH, there is no $O^*(2^{o(m)})$-time algorithm for 3-SAT where m is the clause number of the input formula.

Observe that a graph has a perfect matching cut if and only if each of its connected components has a perfect matching cut. Thus, we may assume that all graphs in this paper are connected.

2 Hardness results

In this section, we give two polynomial time reductions from POSITIVE NAE 3-SAT to PERFECT MATCHING CUT. Recall that an instance for POSITIVE NAE 3-SAT is a 3-CNF formula $F = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ over n variables $x_1, x_2, \ldots, x_n$, in which each clause C_j consists of three distinct variables. The problem asks whether there is a truth assignment of the variables such that every clause in F has one true and one false variable. Such an assignment is called *nae assignment*.

It is well-known that there is a polynomial reduction from 3-SAT to POSITIVE NAE 3-SAT where the variable number of the reduced formula is linear in the clause number of the original formula. Hence, the ETH implies that there is no subexponential time algorithm for POSITIVE NAE 3-SAT in the number of variables.

Theorem 1. *Assuming ETH,* PERFECT MATCHING CUT *cannot be solved in subexponential time in the vertex number, even when restricted to bipartite graphs.*

Proof. We give a polynomial reduction from POSITIVE NAE 3-SAT to PERFECT MATCHING CUT restricted to bipartite graphs.

Given a 3-CNF formula F, construct a graph G as follows. For each clause $C_j = \{c_{j1}, c_{j2}, c_{j3}\}$, let $G(C_j)$ be the cube with *clause vertices* labeled c_{j1}, c_{j2}, c_{j3}, respectively, as depicted in Fig. 1. For each variable x_i, we introduce a *variable vertex* x_i and a dummy vertex x'_i adjacent only to x_i. Finally, we connect a variable vertex x_i to a clause vertex in $G(C_j)$ if and only if C_j contains the variable x_i, i.e., $x_i = c_{jk}$ for some $k \in \{1, 2, 3\}$.

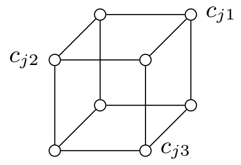


Fig. 1. The graph $G(C_j)$.

Observe that G is bipartite and has the following property: no perfect matching M of G (in particular, no perfect matching cut) contains an edge between a clause vertex and a variable vertex. Thus, for every perfect matching cut $M = E(X, Y)$ of G, the restriction $M_j = E(X_j, Y_j)$ on $G(C_j)$ is a perfect matching cut of $G(C_j)$. Moreover, $G(C_j)$ has the following property: it has exactly three perfect matching cuts, and in any perfect matching cut of $G(C_j)$ not all clause vertices belong to the same part. Conversely, any bipartition of C_j can be extended (in a unique way) to a perfect matching cut M_j of $G(C_j)$. See also Fig. 2.

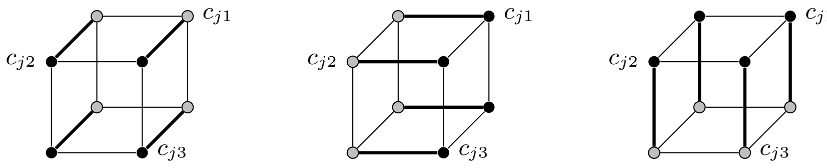


Fig. 2. The three perfect matching cuts of $G(C_j)$; black vertices in X, gray vertices in Y.

We are now ready to see that F has a nae assignment if and only if G has a perfect matching cut: First, if there is a nae assignment for F then put all true variable vertices into X, all false variable vertices into Y, and extend X and Y (in a unique way) to a perfect matching cut of G; note that x'_i and x_i have to belong to different parts. Second, if (X, Y) is a perfect matching cut of G then defining x_i be `true` if $x_i \in X$ and `false` if $x_i \in Y$ we obtain a nae assignment for F.

Observe that G has $N = O(n + m)$ vertices. Hence the reduction implies that, assuming ETH, PERFECT MATCHING CUT has no subexponential time algorithm in vertex number N, even when restricted to bipartite graphs. □

We now describe how to avoid vertices of degree 4 and larger (the clause and variable vertices) in the previous reduction to obtain a bipartite graph with maximum degree 3 and large girth.

Theorem 2. *Let $g > 0$ be a given integer.* PERFECT MATCHING CUT *remains* NP-*complete when restricted to bipartite graphs of maximum degree three and girth at least g.*

Proof. We modify the gadgets used in the proof of Theorem 1. Let $h \geq 0$ be a fixed integer, which will be more concrete later.

Clause gadget: we subdivide every edge of the cube with $4h+4$ new vertices, fix a vertex c_j of degree 3 and label the three neighbors of c_j with c_{j1}, c_{j2} and c_{j3}, respectively. We denote the obtained graph again by $G(C_j)$ and call the labeled vertices the *clause vertices*. The case $h = 0$ is shown in Fig. 3. Observe, $G(C_j)$ has the same properties of the cube used in the previous reduction: it has exactly three perfect matching cuts, and in any perfect matching cut of $G(C_j)$ not all clause vertices belong to the same part. Moreover, any bipartition of C_j can be extended (in a unique way) to a perfect matching cut M_j of $G(C_j)$. See also Fig. 4.

Variable gadget: for each variable x_i we introduce m *variable vertices* x_i^j one for each clause C_j, $1 \leq j \leq m$, as follows. (We assume that the formula F consists of $m \geq 3$ clauses.) First, take a cycle with m vertices $x_i^1, x_i^2, \ldots, x_i^m$ and edges $x_i^1 x_i^2$, $x_i^2 x_i^3, \ldots,$ $x_i^{m-1} x_i^m$ and $x_i^1 x_i^m$. Then subdivide every edge with $4h+3$ new vertices to obtain the graph $G(x_i)$. Thus, $G(x_i)$ is a cycle on $4m(h+1)$ vertices. The case $m = 3, h = 0$ is shown in Fig. 3. The following property of $G(x_i)$ can be verified immediately: in any perfect matching cut of $G(x_i)$, all variable vertices x_i^j, $1 \leq j \leq m$, belong to the same part.

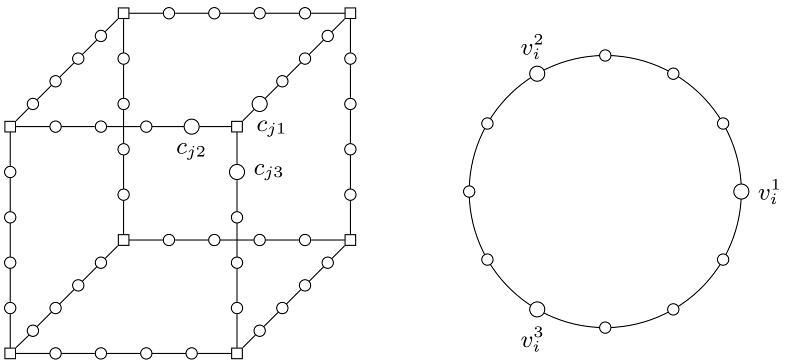


Fig. 3. The clause gadget $G(C_j)$ (left) and the variable gadget $G(x_i)$ (right) in case $m = 3$ and $h = 0$.

Finally, the graph G is obtained by connecting the variable vertex x_i^j in $G(x_i)$ to a clause vertex in $G(C_j)$ by an edge whenever x_i appears in clause C_j, i.e., $x_i = c_{jk}$ for some $k \in \{1, 2, 3\}$.

It follows from construction, that

- G has maximum degree 3;
- G is bipartite. This can be seen as follows. The bipartite subgraph formed by all $G(C_j)$ has a bipartition into independent sets A and B such that all clause vertices c_{jk} are in A. The bipartite subgraph formed by all $G(x_i)$ has a bipartition into independent

sets C and D such that all variable vertices x_i^j are in C. Since the edges in G between these two subgraphs connect clause vertices and variable vertices, therefore the vertex set of G can be partitioned into independent sets $A \cup D$ and $B \cup C$;

– G has girth at least $\min\{4m(h+1), 8(h+2)\}$. This can be seen as follows. There are 3 types of cycles in G. Any of the cycles $G(x_i)$ has length $4m(h+1)$. A shortest cycle in any $G(C_j)$ is a subdivision of a 4-cycle and has length $4(4h+5)$. The cycles of the last type go through some $G(x_i)$s and some $G(C_j)$s; the length of a shortest one among them is at least $4+(4h+4)+4+(4h+4) = 8(h+2)$.

Moreover, as in the previous construction, G has the following property: no perfect matching M of G (in particular, no perfect matching cut) contains an edge between a clause vertex and a variable vertex. Thus, for every perfect matching cut $M = E(X,Y)$ of G, the restrictions of M on $G(C_j)$ and on $G(x_i)$ are perfect matching cuts of $G(C_j)$ and of $G(x_i)$, respectively.

Now, as in the proof of Theorem 1, we can argue that F has a nae assignment if and only if G has a perfect matching cut. First, if there is a nae assignment for F then put all true variable vertices and clause vertices into X, all false variable vertices and clause vertices into Y, and extend X and Y (in a unique way) to a perfect matching cut of G. See Fig.4 for an extension in $G(C_j)$. Second, if (X,Y) is a perfect matching cut of G then defining x_i be `true` if $x_i \in X$ and `false` if $x_i \in Y$ we obtain a nae assignment for F.

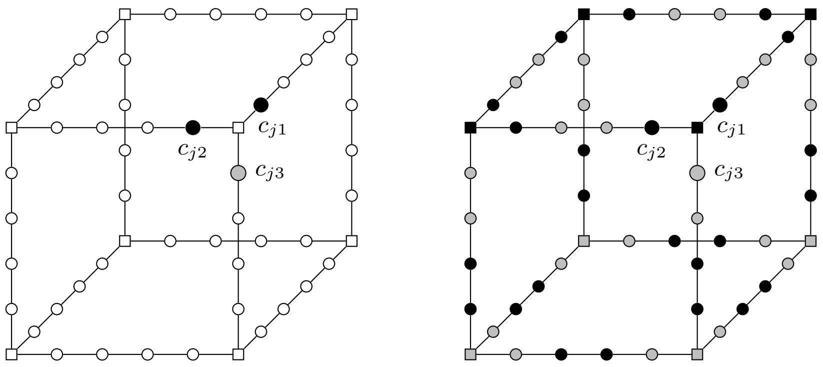


Fig. 4. How to extend X (black) and Y (gray) on the left-hand site to a perfect matching cut in $G(C_j)$ on the right-hand side.

Finally, given $g > 0$, let $h \geq 0$ be an integer at least $\max\{\frac{g}{4m} - 1, \frac{g}{8} - 2\}$. Then G has girth at least $\min\{4m(h+1), 8(h+2)\} \geq g$. This completes the proof. $\square$

Note that the graph G in the proof of Theorem 2 has $N = O(m + nm)$ vertices, where n and m are the variable number and clause number, respectively, of the formula F. Since we may assume that F has $m = O(n)$ clauses, G has $N = O(n^2)$ vertices. Hence we obtain the following.

Theorem 3. *Assuming ETH, there is no $O^*(2^{o(\sqrt{n})})$-time algorithm for* PERFECT MATCHING CUT *even when restricted to n-vertex bipartite graphs with maximum degree 3 and arbitrary large girth.*

Observe that PERFECT MATCHING CUT is trivial for graphs with maximum degree 2: a (connected) graph with maximum degree 2 has a perfect matching cut if and only if it is a path with even number of vertices or a cycle with $4k$ vertices. Thus, the maximum degree constraint in Theorems 2 and 3 is optimal.

3 An exact exponential algorithm

Recall that, assuming ETH, there is no $O^*(2^{o(n)})$-time algorithm for PERFECT MATCHING CUT on n-vertex (bipartite) graphs. The main result in this section is an algorithm solving PERFECT MATCHING CUT in $O^*(1.2721^n)$ time.

Recall that all graphs considered are connected. Our algorithm follows the idea of known branching algorithms for MATCHING CUT [7,17,18]. We adapt basic reduction rules for matching cuts to perfect matching cuts, and add new reduction and branching rules for perfect matching cuts.

If the input graph $G = (V, E)$ has a perfect matching cut (X, Y), then some edge has an endvertex a in X and the other endvertex b in Y. The branching algorithm will be executed for all possible edges $ab \in E$, hence $O(m)$ 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 B will be extended or it will be determined that there is no perfect matching cut *separating* A and B, that is a perfect matching cut (X, Y) with $A \subseteq X$ and $B \subseteq Y$. 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 subproblem while a branching rule results in at least two subproblems, with different extensions of A and B. Note that G has a perfect 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 perfect 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 $F = V \setminus (A \cup B)$.

Reduction Rules 1 (except the last three items), 2 (except the last two items), 3 and 4 below are given in [18] for matching cuts. As perfect matching cuts are matching cuts, they remain correct for perfect matching cuts.

Reduction Rule 1

- *If a vertex in A has two B-neighbors, or a vertex in B has two A-neighbors then STOP: "G has no matching cut separating A, B".*
- *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 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 a vertex in A and a vertex in B have three or more common neighbors in F then STOP: "G has no matching cut separating A, B".*

- *If a vertex in A (respectively in B) has no neighbor in $B \cup F$ (respectively in $A \cup F$) then STOP: "G has no* perfect *matching cut separating A, B".*
- *If there are $x \in A$ and $y \in B$ such that $N(x) \cap F = N(y) \cap F = \{v\}$ then STOP: "G has no* perfect *matching cut separating A, B".*

Reduction Rule 2

- *If $v \in F$ and $|N(v) \cap A| \geq 2$ (respectively $|N(v) \cap B| \geq 2$) then $A := A \cup \{v\}$ (respectively $B := B \cup \{v\}$).*
- *If $v \in F$ and $|N(v) \cap N(x) \cap F| \geq 3$ for some $x \in A$ (respectively $y \in B$) then $A := A \cup \{v\} \cup (N(v) \cap N(x) \cap F)$ (respectively $B := B \cup \{v\} \cup (N(v) \cap N(y) \cap F)$).*

Reduction Rule 3 *If $x \in A$ (respectively $y \in B$) has two adjacent F-neighbors u, v then $A := A \cup \{u, v\}$ (respectively $B := B \cup \{u, v\}$).*

Reduction Rule 4 *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 following facts hold:

- The edge cut $E(A, B)$ is a (not necessary perfect) matching cut of $G[A \cup B] = G - F$ due to Reduction Rule 1. Moreover, any vertex in A and any vertex in B have at most two common neighbors in F.
- Every vertex in F is adjacent to at most one vertex in A and at most one vertex in B due to Reduction Rule 2.
- The neighbors in F of any vertex in A and the neighbors in F of any vertex in B form an independent set due to Reduction Rule 3, and
- Every vertex in A adjacent to a vertex in B has no neighbor in F and every vertex in B adjacent to a vertex in A has no neighbor in F due to Reduction Rule 4.

Reduction Rule 5 below is given in [17] and remains correct for perfect matching cuts.

Reduction Rule 5 *If there are vertices $u, v \in F$ such that $N(u) = N(v) = \{x, y\}$ with $x \in A, y \in B$, then $A := A \cup \{u\}$, $B := B \cup \{v\}$.*

The remaining reduction rules work for perfect matching cuts but not for matching cuts in general.

Reduction Rule 6 *If $x \in A$ (respectively $y \in B$) has exactly one neighbor $v \in F$ then $B := B \cup \{v\}$ (respectively $A := A \cup \{v\}$).*

Proof (of safeness). Let $x \in A$ with $N(x) \cap F = \{v\}$. By Reduction Rule 4, $N(x) \cap B = \emptyset$. If (X, Y) is a perfect matching separating A and B, then $N(x) \setminus \{v\} \subseteq X$, hence the neighbor v of x must belong to Y. The case $y \in B$ is symmetric. □

Reduction Rule 7 *Let $z \in A$ (respectively $z \in B$) and let $v \in N(z) \cap F$.*

- *If $\deg(v) = 1$ then $B := B \cup \{v\}$ (respectively $A := A \cup \{v\}$).*
- *If $\deg(v) = 2$ and $w \in F$ is other neighbor of v then $B := B \cup \{w\}$ (respectively $A := A \cup \{w\}$).*

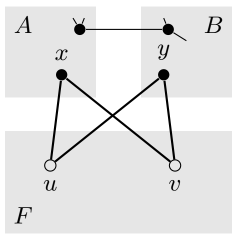
Proof (of safeness). Let $z \in A$ and $v \in N(z) \cap F$. Let (X, Y) be a perfect matching of G separating A and B. If z is the only neighbor of v, then, as $z \in X$, v must belong to Y. If $N(v) = \{z, w\}$ with $w \in F$, then w must belong to Y, otherwise both neighbors of v were in X. The case $z \in B$ is symmetric. □

Reduction Rule 8 *Let $x \in A$ and $y \in B$ with $|N(x) \cap N(y) \cap F| = 2$. If $|N(x) \cap F| \geq 3$ or $|N(y) \cap F| \geq 3$ then $A := A \cup N(x) \setminus N(y)$, $B := B \cup N(y) \setminus N(x)$.*

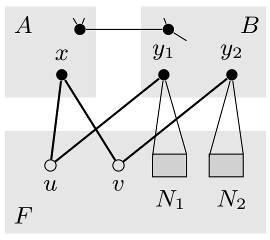
Proof (of safeness). Assume that (X, Y) is a perfect matching cut of G separating A and B. Then $N(x) \cap N(y) \cap F$ must contain one vertex in X and one vertex in Y. Hence $N(x) \setminus N(y) \subseteq X$ and $N(y) \setminus N(x) \subseteq Y$. □

We now describe the branching rules. All branching rules are based on the fact that, in any perfect matching cut (X, Y) separating A and B, every vertex in X has exactly one neighbor in Y and every vertex in Y has exactly one neighbor in X. Thus, if some vertex in A has no neighbor in B, it must have a neighbor in F that must go to Y, and if some vertex in B has no neighbor in A, it must have a neighbor in F that must go to X. Note that by Reduction Rule 6, every vertex in $A \cup B$ has none or at least two neighbors in F. By Reduction Rule 1, any $x \in A$ and $y \in B$ have at most two common neighbors in F.

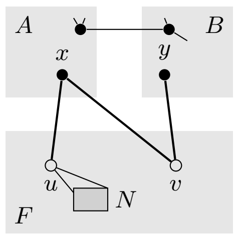
To determine the branching vectors which correspond to our branching rules, we set the size of an instance (G, A, B) as its number of free vertices, i.e., $|V(G)| - |A| - |B|$. There are seven branching rules. Vertices in $A \cup B$ having exactly two neighbors in F will be covered by the first four branching rules.



Branching Rule 1



Branching Rule 2



Branching Rule 3

Fig. 5. When Branching Rules 1, 2 and 3 are applicable.

Branching Rule 1 *Let $x \in A$ and $y \in B$ with $N(x) \cap N(y) \cap F = \{u, v\}$. By Reduction Rule 8, $N(x) \cap F = N(y) \cap F = \{u, v\}$. We branch into two subproblems.*

- *First, add $N[u] \cap F$ to A. Then $N[v] \cap F$ has to be added to B.*
- *Second, add $N[u] \cap F$ to B. Then $N[v] \cap F$ has to be added to A.*

The branching vector of Branching Rule 1 is

$$\big(|(N[u] \cup N[v]) \cap F|, |(N[u] \cup N[v]) \cap F|\big).$$

By Reduction Rule 5, $|(N[u] \cup N[v]) \cap F| \geq 3$, hence the branching factor of Branching Rule 1 is at most $\tau(3, 3) < 1.2560$.

Branching Rule 2 *Let $x \in A$ with $N(x) \cap F = \{u, v\}$ and $N(u) \cap B = \{y_1\}$, $N(v) \cap B = \{y_2\}$. We branch into 2 subproblems.*

- *First, add u to B. Then v has to be added to A and $N_2 := N(y_2) \cap F \setminus \{v\}$ has to be added to B.*

– *Second, add v to B. Then u has to be added to A and $N_1 := N(y_1) \cap F \setminus \{u\}$ has to be added to B.*

Symmetrically for $y \in B$ with $N(y) \cap F = \{u, v\}$ and $N(u) \cap A = \{x_1\}$, $N(v) \cap A = \{x_2\}$.

By Branching Rule 1, $v \notin N_1$, $u \notin N_2$. Hence, the branching vector of Branching Rule 2 is

$$\big(2 + |N_1|, 2 + |N_2|\big).$$

By Reduction Rule 6, $|N_1| \geq 1, |N_2| \geq 1$. Hence the branching factor is at most $\tau(3,3) = \sqrt[3]{2} < 1.2600$.

Branching Rule 3 *Let $x \in A$ with $N(x) \cap F = \{u, v\}$ and $N(u) \cap B = \emptyset$, $N(v) \cap B = \{y\}$. We branch into two subproblems.*

– *First, add u to B. Then v has to be added to A and $N := N(u) \cap F$ has to be added to B.*

– *Second, add v to B. Then u has to be added to A.*

Symmetrically for $y \in B$ with $N(y) \cap F = \{u, v\}$ and $N(u) \cap A = \emptyset$, and $N(v) \cap A = \{x\}$.

The branching vector of Branching Rule 3 is

$$\big(2 + |N|, 2\big).$$

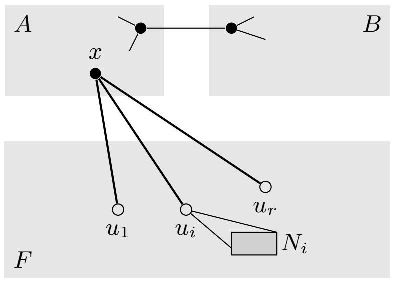
By Reduction Rule 7, $|N| \geq 2$, hence the branching factor of Branching Rule 3 is at most $\tau(4,2) < 1.2721$.

Branching Rule 4 *Let $x \in A$ with $N(x) \cap F = \{u_1, u_2, \ldots, u_r\}$, $r \geq 2$, and $N(u_i) \cap B = \emptyset$, $1 \leq i \leq r$. We branch into r subproblems. For each $1 \leq i \leq r$, the instance of the i-th subproblem is obtained by adding u_i to B. Then $N(x) \cap F \setminus \{u_i\}$ has to be added to A and $N_i := N(u_i) \cap F$ has to be added to B.*
Symmetrically for $y \in B$ with $N(y) \cap F = \{v_1, v_2, \ldots, v_r\}$ and v_i has no neighbor in A, $1 \leq i \leq r$.

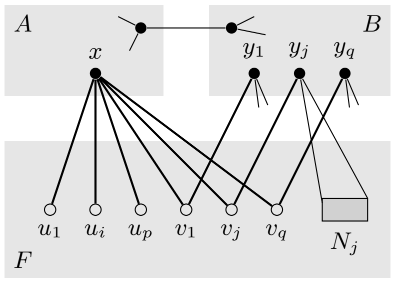
The branching vector of Branching Rule 4 is

$$\big(r + |N_1|, r + |N_2|, \ldots, r + |N_r|\big).$$

By Reduction Rule 7, $|N_i| \geq 2$, hence the branching factor of Branching Rule 4 is at most $\tau(r+2, r+2, \ldots, r+2) = \sqrt[r+2]{r} < 1.2600$.



Branching Rule 4



Branching Rule 5

Fig. 6. When Branching Rules 4 and 5 are applicable.

Branching Rules 1 and 4 together with the remaining branching rules cover vertices in $A \cup B$ having at least three neighbors in F. Branching Rule 5 deals with the case $z \in A$ (respectively $z \in B$) in which at least two vertices in $N(z) \cap F$ have neighbors in B (respectively in A).

Branching Rule 5 *Let $x \in A$ with $N(x) \cap F = \{u_1, \ldots, u_p, v_1, v_2, \ldots, v_q\}$, $p \geq 0$, $q \geq 2$, such that $N(u_i) \cap B = \emptyset$, $1 \leq i \leq p$ and $N(v_j) \cap B = \{y_j\}$, $1 \leq j \leq q$. We branch into $r = p + q$ subproblems.*

- *For each $1 \leq i \leq p$, the instance of the i-th subproblem is obtained by adding u_i to B. Then $N(x) \cap F \setminus \{u_i\}$ has to be added to A and all $N_j := N(y_j) \cap F \setminus \{v_j\}$, $1 \leq j \leq q$, have to be added to B.*
- *For each $1 \leq j \leq q$, the instance of the $p+j$-th subproblem is obtained by adding v_j to B. Then $N(x) \cap F \setminus \{v_j\}$ has to be added to A and all $N_k := N(y_j) \cap F \setminus \{v_j\}$, $1 \leq k \leq q$, $k \neq j$, have to be added to B.*

Symmetrically for $y \in B$ with $N(y) \cap F = \{u_1, \ldots, u_p, v_1, v_2, \ldots, v_q\}$, $p \geq 0$, $q \geq 2$ such that $N(u_i) \cap A = \emptyset$, $1 \leq i \leq p$ and $N(v_j) \cap A = \{x_j\}$, $1 \leq j \leq q$.

By Branching Rule 1 and Reduction Rule 2, N_j are pairwise disjoint and $N_j \cap \{v_1, \ldots, v_q\} = \emptyset$. Hence, the branching vector of Branching Rule 5 is

$$\big(r + \sum_j |N_j|, \ldots, r + \sum_j |N_j|, r + \sum_{k \neq 1} |N_k|, \ldots, r + \sum_{k \neq q} |N_k|\big).$$

Due to Branching Rules 1–4, each y_j has at least three neighbors in F. Hence $|N_j| \geq 2$, $1 \leq j \leq q$. Thus, the branching factor is at most $\tau(r+2q, \ldots, r+2q, r+2(q-1), \ldots, r+2(q-1)) \leq \tau(r+2, \ldots, r+2) = \sqrt[r+2]{r} < 1.2600$.

The last two branching rules deal with the case $z \in A$ (respectively $z \in B$) in which exactly one vertex in $N(z) \cap F$ has a unique neighbor in B (respectively in A).

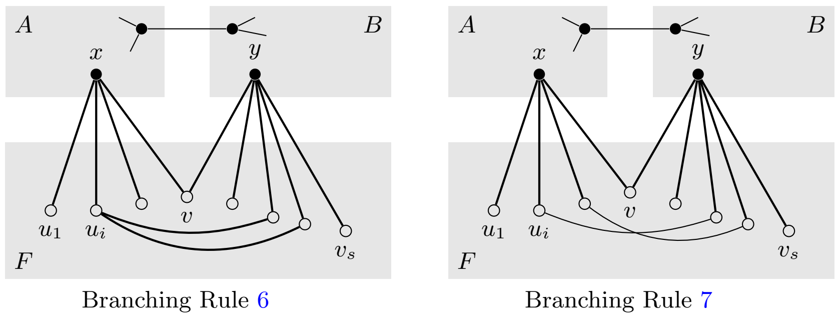


Fig. 7. When Branching Rules 6 and 7 are applicable.

Branching Rule 6 *Let $x \in A$ with $N(x) \cap F = \{u_1, u_2, \ldots, u_r, v\}$, $r \geq 2$, such that $N(u_i) \cap B = \emptyset$, $1 \leq i \leq r$, and $N(v) \cap B = \{y\}$. Write $N(y) \cap F \setminus \{v\} = \{v_1, \ldots, v_s\}$, $s \geq 2$. Assume that some u_i has two neighbors in $\{v_1, \ldots, v_s\}$. We branch into 2 subproblems.*

– *First, add v to A. Then $\{v_1, \ldots, v_s\}$ and u_i have to be added to B, and $\{u_1, \ldots, u_r\} \setminus \{u_i\}$ has to be added to A.*
– *Second, add v to B. Then $\{u_1, \ldots, u_r\}$ has to be added to A.*

Symmetrically for $y \in B$ with $N(y) \cap F = \{u_1, u_2, \ldots, u_r, v\}$ such that $N(u_i) \cap A = \emptyset$, $1 \le i \le r$, and $N(v) \cap A = \{x\}$ and some u_i has two neighbors in $N(x) \cap F \setminus \{v\}$.

The branching vector of Branching Rule 6 is

$$(r + s + 1, r + 1).$$

Since $r \ge 2$ and $s \ge 2$, we have $\tau(r + s + 1, r) \le \tau(5, 3) < 1.1939$.

Branching Rule 7 *Let $x \in A$ with $N(x) \cap F = \{u_1, u_2, \ldots, u_r, v\}$, $r \ge 2$, such that $N(u_i) \cap B = \emptyset$, $1 \le i \le r$, and $N(v) \cap B = \{y\}$. Write $N(y) \cap F \setminus \{v\} = \{v_1, \ldots, v_s\}$, $s \ge 2$. We branch into $r + s$ subproblems.*

– *For each $1 \le i \le r$, the instance of the i-th subproblem is obtained by adding u_i to B. Then $\{u_1, \ldots, u_r\} \setminus \{u_i\}$ and v have to be added to A, $N_i := N(u_i) \cap F$ and $\{v_1, \ldots, v_s\}$ have to be added to B.*
– *For each $1 \le j \le s$, the instance of the $r + j$-th subproblem is obtained by adding v_j to A. Then $\{v_1, \ldots, v_s\} \setminus \{v_j\}$ and v have to be added to B, $M_j := N(v_j) \cap F$ and $\{u_1, \ldots, u_r\}$ have to be added to A.*

Symmetrically for $y \in B$ with $N(y) \cap F = \{u_1, u_2, \ldots, u_r, v\}$ such that $N(u_i) \cap A = \emptyset$, $1 \le i \le r$, and $N(v) \cap A = \{x\}$.

Write $\alpha_i = |N_i \cap \{v_1, \ldots, v_s\}|$, $1 \le i \le r$, and $\beta_j = |M_j \cap \{u_1, \ldots, u_r\}|$, $1 \le j \le s$. The branching vector of Branching Rule 7 is

$$\big(r+s+1+|N_1|-\alpha_1, \ldots, r+s+1+|N_r|-\alpha_r, r+s+1+|M_1|-\beta_1, \ldots, r+s+1+|M_s|-\beta_s\big).$$

By Reduction Rule 7, $|N_i| \ge 2$. By Branching Rule 5, v_j has no neighbor in A, hence, by Reduction Rule 7, $|M_j| \ge 2$. By Branching Rule 6, $\alpha_i \le 1$, $\beta_j \le 1$. Hence the branching factor is at most $\tau(r + s + 2, \ldots, r + s + 2) = \sqrt[r+s+2]{r + s} < 1.2600$.

The description of all seven branching rules is completed. Among all branching rules, Branching Rule 3 has the largest branching factor of 1.2721. Consequently, the running time of our algorithm is $O^*(1.2721^n)$.

It remains to show that if none of the reduction rules and none of the branching rules is applicable to an instance (G, A, B) then the graph G has a perfect matching cut (X, Y) such that $A \subseteq X$ and $B \subseteq Y$ if and only if (A, B) is a perfect matching cut of G. In fact, if all reduction and branching rules are not longer applicable, then no vertex in $A \cup B$ has a neighbor in F. Hence, by connectedness of G, $F = \emptyset$. Therefore, G has a perfect matching cut separating A and B if and only if (A, B) is a perfect matching cut. In summary, we obtain:

Theorem 4. *There is an algorithm for* PERFECT MATCHING CUT *running in $O^*(1.2721^n)$ time.*

4 Two polynomial solvable cases

In this section, we provide two graph classes in which PERFECT MATCHING CUT is solvable in polynomial time. Both classes are well motivated by the hardness results.

4.1 Excluding a (small) tree of maximum degree three

Let H be a fixed graph. A graph G is H-free if G contains no induced subgraph isomorphic to H. Since by Theorem 2 PERFECT MATCHING CUT remains NP-complete on the class of graphs having maximum degree three and arbitrarily high girth, it is also NP-complete on H-free graphs whenever H is outside this class, e.g. if H has a vertex of degree larger than three or has a (fixed-size) cycle. This suggests studying the computational complexity of PERFECT MATCHING CUT restricted to H-free graphs for a fixed forest H with maximum degree at most three.

As the first step in this direction, we show that PERFECT MATCHING CUT is solvable in polynomial time for H-free graphs, where H is the tree T with 6 vertices obtained from the claw $K_{1,3}$ by subdividing two edges each with one new vertex; see Fig. 8. In particular, PERFECT MATCHING CUT is polynomial time solvable for $K_{1,3}$-free graphs but hard for $K_{1,4}$-free graphs (by Theorem 2).

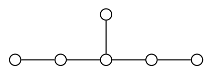


Fig. 8. The tree T.

Given a connected T-free graph $G = (V, E)$, our algorithm works as follows. Fix an edge $ab \in E$ and decide if G has a perfect matching cut $M = E(X, Y)$ separating $A = \{a\}$ and $B = \{b\}$. We use the notations and reduction rules from Section 3. In addition, we need one new reduction rule; recall that $F = V \setminus (A \cup B)$. This additional reduction rule is correct for matching cuts in general and is already used in [7]. For completeness, we give a correctness proof for perfect matching cuts.

Reduction Rule 9

- *If there are vertices $u, v \in F$ with a common neighbor in A and $|N(u)\cap N(v)\cap F| \geq 2$, then $A := A \cup \{u, v\}$.*
- *If there are vertices $u, v \in F$ with a common neighbor in B and $|N(u)\cap N(v)\cap F| \geq 2$, then $B := B \cup \{u, v\}$.*

Proof (of safeness). Let $u, v \in F$ with $N(u)\cap N(v)\cap A = \{x\}$ and $|N(u)\cap N(v)\cap F| \geq 2$. We show that G has a perfect matching cut separating A, B if and only if G has a perfect matching cut separating $A \cup \{u, v\}$ and B. First, let (X, Y) be a perfect matching cut of G with $A \subseteq X$ and $B \subseteq Y$. If $u \in Y$ then, as $x \in A$, $N(u) \cap F$ must belong to Y and v must belong to X. But then, as $|N(v) \cap N(u) \cap F| \geq 2$, v has two neighbors in Y, a contradiction. Thus, $u \in X$, and similarly, $v \in X$. That is (X, Y) separates $A \cup \{u, v\}$ and B. The other direction is obvious: any perfect matching cut separating $A \cup \{u, v\}$ and B separates A and B.

The second case is symmetric. □

Now, we apply the Reduction Rules 1–9 exhaustively. Note that this part takes polynomial time. If $F = V \setminus (A \cup B)$ is empty, then G has a perfect matching cut separating A

and B if and only if (A, B) is a perfect matching cut of G. Verifying whether (A, B) is a perfect matching cut also takes polynomial time.

So, let us assume that $F \neq \emptyset$. Then due to the reduction rules (recall that G is connected),

- any vertex in A (in B) having no neighbor in B (in A) has at least two neighbors in F, and
- any vertex in A (in B) having a neighbor in F has no neighbor in B (in A).

At this point, we will explicitly give an induced subgraph in G isomorphic to the tree T or correctly decide that G has no perfect matching cut separating A and B. Write

$$A^* = \{x \in A \mid N(x) \cap B \neq \emptyset\},\ B^* = \{y \in B \mid N(y) \cap A \neq \emptyset\}.$$

Recall that $A^* \neq \emptyset$ and $B^* \neq \emptyset$, and there are no edges between $A^* \cup B^*$ and F, no edges between $A \setminus A^*$ and $B \setminus B^*$.

Thus, as G is connected and $F \neq \emptyset$, there is a vertex in $A \setminus A^*$ adjacent to a vertex in A^*, or there is a vertex in $B \setminus B^*$ adjacent to a vertex in B^*. By symmetry, let us assume that there is a vertex $x \in A \setminus A^*$ adjacent to a vertex $x^* \in A^*$. Let $y^* \in B^*$ be the unique neighbor of x in B^*. Recall that, every vertex in $(A \setminus A^*) \cup (B \setminus B^*)$ has at least two neighbors in F.

First, suppose that there is a vertex $y \in B$ with $|N(x) \cap N(y) \cap F| \geq 2$. Let $u, v \in N(x) \cap N(y) \cap F$. If (X, Y) is a perfect matching cut with $A \subseteq X$ and $B \subseteq Y$, then u and v must belong to different parts, say $u \in X, v \in Y$. Now, if there were some vertex $w \in N(u) \cap N(v) \cap F$, then u would have two neighbors in Y (if $w \in Y$) or v would have two neighbors in X (if $w \in X$). So, let us assume that $N(u) \cap N(v) \cap F = \emptyset$. Then due to Reduction Rule 5, there exists a vertex $w \in N(u) \cap F \setminus N(v)$. Due to Reduction Rule 3, $N(x) \cap F$ is an independent set, hence w, u, x, x^*, y^* and v induce the tree T in G. Thus, we may assume that

$$\text{for any vertex } y \in B,\ |N(x) \cap N(y)| \leq 1. \tag{1}$$

Next, observe that

$$\text{every vertex in } B \setminus B^* \text{ adjacent to a vertex in } N(x) \text{ is adjacent to } y^*. \tag{2}$$

This can be seen as follows: Let $z \in B \setminus B^*$ be adjacent to some $u \in N(x)$. Then $u \in F$. By (1), z is non-adjacent to all vertices in $N(x) \cap F \setminus \{u\}$. Recall that some vertex $v \in N(x) \cap F \setminus \{u\}$ exists. So, if z is not adjacent to y^*, then z, u, x, x^*, y^* and v induce the tree T in G.

Now, fix two vertices $u, v \in N(x) \cap F$. Suppose that $N(u) \cap B = \emptyset$. Then, due to Reduction Rules 7 and 9, there exists a vertex $w \in N(u) \cap F \setminus N(v)$, and as above, w, u, v, x, x^* and y^* induce the tree T in G. Thus, we may assume that $N(u) \cap B \neq \emptyset$ and, by symmetry, $N(v) \cap B \neq \emptyset$.

Let $y_1, y_2 \in B \setminus B^*$ be the unique neighbors of u and v in B, respectively. By (1), y_1 is non-adjacent to v, and y_2 is non-adjacent to u. By (2), y_1 and y_2 are adjacent to y^*. If y_1 and y_2 are non-adjacent, then u, y_1, y^*, y_2, v and x^* induce the tree T. So, let us assume that y_1 and y_2 are adjacent.

Let $u' \neq u$ be a second neighbor of y_1 in F, and $v' \neq v$ be a second neighbor of y_2 in F. By (1), x is non-adjacent to u' and v'. Now, consider two cases:

– assume that u and v' are adjacent. Then v', u, x, x^*, y^* and v induce the tree T in G, and
– assume that u and v' are non-adjacent. Then v', y_2, y_1, u, x and u' (if u' and v' are non-adjacent), or else u', v', y_2, y^*, x^* and v (if u' and v' are adjacent) induce the tree T in G.

In each case, we reach a contradiction.

Thus, we have seen that, in case $F \neq \emptyset$, G has no perfect matching cut separating A and B, or G contains the tree T as an induced subgraph. So, after at most $|E|$ rounds, each for a candidate $ab \in E$ and in polynomial time, our algorithm will find out whether G has a perfect matching cut at all. In summary, we obtain:

Theorem 5. PERFECT MATCHING CUT *is solvable in polynomial time for T-free graphs.*

4.2 Interval, chordal and pseudo-chordal graphs

Recall that a graph has girth at least g if and only if it has no induced cycles of length less than g. Thus, Theorem 2 implies that PERFECT MATCHING CUT remains hard when restricted to graphs without short induced cycles. This suggests studying PERFECT MATCHING CUT restricted to graphs without long induced cycles, i.e., k-chordal graphs. Here, given an integer $k \geq 3$, a graph is *k-chordal* if it has no induced cycles of length larger than k; the 3-chordal graphs are known as chordal graphs.

In this subsection we show that PERFECT MATCHING CUT can be solved in polynomial time when restricted to what we call pseudo-chordal graphs, that contain the class of 3-chordal graphs and thus known to have unbounded mim-width [16].

We begin with a concise characterization of interval graphs having perfect matching cuts, to yield a polynomial-time algorithm deciding if an interval graph has a perfect matching cut which is much simpler than what we get by the mim-width approach [6].

Fact 1 *Let G have a vertex set $U \subseteq V(G)$ such that $G[U]$ is connected with every edge of $G[U]$ belonging to a triangle. Then if (X, Y) is a perfect matching cut of G we must have $U \subseteq X$ or $U \subseteq Y$.*

This since otherwise we must have a triangle K and two vertices u, v with $u \in K \cap X$ and $v \in K \cap Y$ having a common neighbor in K so this cannot be a perfect matching cut.

If an interval graph G has a cycle then it has a 3-clique. By Fact 1 these 3 vertices would have to belong to the same side of the cut, and each would need to have a unique neighbor on the other side of the cut. But then those 3 neighbors would form an asteroidal triple, contradicting that G was an interval graph. Thus an interval graph which is not a tree does not have a perfect matching cut. A tree T is an interval graph if and only if it does not have the subdivided claw as a subgraph. Thus, T is a caterpillar with basic path $x_1, \ldots, x_k$, where x_1 and x_k does not have a leaf attached, while the other x_i may have any number of leaves attached. If some x_i has at least two leaves attached then T does not have a perfect matching cut (X, Y), as $x_i \in X$ would imply that at least one of those leaves is in X and this leaf would not have a neighbor in Y. Since a leaf vertex and its neighbor must belong to opposite sides of the cut, it is not hard to verify the following.

(A *caterpillar* is a tree with a (basic) path such that all vertices outside the path has a neighbor on the path.)

Fact 2 *An interval graph has a perfect matching cut if and only if it is a caterpillar with basic path $x_1, \ldots, x_k$ such that any x_i for $1 < i < k$ has either zero or one leaf, and any maximal sub-path of $x_1, \ldots, x_k$ with zero leaves contains an even number of vertices.*

In particular, caterpillars having a perfect matching cut can be recognized in polynomial time. For an arbitrary tree T we can decide whether T has a perfect matching as follows: Root T at a vertex r and let $r_1, \ldots, r_k$ be the children of r. Then T has a perfect matching cut if and only if there exists some $1 \leq i \leq k$ such that each subtree T_j rooted at r_j, $j \neq i$, has a perfect matching cut, and $T_i - r_i$ has a perfect matching cut such that all children of r_i are in the same side. This fact implies a bottom-up dynamic programming to decide if T has a perfect matching cut.

A similar idea works for a large graph class that properly contains all chordal graphs. We will show a polynomial-time algorithm for what we call pseudo-chordal graphs. The maximal 2-connected subgraphs of a graph are called its blocks, and a block is non-trivial if it contains at least 3 vertices.

Definition 1. *A graph is* pseudo-chordal *if, for every non-trivial block B, every edge of B belongs to a triangle.*

Note that chordal graphs are pseudo-chordal, but pseudo-chordal graphs may contain induced cycles of any length, e.g. take a cycle and for any two neighbors add a new vertex adjacent to both of them.

Theorem 6. *There is a polynomial-time algorithm deciding if a pseudo-chordal graph G has a perfect matching cut.*

Proof. We first compute the blocks of G and let D be the subgraph of G formed by the edges of non-trivial blocks of G. Let $D_1, D_2, \ldots, D_k$ be the connected components of D. Note that by collapsing each D_i into a supernode we can treat the graph G as having a tree structure T (related to the block structure) with one node for each $v \in V(G) \setminus V(D)$, and a supernode for each D_i. See Fig. 9.

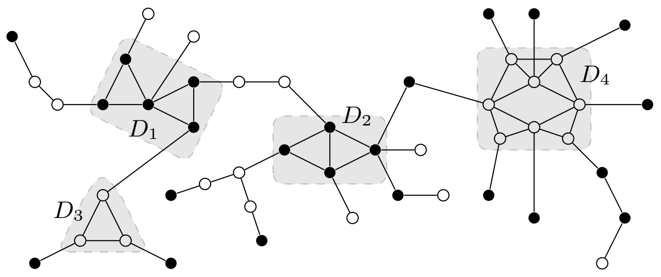


Fig. 9. A pseudo-chordal graph and perfect matching cut given by (X, Y) with X being black vertices. Note the tree structure composed of (i) those vertices that do not belong to a clique of size 3 and (ii) the four supernodes D_1, D_2, D_3, D_4.

Note that since G is pseudo-chordal then by Fact 1 all the vertices in a fixed supernode D_i must be on the same side in any perfect matching cut of G.

Our algorithm will pick a root R of T and proceed by bottom-up dynamic programming on the rooted tree T. Each node S of T will be viewed as the set of vertices it represents in G. If S is not the root of T then we denote by $r(S)$ the unique vertex of S that has a parent in T. For each node S of T we will compute two boolean values that concern the subgraph G_S of G induced by vertices of G contained in the subtree of T rooted at S. These boolean values are defined as follows:

– $pmc(S) =$ `true` if and only if G_S has a perfect matching cut
– $m(S) =$ `true` if and only if $G_S \setminus r(S)$ has a perfect matching cut where all vertices of $S \setminus r(S)$ are on the same side.

We first initialize $pmc(S)$ and $m(S)$ to `false` for all nodes S of T. For a leaf S of T we set $m(S) =$ `true` if $|S| = 1$, i.e. if S is not a supernode.

Consider an inner node S of T, with $S = \{v_1, \ldots, v_q\} \subseteq V(G)$. In the rooted tree T, let the children of S that contain a neighbor of v_i be $C(v_i)$ (note that each child of S in T has a unique vertex that has a unique neighbor $v_i \in S$). Assuming the values $pmc(\cdot)$ and $m(\cdot)$ have been computed for all children of S, we do the following:

– set $pmc(S) =$ `true` if for each $v_i \in S$ we have $C(v_i) = \{S_1, \ldots, S_k\}$ with $k \geq 1$ and we can find a child with $m(S_i) =$ `true` such that $pmc(S_j) =$ `true` for the other $k-1$ children $j \neq i$.
– set $m(S) =$ `true` if (i) for each $v_i \in S \setminus r(S)$ we have $C(v_i) = \{S_1, \ldots, S_k\}$ with $k \geq 1$ and we can find a child with $m(S_i) =$ `true` such that $pmc(S_j) =$ `true` for the other $k-1$ children $j \neq i$, and (ii) for every child $S' \in C(r(S))$ we have $pmc(S') =$ `true`.

For the root R of T we update $pmc(R)$ but not $m(R)$ since $r(R)$ is not defined. When we are done with the bottom-up dynamic programming, then for the root R of T we note that $G = G_R$ so that by the definition of the values G has a perfect matching cut if and only if $pmc(R) =$ `true`.

The correctness follows by structural induction on the tree T. By definition, $pmc(S) =$ `true` (respectively $m(S) =$ `true`) if and only if there is a cut of G_S so that every node (respectively every node except $r(S)$) has a single neighbor, its 'mate', on the other side of the cut. The values at leaves are initialized correctly according to this definition. At an inner node S we inductively assume the values at children are correct and end up setting $pmc(S)$ to `true` if and only if G_S has a perfect matching cut, since for each node in S we require a single child neighbor that needs a mate, while all other child neighbors are required to already have a mate. Similarly for $m(S)$ but now all children of $r(S)$ are required to already have a mate. Since each node v of S is a cut vertex of G separating G_S so that each child of S defines its own unique component, we can merge all the cuts in all children while keeping all the nodes of S on the same side of the cut, to satisfy Fact 1 that requires all the nodes in S to be on the same side of the cut. The runtime is clearly polynomial. □

5 Conclusion

We have shown that, assuming ETH, there is no $O^*(2^{o(n)})$-time algorithm for PERFECT MATCHING CUT even when restricted to n-vertex bipartite graphs, and that PERFECT MATCHING CUT remains NP-complete when restricted to bipartite graphs of maximum degree 3 and arbitrary large girth. This implies that PERFECT MATCHING CUT remains NP-complete when restricted to H-free graphs where H is any fixed graph having a vertex of degree at least 4 or a cycle. This suggests the following problem for further research:

> Let F be a fixed forest with maximum degree at most 3. What is the computational complexity of PERFECT MATCHING CUT restricted to F-free graphs?

We have proved a first polynomial case for this problem where F is a certain 6-vertex tree, including claw-free graphs and graphs without an induced 5-path.

Our hardness result also suggests studying PERFECT MATCHING CUT restricted to graphs without long induced cycles:

> What is the computational complexity of PERFECT MATCHING CUT on k-chordal graphs?

It follows from our results that PERFECT MATCHING CUT is polynomially solvable for 3-chordal graphs.

We have also given an exact branching algorithm for PERFECT MATCHING CUT running in $O^*(1.2721^n)$ time. It is natural to ask whether the running time of the branching algorithm can be improved. Finally, as for matching cuts, also for perfect matching cuts it would be interesting to study counting and enumeration as well as FPT and kernelization algorithms.

References

1. N. R. Aravind, Subrahmanyam Kalyanasundaram, and Anjeneya Swami Kare. On structural parameterizations of the matching cut problem. In *Combinatorial Optimization and Applications - 11th International Conference, COCOA 2017, Shanghai, China, December 16-18, 2017, Proceedings, Part II*, pages 475–482, 2017. doi:10.1007/978-3-319-71147-8_34.
2. N. R. Aravind and Roopam Saxena. An FPT algorithm for matching cut. *CoRR*, abs/2101.06998, 2021. URL: https://arxiv.org/abs/2101.06998, arXiv:2101.06998.
3. Rémy Belmonte and Martin Vatshelle. Graph classes with structured neighborhoods and algorithmic applications. *Theor. Comput. Sci.*, 511:54–65, 2013. doi:10.1016/j.tcs.2013.01.011.
4. Paul S. Bonsma. The complexity of the matching-cut problem for planar graphs and other graph classes. *Journal of Graph Theory*, 62(2):109–126, 2009. URL: http://dx.doi.org/10.1002/jgt.20390, doi:10.1002/jgt.20390.
5. Valentin Bouquet and Christophe Picouleau. The complexity of the perfect matching-cut problem. *CoRR*, abs/2011.03318, 2020. URL: https://arxiv.org/abs/2011.03318, arXiv:2011.03318.
6. Binh-Minh Bui-Xuan, Jan Arne Telle, and Martin Vatshelle. Fast dynamic programming for locally checkable vertex subset and vertex partitioning problems. *Theor. Comput. Sci.*, 511:66–76, 2013. doi:10.1016/j.tcs.2013.01.009.
7. Chi-Yeh Chen, Sun-Yuan Hsieh, Hoàng-Oanh Le, Van Bang Le, and Sheng-Lung Peng. Matching cut in graphs with large minimum degree. *Algorithmica*, 83(5):1238–1255, 2021. doi:10.1007/s00453-020-00782-8.

8. Vasek Chvátal. Recognizing decomposable graphs. *Journal of Graph Theory*, 8(1):51–53, 1984. URL: http://dx.doi.org/10.1002/jgt.3190080106, doi:10.1002/jgt.3190080106.
9. Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Springer, 2010.
10. Petr A. Golovach, Christian Komusiewicz, Dieter Kratsch, and Van Bang Le. Refined notions of parameterized enumeration kernels with applications to matching cut enumeration. In Markus Bläser and Benjamin Monmege, editors, *38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021, March 16-19, 2021, Saarbrücken, Germany (Virtual Conference)*, volume 187 of *LIPIcs*, pages 37:1–37:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.STACS.2021.37.
11. Guilherme C. M. Gomes and Ignasi Sau. Finding cuts of bounded degree: Complexity, FPT and exact algorithms, and kernelization. In *14th International Symposium on Parameterized and Exact Computation, IPEC 2019, September 11-13, 2019, Munich, Germany*, pages 19:1–19:15, 2019. doi:10.4230/LIPIcs.IPEC.2019.19.
12. Ron L. Graham. On primitive graphs and optimal vertex assignments. *Ann. N. Y. Acad. Sci.*, 175(1):170–186, 1970.
13. Pinar Heggernes and Jan Arne Telle. Partitioning graphs into generalized dominating sets. *Nord. J. Comput.*, 5(2):128–142, 1998.
14. Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.
15. Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.
16. Dong Yeap Kang, O-joung Kwon, Torstein J. F. Strømme, and Jan Arne Telle. A width parameter useful for chordal and co-comparability graphs. *Theor. Comput. Sci.*, 704:1–17, 2017. doi:10.1016/j.tcs.2017.09.006.
17. Christian Komusiewicz, Dieter Kratsch, and Van Bang Le. Matching cut: Kernelization, single-exponential time fpt, and exact exponential algorithms. *Discret. Appl. Math.*, 283:44–58, 2020. doi:10.1016/j.dam.2019.12.010.
18. Dieter Kratsch and Van Bang Le. Algorithms solving the matching cut problem. *Theor. Comput. Sci.*, 609:328–335, 2016. URL: http://dx.doi.org/10.1016/j.tcs.2015.10.016, doi:10.1016/j.tcs.2015.10.016.
19. Hoàng-Oanh Le and Van Bang Le. A complexity dichotomy for matching cut in (bipartite) graphs of fixed diameter. *Theor. Comput. Sci.*, 770:69–78, 2019. doi:10.1016/j.tcs.2018.10.029.
20. Augustine M. Moshi. Matching cutsets in graphs. *Journal of Graph Theory*, 13(5):527–536, 1989. URL: http://dx.doi.org/10.1002/jgt.3190130502, doi:10.1002/jgt.3190130502.
21. Jan Arne Telle and Andrzej Proskurowski. Algorithms for vertex partitioning problems on partial k-trees. *SIAM J. Discret. Math.*, 10(4):529–550, 1997. doi:10.1137/S0895480194275825.
22. Martin Vatshelle. *New width parameters of graphs*. PhD thesis, University of Bergen, Norway, 2012. URL: https://bora.uib.no/bora-xmlui/handle/1956/6166.