Finding d-Cuts in Graphs of Bounded Diameter, Graphs of Bounded Radius and H-Free Graphs

Felicia Lucke[1][0000−0002−9860−2928], Ali Momeni[2][0009−0009−8280−7847], Daniël Paulusma[3][0000−0001−5945−9287], and Siani Smith[4][0000−0003−0797−0512]

[1] Department of Informatics, University of Fribourg, Fribourg, Switzerland
felicia.lucke@unifr.ch
[2] Faculty of Computer Science, UniVie Doctoral School Computer Science DoCS, University of Vienna, Austria
ali.momeni@univie.ac.at
[3] Department of Computer Science, Durham University, Durham, UK
daniel.paulusma@durham.ac.uk
[4] University of Bristol, Heilbronn Institute for Mathematical Research, Bristol, UK
siani.smith@bristol.ac.uk

Abstract. The d-Cut problem is to decide if a graph has an edge cut such that each vertex has at most d neighbours at the opposite side of the cut. If $d = 1$, we obtain the intensively studied Matching Cut problem. The d-Cut problem has been studied as well, but a systematic study for special graph classes was lacking. We initiate such a study and consider classes of bounded diameter, bounded radius and H-free graphs. We prove that for all $d \geq 2$, d-Cut is polynomial-time solvable for graphs of diameter 2, $(P_3 + P_4)$-free graphs and P_5-free graphs. These results extend known results for $d = 1$. However, we also prove several NP-hardness results for d-Cut that contrast known polynomial-time results for $d = 1$. Our results lead to full dichotomies for bounded diameter and bounded radius and to partial dichotomies for H-free graphs; for $d \geq 3$, our classification of d-Cut for H-free graphs only has three open cases.

Keywords: matching cut · d-cut · diameter · H-free graph

1 Introduction

We consider the generalization d-Cut of a classic graph problem Matching Cut (1-Cut). First, we explain the original graph problem. Consider a connected graph $G = (V, E)$, and let $M \subseteq E$ be a subset of edges of G. The set M is an *edge cut* of G if it is possible to partition V into two non-empty sets B (set of *blue* vertices) and R (set of *red* vertices) in such a way that M is the set of all edges with one end-vertex in B and the other one in R. Now, suppose that M is in addition also a *matching*, that is, no two edges in M have a common end-vertex. Then M is said to be a *matching cut*. See Figure 1 for an example.

Graphs with matching cuts were introduced in the context of number theory [14] and have various other applications [1, 8, 11, 26]. The Matching Cut

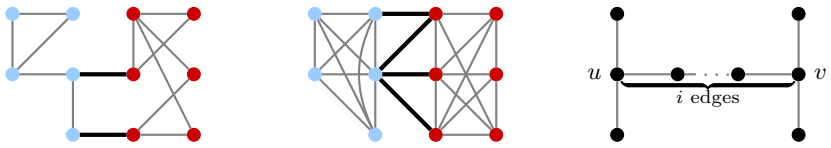

Fig. 1: Left: a graph with a matching cut (1-cut). Middle: a graph with a 3-cut but no d-cut for $d \leq 2$. Right: the graph H_i^*.

problem is to decide if a connected graph has a matching cut. This problem was shown to be NP-complete by Chvátal [7]. Several variants and generalizations of matching cuts are known. In particular, a *perfect matching cut* is a matching cut that is a perfect matching, whereas a *disconnected perfect matching* is a perfect matching containing a matching cut. The corresponding decision problems Perfect Matching Cut [15] and Disconnected Perfect Matching [6] are also NP-complete; see [3, 10, 18, 20, 24] for more complexity results for these two problems. The optimization versions Maximum Matching Cut and Minimum Matching Cut are to find a matching cut of maximum and minimum size in a connected graph, respectively; see [19, 23] for more details.

Our Focus. Matching cuts have also been generalized as follows. For an integer $d \geq 1$ and a connected graph $G = (V, E)$, a set $M \subseteq E$ is a *d-cut* of G if it is possible to partition V into two non-empty sets B and R, such that: (i) the set M is the set of all edges with one end-vertex in B and the other one in R; and (ii) every vertex in B has at most d neighbours in R, and vice versa (see also Figure 1). Note that a 1-cut is a matching cut. We consider the d-Cut problem: does a connected graph have a d-cut? Here, $d \geq 1$ is a fixed integer, so not part of the input. Note that 1-Cut is Matching Cut. The d-Cut problem was introduced by Gomes and Sau [13] who proved its NP-completeness for all $d \geq 1$.

Our Goal. To get a better understanding of the hardness of an NP-complete graph problem, it is natural to restrict the input to belong to some special graph classes. We will first give a brief survey of the known complexity results for Matching Cut and d-Cut for $d \geq 2$ under input restrictions. As we will see, for Matching Cut many more results are known than for d-Cut with $d \geq 2$. Our goal is to obtain the same level of understanding of the d-Cut problem for $d \geq 2$. This requires a currently lacking systematic study into the complexity of this problem. We therefore consider the following research question:

For which graph classes $\mathcal{G}$ does the complexity of d-Cut, restricted to graphs from $\mathcal{G}$, change if $d \geq 2$ instead of $d = 1$?

As *testbeds* we take classes of graphs of bounded diameter, graphs of bounded radius and H-free graphs. The *distance* between two vertices u and v in a connected graph G is the *length* (number of edges) of a shortest path between u and v in G. The *eccentricity* of a vertex u is the maximum distance between u and any other vertex of G. The *diameter* of G is the maximum eccentricity over all vertices of G, whereas the *radius* of G is the minimum eccentricity over all

vertices of G. A graph G is *H-free* if G does not contain a graph H as an *induced subgraph*, that is, G cannot be modified into H by vertex deletions.

Existing Results. We focus on classical complexity results; see [2, 13] for exact and parameterized complexity results for d-Cut. Let $K_{1,r}$ denote the $(r+1)$-vertex star, which has vertex set $\{u, v_1, \ldots, v_r\}$ and edges uv_i for $i \in \{1, \ldots, r\}$. Chvátal [7] showed that Matching Cut is NP-complete even for $K_{1,4}$-free graphs of maximum degree 4, but polynomial-time solvable for graphs of maximum degree at most 3. Gomes and Sau [13] extended these results by proving that for every $d \geq 2$, the d-Cut problem is NP-complete for graphs in which every vertex has degree $2d+2$, but polynomial-time solvable for graphs of maximum degree at most $d+2$. Feghali et al. [10] proved that for every $d \geq 1$ and every $g \geq 3$, there is a function $f(d)$, such that d-Cut is NP-complete for bipartite graphs of girth at least g and maximum degree at most $f(d)$. The *girth* of a graph G that is not a forest is the length of a shortest induced cycle in G. It is also known that Matching Cut is polynomial-time solvable for graphs of diameter at most 2 [5, 17], and even radius at most 2 [22], while being NP-complete for graphs of diameter 3 [17], and thus radius at most 3. Hence, we obtain:

Theorem 1 ([17, 22]). *For $r \geq 1$,* Matching Cut *is polynomial-time solvable for graphs of diameter r and graphs of radius r if $r \leq 2$ and* NP*-complete if $r \geq 3$.*

To study a problem in a systematic way on graph classes that can be characterized by forbidden induced subgraphs, an often used approach is to first focus on the classes of H-free graphs. As Matching Cut is NP-complete for graphs of girth g for every $g \geq 3$ [10] and for $K_{1,4}$-free graphs [7], Matching Cut is NP-complete for H-free graphs whenever H has a cycle or is a forest with a vertex of degree at least 4. What about when H is a forest of maximum degree 3?

We let P_t be the path on t vertices. We denote the disjoint union of two vertex-disjoint graphs $G_1 + G_2$ by $G_1 + G_2 = (V(G_1) \cup V(G_2), E(G_1) \cup E(G_2))$. We let sG be the disjoint union of s copies of G. Feghali [9] proved the existence of an integer t such that Matching Cut is NP-complete for P_t-free graphs, which was narrowed down to $(3P_5, P_{15})$-free graphs in [24] and to $(3P_6, 2P_7, P_{14})$-free graphs in [18]. Let H_1^* be the "H"-graph, which has vertices u, v, w_1, w_2, x_1, x_2 and edges $uv, uw_1, uw_2, vx_1, vx_2$. For $i \geq 2$, let H_i^* be the graph obtained from H_1^* by subdividing uv exactly $i-1$ times; see Figure 1. It is known that Matching Cut is NP-complete for $(H_1^*, H_3^*, H_5^*, \ldots)$-free bipartite graphs [25] and for $(H_1^*, \ldots, H_i^*)$-free graphs for every $i \geq 1$ [10].

On the positive side, Matching Cut is polynomial-time solvable for *claw-free* graphs ($K_{1,3}$-free graphs) and for P_6-free graphs [24]. Moreover, if Matching Cut is polynomial-time solvable for H-free graphs for some graph H, then it is so for $(H + P_3)$-free graphs [24].

For two graphs H and H', we write $H \subseteq_i H'$ if H is an induced subgraph of H'. Combining the above yields a partial classification (see also [10, 23]):

Theorem 2 ([4, 7, 10, 18, 22, 24, 25]). *For a graph H,* Matching Cut *on H-free graphs is*

- *polynomial-time solvable if* $H \subseteq_i sP_3 + K_{1,3}$ *or* $sP_3 + P_6$ *for some* $s \geq 0$*;*
- NP*-complete if* $H \supseteq_i K_{1,4}$*,* P_{14}*,* $2P_7$*,* $3P_5$*,* C_r *for some* $r \geq 3$*, or* H_i^* *for some* $i \geq 1$*.*

Our Results. We first note that d-Cut is straightforward to solve for graphs of radius 1 (i.e., graphs with a dominating vertex) and extend Theorem 1:

Theorem 3. *Let* $d \geq 2$*. For* $r \geq 1$*,* d-Cut *is polynomial-time solvable for graphs of diameter* r *if* $r \leq 2$ *and* NP*-complete if* $r \geq 3$*.*

Theorem 4. *Let* $d \geq 2$*. For* $r \geq 1$*,* d-Cut *is polynomial-time solvable for graphs of radius* r *if* $r \leq 1$ *and* NP*-complete if* $r \geq 2$*.*

Comparing Theorem 1 with Theorems 3 and 4 shows no difference in complexity for diameter but a complexity jump from $d = 1$ to $d = 2$ for radius.

For every $d \geq 2$, we also give polynomial-time algorithms for d-Cut for $(P_3 + P_4)$-free graphs and P_5-free graphs. Our proof techniques use novel arguments, as we can no longer rely on a polynomial-time algorithm for radius 2 or a reduction to 2-SAT as for $d = 1$ [17, 22]. Moreover, we show that for $d \geq 2$, d-Cut is polynomial-time solvable for $(H + P_1)$-free graphs whenever d-Cut is so for H-free graphs, thus the cases $\{H + sP_1 \mid s \geq 0\}$ are all *(polynomially) equivalent*. All these results extend the known results for $d = 1$, as can be seen from Theorem 2.

As negative results, we prove that d-Cut is NP-complete for $3P_3$-free graphs for $d = 2$, which we can strengthen to $3P_2$-free graphs for $d \geq 3$. We also show that for $d \geq 2$, d-Cut is NP-complete for $(H_1^*, H_2^*, \ldots, H_i^*)$-free graphs for every $i \geq 1$. Finally, we prove that d-Cut is NP-complete for $K_{1,4}$-free graphs for $d = 2$, which we can strengthen to line graphs for $d \geq 3$. The NP-completeness for graphs of large girth from [10] implies that for $d \geq 2$, d-Cut is NP-complete for H-free graphs if H has a cycle. Hence, by combining the above results, we obtain the following two partial complexity classifications for $d = 2$ and $d \geq 3$:

Theorem 5. *For a graph* H*,* 2-Cut *on* H*-free graphs is*

- *polynomial-time solvable if* $H \subseteq_i sP_1 + P_3 + P_4$ *or* $sP_1 + P_5$ *for some* $s \geq 0$*;*
- NP*-complete if* $H \supseteq_i K_{1,4}$*,* $3P_3$*,* C_r *for some* $r \geq 3$*, or* H_i^* *for some* $i \geq 1$*.*

Theorem 6. *Let* $d \geq 3$*. For a graph* H*,* d-Cut *on* H*-free graphs is*

- *polynomial-time solvable if* $H \subseteq_i sP_1 + P_3 + P_4$ *or* $sP_1 + P_5$ *for some* $s \geq 0$*;*
- NP*-complete if* $H \supseteq_i K_{1,3}$*,* $3P_2$*,* C_r *for some* $r \geq 3$*, or* H_i^* *for some* $i \geq 1$*.*

Theorem 5 leaves an infinite number of non-equivalent open cases for $d = 2$. In contrast, Theorem 6 leaves only three non-equivalent open cases for $d \geq 3$, namely when $H = 2P_4$, $H = P_6$ and $H = P_7$.

From Theorems 2–6 we can make the following observations. First, the case where $H = K_{1,3}$ is still open for $d = 2$. We could only prove NP-completeness of d-Cut for $K_{1,4}$-free graphs. However, for $K_{1,3}$-free graphs, there is still a complexity jump from $d = 1$ to $d = 3$ (which might possibly occur even at

$d = 2$). Second, there are complexity jumps from $d = 1$ to $d = 2$ for sP_3-free graphs when $s = 3$, and from $d = 1$ to $d = 3$ for sP_2-free graphs when $s = 3$; we do not know if the latter jump even occurs at $d = 2$.

We prove our polynomial-time results in Section 3 and our NP-completeness results in Section 4. We finish our paper with several open problems in Section 5. We start with providing some basic results in Section 2.

2 Preliminaries

The *line graph* $L(G)$ of a graph $G = (V, E)$ has the edges of G as its vertices, with an edge between two vertices in $L(G)$ if and only if the corresponding edges in G share an end-vertex. Let $u \in V$. The set $N(u) = \{v \in V \mid uv \in E\}$ is the *neighbourhood* of u, and $|N(u)|$ is the *degree* of u. Let $S \subseteq V$. The *neighbourhood* of S is the set $N(S) = \bigcup_{u \in S} N(u) \setminus S$, and $G[S]$ denotes the subgraph of G *induced* by S. If every vertex of $V \setminus S$ has a neighbour in S, then S is a *dominating* set of G. We also say that $G[S]$ *dominates* G. The *domination number* of G is the size of a smallest dominating set of G. Let $T \subseteq V \setminus S$. The sets S and T are *complete* to each other if every vertex of S is adjacent to every vertex of T.

A *red-blue colouring* of a graph G assigns every vertex of G either the colour red or blue (see, e.g., [9, 22]). For $d \geq 1$, a red-blue colouring is a *red-blue d-colouring* if every blue vertex has at most d red neighbours; every red vertex has at most d blue neighbours; and both colours red and blue are used at least once. See Figure 1 for examples of a red-blue 1-colouring and a red-blue 3-colouring.

Observation 7 *For every $d \geq 1$, a connected graph G has a d-cut if and only if it has a red-blue d-colouring.*

If every vertex of a set $S \subseteq V$ has the same colour (either red or blue) in a red-blue colouring, then S, and also $G[S]$, are *monochromatic*. An edge with a blue and a red end-vertex is *bichromatic*. Note that for every $d \geq 1$, the graph K_{2d+1} is monochromatic in every red-blue d-colouring, and that every connected graph with a red-blue d-colouring contains a bichromatic edge.

We now generalize a known lemma for Matching Cut (see, e.g., [22])).

Lemma 8. *For $d, g \geq 1$, it is possible to find in $O(2^g n^{dg+2})$-time a red-blue d-colouring (if it exists) of a graph G with n vertices and domination number g.*

Proof. Let $d, g \geq 1$ and G be a graph on n vertices with domination number g. Let D be a dominating set D of G that has size at most g.

We consider all $2^{|D|} \leq 2^g$ options of giving the uncoloured vertices of D either colour red or blue. For each red-blue colouring of D we do as follows. For every red vertex of D, we consider all $O(n^d)$ options of colouring at most d of its uncoloured neighbours blue, and we colour all of its other uncoloured neighbours red. Similarly, for every blue vertex of D, we consider all $O(n^d)$ options of colouring at most d of its uncoloured neighbours red, and we colour all of its other uncoloured neighbours blue. As D dominates G, we obtained a

red-blue colouring c of the whole graph G. We discard the option if c is not a red-blue d-colouring of G.

We note that any red-blue d-colouring of G, if it exists, will be found by the above algorithm. As the total number of options is $O(2^g n^{dg})$ and checking if a red-blue colouring is a red-blue d-colouring takes $O(n^2)$ time, our algorithm has total running time $O(2^g n^{dg+2})$. □

Let $X = \{x_1, x_2, ..., x_n\}$ be a set of variables and $\mathcal{C} = \{C_1, C_2, ..., C_m\}$ be a set of clauses over X. The problem Not-All-Equal Satisfiability asks if $(X, \mathcal{C})$ has a *satisfying not-all-equal truth assignment* ϕ that is, ϕ sets, in each C_i, at least one literal true and at least one literal false. The next theorem is well-known (where the second part follows from a folklore trick; see e.g. [12]).

Theorem 9 ([27]). Not-All-Equal Satisfiability *is* NP*-complete even for*

- *instances, in which each literal occurs only positively, and in which each clause contains exactly three literals;*
- *instances, in which each literal occurs only positively and in two or three different clauses, and in which each clause contains either two or three literals.*

3 Polynomial-Time Results

We now show our polynomial-time results for d-Cut for $d \geq 2$, which complement corresponding known polynomial-time results for $d = 1$ (see Section 1) and their proofs yield alternative proofs for $d = 1$. We omit the proof of our first result.

Theorem 10. *For $d \geq 2$, d-Cut is polynomial-time solvable for graphs of diameter at most 2.*

Theorem 11. *For $d \geq 2$, d-Cut is polynomial-time solvable for P_5-free graphs.*

Proof. Let $d \geq 2$. Let $G = (V, E)$ be a connected P_5-free graph on n vertices. As G is P_5-free and connected, G has a dominating set D, such that either D induces a cycle on five vertices or D is a clique [21]. Moreover, we find such a dominating set D in $O(n^3)$ time [16]. If $|D| \leq 3d$, then we apply Lemma 8. Now assume that $|D| \geq 3d + 1 \geq 7$, and thus D is a clique. If $D = V$, then G has no d-cut, as $|D| \geq 3d + 1$ and D is a clique. Assume that $D \subsetneq V$, so $G - D$ has at least one connected component. Below we explain how to find in polynomial time a d-cut of G, or to conclude that G does not have a d-cut. By Observation 7, we need to decide in polynomial time if G has a red-blue d-colouring.

We first enter the *blue phase* of our algorithm. As $|D| \geq 3d + 1$ and D is a clique, D is monochromatic in any red-blue d-colouring of G. Hence, we may colour, without loss of generality, all the vertices of D blue, and we may colour all vertices of every connected component of $G - D$ except one connected component blue as well. We branch over all $O(n)$ options of choosing the connected

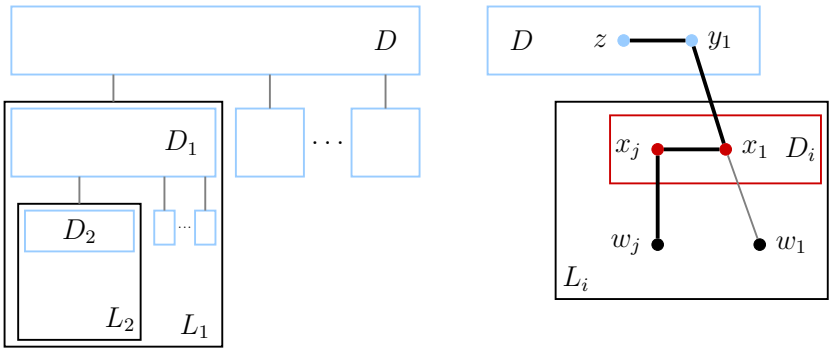

Fig. 2: The graph G in the blue phase (left) and in the red phase: Case 2 (right).

component L_1 of $G - D$ that will contain a red vertex.[5] For each option, we are going to repeat the process of colouring vertices blue until we colour at least one vertex red. We first explain how this process works if we do not do this last step.

As L_1 is connected and P_5-free, we find in $O(n^3)$ time (using the algorithm of [16]) a dominating set D_1 of L_1 that is either a cycle on five vertices or a clique. We colour the vertices of D_1 blue. As we have not used the colour red yet, we colour all vertices of every connected component of $L_1 - D_1$ except one connected component blue. So, we branch over all $O(n)$ options of choosing the connected component L_2 of $L_1 - D_1$ that will contain a red vertex. Note that every uncoloured vertex of G, which belongs to L_2, has both a neighbour in D (as D dominates G) and a neighbour in D_1 (as D_1 dominates L_1). We now find a dominating set D_2 of L_2 and a connected component L_3 of $G - D_2$ with a dominating set D_3, and so on. See also Figure 2.

If we repeat the above process more than d times, we have either coloured every vertex of G blue, or we found $d+1$ pairwise disjoint, blue sets $D, D_1, \ldots, D_d$, such that every uncoloured vertex u has a neighbour in each of them. The latter implies that u has $d+1$ blue neighbours, so u must be coloured blue as well. Hence, in each branch, we would eventually end up with the situation where all vertices of G will be coloured blue. To prevent this from happening, we must colour, in each branch, at least one vertex of D_i red, for some $1 \leq i \leq d-1$. As soon as we do this, we end the blue phase for the branch under consideration, and our algorithm enters the *red phase*.

Each time we have $O(n)$ options to select a connected component L_i and, as argued above, we do this at most d times. Hence, we enter the red phase for $O(n^d)$ branches in total. From now on, we call these branches the *main branches*

[5] For $d = 1$, up to now, the same approach is used for P_5-free graphs [9]. But the difference is that for $d = 1$, the algorithm and analysis is much shorter: one only has to check, in this stage, if there is a component of $G - D$, whose vertices can all be safely coloured red. Then one either finds a matching cut, or G has no matching cut.

of our algorithm. For a main branch, we say that we *quit the blue phase at level i* if we colour at least one vertex of D_i red. If we quit the blue phase for a main branch at level i, for some $1 \leq i \leq d-1$, then we have constructed, in polynomial time, pairwise disjoint sets D, $D_1, \ldots, D_i$ and graphs $L_1, \ldots, L_i$, such that:

- for every $h \in \{1, \ldots, i-1\}$, L_{h+1} is a connected component of $L_h - D_h$;
- every vertex of G that does not belong to L_i has been coloured blue;
- for every $h \in \{1, \ldots, i\}$, D_h induces a cycle on five vertices or is a clique;
- D dominates G, so, in particular, D dominates L_i; and
- for every $h \in \{1, \ldots, i\}$, D_h dominates L_h.

We now prove the following claim, which shows that we branched correctly.

Claim 11.1 *The graph G has a red-blue d-colouring that colours every vertex of D blue if and only if we have quit a main branch at level i for some $1 \leq i \leq d-1$, such that G has a red-blue d-colouring that colours at least one vertex of D_i red and all vertices not in L_i blue.*

Proof of the Claim. Suppose G has a red-blue d-colouring c that colours every vertex of D blue (the reverse implication is immediate). By definition, c has coloured at least one vertex u in $G - D$ red. As we branched in every possible way, there is a main branch that quits the blue phase at level i, such that u belongs to L_i for some $1 \leq i \leq d-1$. We pick a main branch with largest possible i, so u is not in L_{i+1}. Hence, $u \in D_i$. We also assume that no vertex u' that belongs to some D_h with $h < i$ is coloured red by c, as else we could take u' instead of u. Hence, c has coloured every vertex in $D_1 \cup \ldots \cup D_{i-1}$ (if $i \geq 2$) blue. Therefore, we may assume that every vertex $v \notin D \cup D_1 \cup \ldots \cup D_{i-1} \cup V(L_i)$ has been coloured blue by c. If not, may just recolour all such vertices v blue for the following reasons: u is still red and no neighbour of v is red, as after a possible recolouring all red vertices belong to L_i, while v belongs to a different component of $G - (D \cup D_1 \cup \cdots \cup D_{i-1})$ than L_i. So, we proved the claim. $\lhd$

Claim 11.1 allows us to do some specific branching once we quit the blue phase for a certain main branch at level i. Namely, all we have to do is to consider all options to colour at least one vertex of D_i red. We call these additional branches *side branches*. We distinguish between the following two cases:

Case 1. $|D_i| \leq 2d+1$.
We consider each of the at most 2^{2d+1} options to colour the vertices of D_i either red or blue, such that at least one vertex of D_i is coloured red. Next, for each vertex $u \in D_i$, we consider all $O(n^d)$ options to colour at most d of its uncoloured neighbours blue if u is red, or red if u is blue. Note that the total number of side branches is $O(2^{2d+1} n^{d(2d+1)})$. As D_i dominates L_i and the only uncoloured vertices were in L_i, we obtained a red-blue colouring c of G. We check in polynomial time if c is a red-blue d-colouring of G. If so, we stop and return c. If none of the side branches yields a red-blue d-colouring of G, then by Claim 11.1 we can safely discard the main branch under consideration.

Case 2. $|D_i| \geq 2d+2$.
Recall that D_i either induces a cycle on five vertices or is a clique. As $|D_i| \geq 2d+2 \geq 6$, we find that D_i is a clique. As $|D_i| \geq 2d+2$, this means that D_i must be monochromatic, and thus every vertex of D_i must be coloured red. We check in polynomial time if D_i contains a vertex with more than d neighbours in D (which are all coloured blue), or if D contains a vertex with more than d neighbours in D_i (which are all coloured red). If so, we may safely discard the main branch under consideration due to Claim 11.1.

From now on, assume that every vertex in D_i has at most d neighbours in D, and vice versa. By construction, every uncoloured vertex belongs to $L_i - D_i$. Hence, if $V(L_i) = D_i$, we have obtained a red-blue colouring c of G. We check, in polynomial time, if c is also a red-blue d-colouring of G. If so, we stop and return c. Otherwise, we may safely discard the main branch due to Claim 11.1.

Now assume $V(L_i) \supsetneq D_i$. We colour all vertices in $L_i - D_i$ that are adjacent to at least $d+1$ vertices in D blue; we have no choice as the vertices in D are all coloured blue. If there are no uncoloured vertices left, we check in polynomial time if the obtained red-blue colouring is a red-blue d-colouring of G. If so, we stop and return it; else we may safely discard the main branch due to Claim 11.1.

Assume that we still have uncoloured vertices left. We recall that these vertices belong to $L_i - D_i$, and that by construction they have at most d neighbours in D. Consider an uncoloured vertex w_1. As D_i dominates L_i, we find that w_1 has a neighbour x_1 in D_i. As D dominates G, we find that x_1 is adjacent to some vertex $y_1 \in D$. We consider all $O(n^{2d})$ possible ways to colour the uncoloured neighbours of x_1 and y_1, such that x_1 (which is red) has at most d blue neighbours, and y_1 (which is blue) has at most d red neighbours. If afterwards there is still an uncoloured vertex w_2, then we repeat this process: we choose a neighbour x_2 of w_2 in D_i (so x_2 is coloured red). We now branch again by colouring the uncoloured neighbours of x_2. If we find an uncoloured vertex w_3, then we find a neighbour x_3 of w_3 in D_i and so on. So, we repeat this process until there are no more uncoloured vertices. This gives us $2p$ distinct vertices $w_1, \ldots, w_p, x_1, \ldots, x_p$ for some integer $p \geq 1$, together with vertex y_1. The total number of side branches for the main branch is $O(n^{2pd})$.

We claim that $p \leq d$. For a contradiction, assume that $p \geq d+1 \geq 3$. Let $2 \leq j \leq p$. As D_i is a clique that contains x_1 and x_j, we find that x_1 and x_j are adjacent. Hence, G contains the 4-vertex path $w_j x_j x_1 y_1$. By construction, w_j was uncoloured after colouring the neighbours of x_1 and y_1, so w_j is neither adjacent to x_1 nor to y_1. This means that $w_j x_j x_1 y_1$ is an induced P_4 if and only if x_j is not adjacent to y_1. Recall that w_j and every vertex of D_i, so including x_1 and x_j, has at most d neighbours in D. As $|D| \geq 3d+1$, this means that D contains a vertex z that is not adjacent to any of w_j, x_j, x_1. As D is a clique, z is adjacent to y_1. Hence, x_j must be adjacent to y_1, as otherwise $w_j x_j x_1 y_1 z$ is an induced P_5; see also Figure 2. The latter is not possible as G is P_5-free. We now find that y_1 is adjacent to $x_1, \ldots, x_p$, which all belong to D_i. As we assumed that $p \geq d+1$ and every vertex of D, including y_1, is adjacent to at most d vertices of D_i, this yields a contradiction. We conclude that $p \leq d$.

As $p \leq d$, the total number of side branches is $O(n^{2d^2})$. Each side branch yields a red-blue colouring of G. We check in polynomial time if it is a red-blue d-colouring of G. If so, then we stop and return it; else we can safely discard the side branch, and eventually the associated main branch, due to Claim 11.1.

The correctness of our algorithm follows from its description. We now analyze its running time. We started the algorithm with searching for a set D. If D has size at most $3d$, then we applied Lemma 8, which takes polynomial time. Otherwise we argue as follows. The total number of main branches is $O(n^d)$. For each main branch we have either $O(2^{2d+1}n^{d(2d+1)})$ side branches (Case 1) or $O(n^{2d^2})$ side branches (Case 2). Hence, the total number of branches is $O(n^d 2^{2d+1} n^{d(2d+1)})$. As d is fixed, this number is polynomial. As processing each branch takes polynomial time (including the construction of the D_i-sets and L_i-graphs), we conclude that our algorithm runs in polynomial time. This completes the proof. □

We omit the proofs of our final two polynomial-time results.

Theorem 12. *For every $d \geq 2$, d-Cut is polynomial-time solvable for (P_3+P_4)-free graphs.*

Theorem 13. *For every graph H and every $d \geq 2$, if d-Cut is polynomial-time solvable for H-free graphs, then it is so for $(H + P_1)$-free graphs.*

4 NP-Completeness Results

In this section we show our NP-completeness results for d-Cut for $d \geq 2$ (some of these results only hold for $d \geq 3$). As d-Cut is readily seen to be in NP for each $d \geq 1$, we only show NP-hardness in our proofs. The proof of our first result, which we omit, generalizes the arguments of the corresponding proof for Matching Cut [7].

Theorem 14. *For $d = 2$, d-Cut is NP-complete for $K_{1,4}$-free graphs.*

For $d \geq 3$, we prove a stronger result than Theorem 14. An edge colouring of a graph $G = (V, E)$ with colours red and blue is called a *red-blue edge colouring* of G, which is a *red-blue edge d-colouring* of G if every edge of G is adjacent to at most d edges of the other colour and both colours are used at least once. Now, G has a red-blue edge d-colouring if and only if $L(G)$ has a red-blue d-colouring. A set $S \subseteq V$ is *monochromatic* if all edges of $G[S]$ are coloured alike.

Theorem 15. *For $d \geq 3$, d-Cut is NP-complete for line graphs.*

Proof. We reduce from Not-All-Equal Satisfiability with positive literals only and in which each clause contains exactly three literals. This problem is NP-complete by Theorem 9. Let $(X, \mathcal{C})$ be such an instance, where $X = \{x_1, x_2, \ldots, x_n\}$ and $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$.

We construct, in polynomial time, a graph G; see also Figure 3:

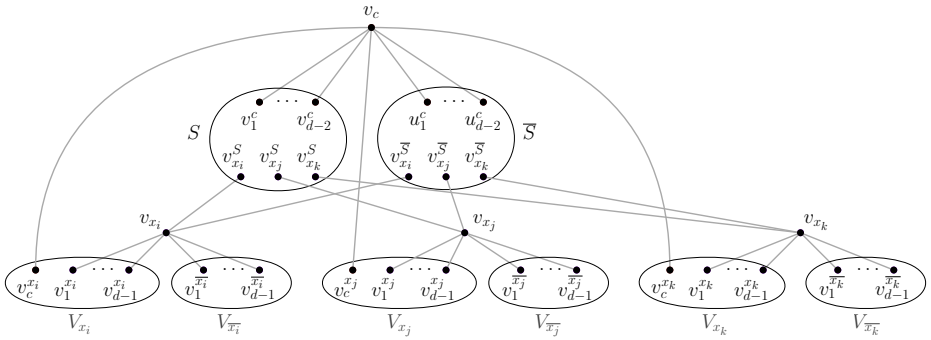

Fig. 3: An example of vertices in the reduction related to clause $C = \{x_i, x_j, x_k\}$.

- Build a clique $S = \{v^S_{x_1}, \ldots, v^S_{x_n}\} \cup \{v^{c_1}_1, \ldots, v^{c_1}_{d-2}\} \cup \cdots \cup \{v^{c_m}_1, \ldots, v^{c_m}_{d-2}\}$.
- Build a clique $\overline{S} = \{v^{\overline{S}}_{x_1}, \ldots, v^{\overline{S}}_{x_n}\} \cup \{u^{c_1}_1, \ldots, u^{c_1}_{d-2}\} \cup \cdots \cup \{u^{c_m}_1, \ldots, u^{c_m}_{d-2}\}$.
- For every $x \in X$, add cliques $V_x = \{v^x_1, \ldots, v^x_{d-1}\}$ and $V_{\overline{x}} = \{v^{\overline{x}}_1, \ldots, v^{\overline{x}}_{d-1}\}$.
- For every $x \in X$, add a vertex v_x with edges $v_x v^S_x$, $v_x v^{\overline{S}}_x$, $v_x v^x_1, \ldots, v_x v^x_{d-1}$, $v_x v^{\overline{x}}_1, \ldots, v_x v^{\overline{x}}_{d-1}$.
- For every $C \in \mathcal{C}$, add a clause vertex v_c with edges $v_c v^c_1, \ldots, v_c v^c_{d-2}$, $v_c u^c_1, \ldots, v_c u^c_{d-2}$, and if $C = \{x_i, x_j, x_k\}$, also add a vertex $v^{x_i}_c$ to V_{x_i}, a vertex $v^{x_j}_c$ to V_{x_j} and a vertex $v^{x_k}_c$ to V_{x_k}, and add the edges $v_c v^{x_i}_c$, $v_c v^{x_j}_c$, $v_c v^{x_k}_c$.
- Add, if needed, some auxiliary vertices to $S, \overline{S}, V_{x_1}, \ldots, V_{x_n}, V_{\overline{x_1}}, \ldots, V_{\overline{x_n}}$, such that in the end all these sets are cliques of size at least $2d + 2$.

We claim that $(X, \mathcal{C})$ has a satisfying not-all-equal truth assignment if and only if the line graph $L(G)$ has a d-cut. Recall that, by Observation 7, $L(G)$ has a d-cut if and only if $L(G)$ has a red-blue d-colouring. Furthermore, $L(G)$ has a red-blue d-colouring if and only if G has a red-blue edge d-colouring. Hence, we will show that $(X, \mathcal{C})$ has a satisfying not-all-equal truth assignment if and only if G has a red-blue edge d-colouring.

First suppose $(X, \mathcal{C})$ has a satisfying not all-equal truth assignment. We colour all edges in S red and in $\overline{S}$ blue. For every $x \in X$ set to true, we colour the edges in V_x red and those in $V_{\overline{x}}$ blue. For every $x \in X$ set to false, we colour the edges in V_x blue and those in $V_{\overline{x}}$ red. Consider an edge uv, with $v \in \{v_x, v_c \mid x \in X, c \in C\}$. Then u is contained in a clique $D \in \{S, \overline{S}, V_{x_1}, V_{\overline{x_1}}, \ldots, V_{x_n}, V_{\overline{x_n}}\}$. Colour uv with the same colour as the edges of D.

Now, let $D \in \{S, \overline{S}, V_{x_1}, V_{\overline{x_1}}, \ldots, V_{x_n}, V_{\overline{x_n}}\}$. Every $uu' \in E(D)$ is adjacent to only edges of the same colour. For $u \in V(D)$ and $v \in \{v_x, v_c \mid x \in X, c \in C\}$, the edge uv has the same colour as all edges in D. Since S and $\overline{S}$ have different colours and V_x and $V_{\overline{x}}$ have different colours for every $x \in X$, uv has at most d adjacent edges of each colour. Hence, we obtained a red-blue edge d-colouring of G.

Now suppose that G has a red-blue edge d-colouring. We prove a series of claims:

Claim 15.1 *Every clique $D \in \{S, \overline{S}, V_{x_1}, V_{\overline{x_1}}, \ldots, V_{x_n}, V_{\overline{x_n}}\}$ is monochromatic.*

Proof of the Claim. First assume $G[D]$ has a red edge uv and a blue edge uw. As $|D| \geq 2d+2$, we know that u is incident to at least $2d+1$ edges. Hence, we may assume without loss of generality that u is incident to at least $d+1$ red edges. However, now the blue edge uw is adjacent to $d+1$ red edges, a contradiction. As every $u \in D$ is incident to only edges of the same colour and D is a clique, it follows that D is monochromatic. ◁

By Claim 15.1, we can speak about the *colour* (either red or blue) of a clique D if D belongs to $\{S, \overline{S}, V_{x_1}, V_{\overline{x_1}}, \ldots, V_{x_n}, V_{\overline{x_n}}\}$.

Claim 15.2 *For each $x \in X$ and $c \in C$, each edge from v_x or v_c to a vertex in a clique $D \in \{S, \overline{S}, V_{x_1}, V_{\overline{x_1}}, \ldots, V_{x_n}, V_{\overline{x_n}}\}$ has the same colour as D.*

Proof of the Claim. This follows directly from the fact that $|D| \geq 2d+1$. ◁

Claim 15.3 *The cliques S and $\overline{S}$ have different colours if and only if for every variable $x \in X$, it holds that V_x and $V_{\overline{x}}$ have different colours.*

Proof of the Claim. First suppose S and $\overline{S}$ have different colours, say S is red and $\overline{S}$ is blue. For a contradiction, assume there exists a variable $x \in X$, such that V_x and $V_{\overline{x}}$ have the same colour, say blue. By Claim 15.2, we have that the $2d-2$ edges between v_x and $V_x \cup V_{\overline{x}}$ and the edge $v_x v_x^{\overline{S}}$ are all blue, while $v_x v_x^S$ is red. Hence, the red edge $v_x v_x^S$ is adjacent to at least $2d-1 \geq d+1$ blue edges, a contradiction.

Now suppose that for all $x \in X$, V_x and $V_{\overline{x}}$ have different colours, say V_x is red and $V_{\overline{x}}$ is blue. For a contradiction, assume that S and $\overline{S}$ have the same colour, say blue. Let $x \in X$. By Claim 15.2, we have that the edges between v_x and V_x are red, while all other edges incident to v_x are blue. Now every (red) edge between v_x and V_x is incident to $d+1$ blue edges, a contradiction. ◁

Claim 15.4 *The cliques S and $\overline{S}$ have different colours.*

Proof of the Claim. For a contradiction, assume S and $\overline{S}$ have the same colour, say blue. By Claim 15.2, we have that $v_x v_x^S$ is blue. By Claim 15.3, we find that for every $x \in X$, V_x and $V_{\overline{x}}$ have the same colour. If V_x and $V_{\overline{x}}$ are both red, then the $2d-2$ edges between v_x and $V_x \cup V_{\overline{x}}$ are red due to Claim 15.2. Consequently, the blue edge $v_x v_x^S$ is adjacent to $2d-2 \geq d+1$ red edges. This is not possible. Hence, V_x and $V_{\overline{x}}$ are blue, and by Claim 15.2, all edges between v_x and $V_x \cup V_{\overline{x}}$ are blue as well. This means that every edge of G is blue, a contradiction. ◁

Claim 15.5 *For every clause $C = \{x_i, x_j, x_k\}$ in $\mathcal{C}$, the cliques V_{x_i}, V_{x_j} and V_{x_k} do not all have the same colour.*

Proof of the Claim. For a contradiction, assume V_{x_i}, V_{x_j} and V_{x_k} have the same colour, say blue. By Claim 15.4, S and $\overline{S}$ are coloured differently, say S is red and $\overline{S}$ is blue. By Claim 15.2, we have that the three edges $v_c v_c^{x_i}$, $v_c v_c^{x_j}$ and $v_c v_c^{x_k}$ are all blue, just like the $d-2$ edges between v_c and $\overline{S}$, while every edge between v_c and S is red. Consider such a red edge e. We find that e is adjacent to $d+1$ blue edges, a contradiction. ◁

For each variable x, if the clique V_x is coloured red, then set x to true, and else to false. By Claim 15.5, this yields a satisfying not-all-equal truth assignment. □

We now show a result for $d = 2$ and strengthen it for $d \geq 3$. The gadgets in the two proofs, which we omit, are similar to each other. They are not $2P_4$-free, P_6-free or P_7-free.

Theorem 16. *For $d = 2$, d-Cut is NP-complete for $3P_3$-free graphs of radius 2 and diameter 3.*

Theorem 17. *For $d \geq 3$, d-Cut is NP-complete for $3P_2$-free graphs of radius 2 and diameter 3.*

Our final theorem is a straightforward generalization of the case $d = 1$ from [24], and we omit its proof.

Theorem 18. *For $d \geq 2$ and $i \geq 1$, d-Cut is NP-complete for $(H_1^*, \ldots, H_i^*)$-free graphs.*

5 Conclusions

We considered the natural generalization of Matching Cut to d-Cut [13] and proved dichotomies for graphs of bounded diameter and graphs of bounded radius. We also started a systematic study on the complexity of d-Cut for H-free graphs. For $d = 2$, our results for H-free graphs still left an infinite number of non-equivalent open cases, just like for $d = 1$. However, for $d = 3$ we were able to obtain an almost-complete complexity classification of d-Cut for H-free graphs, with only three non-equivalent open cases left. We finish our paper with some open problems on H-free graphs resulting from our systematic study.

We recall that 1-Cut is polynomial-time solvable for claw-free graphs [4], while we showed that d-Cut is NP-complete even for line graphs if $d \geq 3$. What is the

complexity of 2-Cut for line graphs and for claw-free graphs? We also ask whether 2-Cut is polynomial-time solvable for $(H + P_2)$-free graphs if it is for H-free graphs. A positive answer would reduce the number of non-equivalent open cases to a finite number for $d = 2$. In particular, we ask for the complexity of 2-Cut for $3P_2$-free graphs. Finally, we recall the only three non-equivalent open cases $H = 2P_4$, $H = P_6$, $H = P_7$ for d-Cut on H-free graphs for $d \geq 3$. The first and last case are also open for $d = 1$; for $d = 2$ all three cases are open.

Acknowledgments. We thank Carl Feghali and Édouard Bonnet for fruitful discussions.

References

1. Araújo, J., Cohen, N., Giroire, F., Havet, F.: Good edge-labelling of graphs. Discrete Applied Mathematics **160**, 2502–2513 (2012)
2. Aravind, N.R., Saxena, R.: An FPT algorithm for Matching Cut and d-Cut. Proc. IWOCA 2021, LNCS **12757**, 531–543 (2021)
3. Bonnet, E., Chakraborty, D., Duron, J.: Cutting Barnette graphs perfectly is hard. Proc. WG 2023, LNCS **14093**, 116–129 (2023)
4. Bonsma, P.S.: The complexity of the Matching-Cut problem for planar graphs and other graph classes. Journal of Graph Theory **62**, 109–126 (2009)
5. Borowiecki, M., Jesse-Józefczyk, K.: Matching cutsets in graphs of diameter 2. Theoretical Computer Science **407**, 574–582 (2008)
6. Bouquet, V., Picouleau, C.: The complexity of the Perfect Matching-Cut problem. CoRR **abs/2011.03318** (2020)
7. Chvátal, V.: Recognizing decomposable graphs. Journal of Graph Theory **8**, 51–53 (1984)
8. Farley, A.M., Proskurowski, A.: Networks immune to isolated line failures. Networks **12**, 393–403 (1982)
9. Feghali, C.: A note on Matching-Cut in P_t-free graphs. Information Processing Letters **179**, 106294 (2023)
10. Feghali, C., Lucke, F., Paulusma, D., Ries, B.: Matching cuts in graphs of high girth and H-free graphs. Proc. ISAAC 2023, LIPIcs **283**, 28:1–28:16 (2023)
11. Golovach, P.A., Paulusma, D., Song, J.: Computing vertex-surjective homomorphisms to partially reflexive trees. Theoretical Computer Science **457**, 86–100 (2012)
12. Golovach, P.A., Paulusma, D., Song, J.: Closing complexity gaps for coloring problems on H-free graphs. Information and Computation **237**, 204–214 (2014)
13. Gomes, G., Sau, I.: Finding cuts of bounded degree: complexity, FPT and exact algorithms, and kernelization. Algorithmica **83**, 1677–1706 (2021)
14. Graham, R.L.: On primitive graphs and optimal vertex assignments. Annals of the New York Academy of Sciences **175**, 170–186 (1970)
15. Heggernes, P., Telle, J.A.: Partitioning graphs into generalized dominating sets. Nordic Journal of Computing **5**, 128–142 (1998)
16. van't Hof, P., Paulusma, D.: A new characterization of P_6-free graphs. Discrete Applied Mathematics **158**, 731–740 (2010)
17. Le, H.O., Le, V.B.: A complexity dichotomy for Matching Cut in (bipartite) graphs of fixed diameter. Theoretical Computer Science **770**, 69–78 (2019)
18. Le, H., Le, V.B.: Complexity results for matching cut problems in graphs without long induced paths. Proc. WG 2023, LNCS **14093**, 417–431 (2023)
19. Le, V.B., Lucke, F., Paulusma, D., Ries, B.: Maximizing matching cuts. CoRR **abs/2312.12960** (2023)
20. Le, V.B., Telle, J.A.: The Perfect Matching Cut problem revisited. Theoretical Computer Science **931**, 117–130 (2022)
21. Liu, J., Zhou, H.: Dominating subgraphs in graphs with some forbidden structures. Discrete Mathematics **135**, 163–168 (1994)
22. Lucke, F., Paulusma, D., Ries, B.: On the complexity of Matching Cut for graphs of bounded radius and H-free graphs. Theoretical Computer Science **936** (2022)
23. Lucke, F., Paulusma, D., Ries, B.: Dichotomies for Maximum Matching Cut: H-Freeness, Bounded Diameter, Bounded Radius. Proc. MFCS 2023, LIPIcs **272**, 64:1–64:15 (2023)

24. Lucke, F., Paulusma, D., Ries, B.: Finding matching cuts in H-free graphs. Algorithmica **85**, 3290–3322 (2023)
25. Moshi, A.M.: Matching cutsets in graphs. Journal of Graph Theory **13**, 527–536 (1989)
26. Patrignani, M., Pizzonia, M.: The complexity of the Matching-Cut problem. Proc. WG 2001, LNCS **2204**, 284–295 (2001)
27. Schaefer, T.J.: The complexity of satisfiability problems. Proc. STOC 1978 pp. 216–226 (1978)