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[2,3], Daniël Paulusma[4][0000−0001−5945−9287], and Siani Smith[5][0000−0003−0797−0512]

[1] Department of Informatics, University of Fribourg, Fribourg, Switzerland
felicia.lucke@unifr.ch
[2] Faculty of Computer Science, University of Vienna, Vienna, Austria
[3] UniVie Doctoral School Computer Science DoCS
ali.momeni@univie.ac.at
[4] Department of Computer Science, Durham University, Durham, UK
daniel.paulusma@durham.ac.uk
[5] 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 almost-complete dichotomies for H-free graphs.

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 [1,9,12,27]. The Matching Cut problem is to decide if a connected graph has a matching cut. This problem was

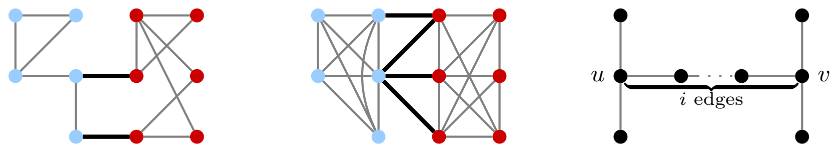

Fig. 1: Left: a graph with a matching cut (1-cut). Middle: a graph with a 3-cut but no d-cut for $d \leq 2$. Right: the graph H_i^*.

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 [3,11,18,20,25] 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 [19,24] 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. [11] 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 [23], while being NP-complete for graphs of diameter 3 [17], and thus radius at most 3. Hence, we obtain:

Theorem 1 ([17,23]). *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 $g \geq 3$ [11] 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 [10] 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 [25] 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 [26] and for $(H_1^*, \ldots, H_i^*)$-free graphs for every $i \geq 1$ [11].

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 [25]. 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 [25].

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 [11,24]):

Theorem 2 ([4,7,11,18,23,25,26]). *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 Theorems 1–4 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 $d = 1$ [17,23]. 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 for all $d \geq 2$, d-Cut is NP-complete for $3P_2$-free graphs, and also 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, and thus $K_{1,3}$-free graphs, for $d \geq 3$. The NP-completeness for graphs of large girth from [11] 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* $H \supseteq_i K_{1,4}$*,* $3P_2$*,* 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* $i \geq 1$*.*

Theorems 5 and 6 show that both for $d = 2$ and $d \geq 3$, there are only a finite number of non-equivalent open cases left; for $d = 3$, there are even only three non-equivalent open cases, 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

Throughout the paper, we only consider finite, undirected graphs without multiple edges and self-loops. We first define some general graph terminology.

Let $G = (V, E)$ be a graph. The *line graph* $L(G)$ of G 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 *(open) neighbourhood* of S is the set $N(S) = \bigcup_{u \in S} N(u) \setminus S$, and the *closed neighbourhood* $N[S] = N(S) \cup S$. We let $G[S]$ denote the subgraph of G *induced* by S, which is obtained from G by deleting the vertices not in S. If no two vertices in S are adjacent, then S is an *independent set* of G. If every two vertices in S are adjacent, then S is a *clique* of G. 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.

We now generalize some colouring terminology that was used in the context of matching cuts (see, e.g., [10,23]). A *red-blue colouring* of a graph G assigns every vertex of G either the colour red or blue. 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.

We make the following observation.

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., [23]).

Lemma 8. *For $d, g \geq 1$, it is possible to find in $O(2^g n^{d+2})$-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 $O(2^g n^d)$ and checking if a red-blue colouring is a red-blue d-colouring takes $O(n^2)$ time, our algorithm has total running time $O(2^g n^{d+2})$. □

Let $d \geq 1$. Let $G = (V, E)$ be a connected graph and $S, T \subseteq V$ be two disjoint sets. A *red-blue* (S,T)*-d-colouring* of G is a red-blue d-colouring of G that colours all the vertices of S red and all the vertices of T blue. We call (S,T) a *precoloured pair* of (G,d) which is *colour-processed* if every vertex of $V \setminus (S \cup T)$ is adjacent to at most d vertices of S and to at most d vertices of T.

By the next lemma, we may assume without loss of generality that a precoloured pair (S,T) is always colour-processed.

Lemma 9. *Let G be a connected graph with a precoloured pair (S,T). It is possible, in polynomial time, to either colour-process (S,T) or to find that G has no red-blue (S,T)-d-colouring.*

Proof. We apply the following rules on G. Let $Z = V \setminus (S \cup T)$. If v is adjacent to $d+1$ vertices in S, then move v from Z to S. If v is adjacent to $d+1$ vertices in T, then move v from Z to T. Return `no` if a vertex $v \in Z$ at some point becomes adjacent to $d+1$ vertices in S as well as to $d+1$ vertices in T. We apply these three rules exhaustively. It is readily seen that each of these rules is safe to use, and moreover, can be verified and applied in polynomial time. After each application of a rule, we either stop or have decreased the size of Z by at least one vertex. Hence, the procedure is correct and takes polynomial time. □

In our NP-hardness proofs we reduce from Not-All-Equal Satisfiability, which as we discuss below remains NP-complete under certain input restrictions. Let $X = \{x_1, x_2, ..., x_n\}$ be a set of logical 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 first part of the next theorem is well-known [28] and the second part is proven by Darmann and Döcker [8].

Theorem 10 ([8,28]). Not-All-Equal Satisfiability *is* NP*-complete even for*

– *instances, in which each clause consists of three distinct literals that are all positive;*
– *instances, in which each variable occurs as a positive literal in exactly two clauses and as a negative literal in exactly two other clauses, and in which each clause consists of three distinct literals that are either all positive or all negative.*

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).

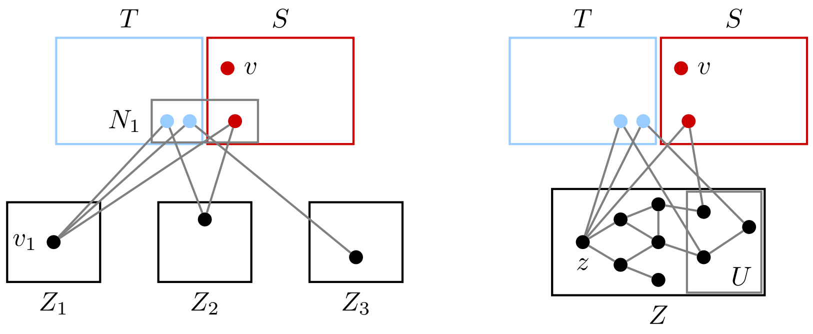

Fig. 2: Cases 1 (left) and 2 (right) in the proof of Theorem 11.

Theorem 11. *For $d \geq 2$, d-Cut is polynomial-time solvable for graphs of diameter at most* 2.

Proof. Let $G = (V, E)$ be a graph of diameter at most 2. Note that this implies that G is connected. Let $v \in V$. Without loss of generality we may colour v red. Since v has at most d blue neighbours in any red-blue d-colouring, we can branch over all $O(n^d)$ options to colour the neighbourhood of v. We consider each option separately.

Let R and B be the red and the blue set, respectively. Let $S' = \{v\} \cup \{u \mid u \in N(v), u \in R\}$ and let $T' = \{u \mid u \in N(v), u \in B\}$. We use Lemma 9 to colour-process (S', T') in polynomial time, resulting in a pair (S, T). Let $Z = V \setminus (S \cup T)$ be the set of uncoloured vertices. As (S, T) is colour-processed, every vertex in Z has at most d red and d blue neighbours and thus in total at most $2d$ coloured neighbours. We distinguish between the following two cases:

Case 1. $G[Z]$ consists of at least two connected components.
Let $Z_1, \ldots, Z_r$ be the connected components of $G[Z]$. Let $v_1 \in V(Z_1)$ and $v_2 \in V(Z_2)$. Since G has diameter 2, we know that v_1 has a common neighbour with

every vertex in $Z_2, \dots, Z_r$. Let N_1 be the set of common neighbours of v_1 and $Z_2, \dots, Z_r$. Note that $N_1 \subseteq S \cup T$. Thus, $|N_1| \leq 2d$, since v_1 has at most $2d$ coloured neighbours. Using the same arguments, we can find a set N_2, with $|N_2| \leq 2d$ containing the common neighbours of v_2 and Z_1. In a red-blue d-colouring, every vertex in $N_1 \cup N_2$ has at most d neighbours of the other colour. Thus, we can try all $O(n^{4d^2})$ options to colour $N_{G[Z]}(N_1 \cup N_2)$. If in the resulting colouring any vertex has more than d neighbours of the other colour we discard it, otherwise we found a red-blue d-colouring and thus a d-cut.

Case 2. $G[Z]$ is connected.
We first assume that $\mathsf{radius}(G[Z]) > 2$. We let $z \in Z$, and we let $U = \{u \in Z : \mathsf{dist}_{G[Z]}(z, u) > 2\}$. Note that $U \neq \emptyset$. As G has diameter 2, we find that z has a common neighbour with every vertex in U. These common neighbours are not contained in Z and thus they are in $S \cup T$, the set of coloured vertices. Vertex z has at most $2d$ coloured neighbours. Each of them has at most d neighbours of the other colour. So we can branch over all $O(n^{2d^2})$ options to colour U. If any colouring of U leads to a vertex having more than d neighbours of the other colour, we discard the branch. For each remaining colouring, we know that the graph $G[Z \setminus U]$ which remains uncoloured has radius at most 2. Let $Z = Z \setminus U$.

So we may assume (possibly after some branching) that $\mathsf{radius}(G[Z]) \leq 2$. Let $z \in Z$ such that $\mathsf{dist}_{G[Z]}(z, u) \leq 2$ for all $u \in Z$. We branch over all $O(n^d)$ colourings of $\{z\} \cup N_{G[Z]}(z)$. Since $\mathsf{radius}(G[Z]) \leq 2$, we get that $\{z\} \cup N_{G[Z]}(z)$ is a dominating set of $G[Z]$. Hence every uncoloured vertex got a new coloured neighbour. We colour-process the pair of red and blue sets, which takes polynomial time by Lemma 9. This results in a new uncoloured set of vertices Z'. If there is a vertex with $d+1$ neighbours of the other colour or an uncoloured vertex with more than d neighbours of each colour we discard the branch. Otherwise we can proceed by choosing either Case 1 or Case 2.

If at some point we apply Case 1, we are done. Every time we apply Case 2, all vertices that remain uncoloured get a new coloured neighbour. Hence, we can apply Case 2 at most $2d$ times, since afterwards every uncoloured vertex has more than $2d$ coloured neighbours and thus, they are coloured when we colour-process the sets of red and blue vertices.

The correctness of our algorithm follows from its description. We now discuss its running time. We always have $O(n^d)$ branches in the beginning. Case 1 give $O(n^{4d^2})$ branches, while Case 2 only gives $O(n^{2d^2+d})$ branches. Since after one application of Case 1 we are done, in the worst case we apply Case 2 $2d-1$ times and then apply Case 1. This leads to

$$O\left(n^d * \left(n^{2d^2+d}\right)^{2d-1} * n^{4d^2}\right) = O\left(n^{4d^2(d+1)}\right)$$

branches, a polynomial number, to consider in the worst case. Since by Lemma 9 the colour-processing can be done in polynomial time, the algorithm runs in polynomial time. □

Theorem 12. *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 component L_1 of $G - D$ that will contain a red vertex.[6] 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 3.

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* 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

[6] For $d = 1$, up to now, the same approach is used for P_5-free graphs [10]. 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.

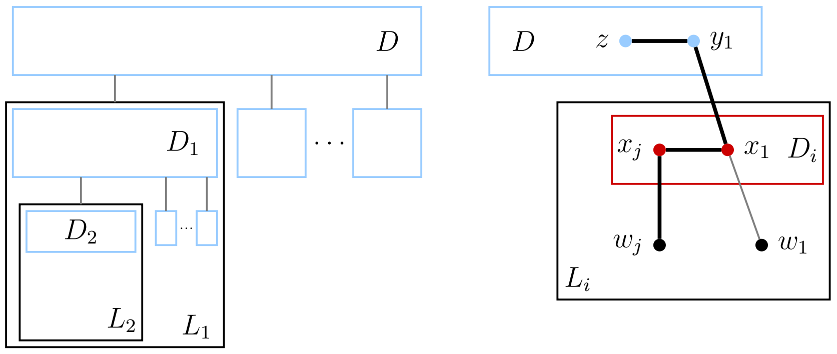

Fig. 3: The graph G in the blue phase (left) and in the red phase: Case 2 (right).

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 12.1 *The graph G has a red-blue d-colouring that colours every vertex of D without loss of generality 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 12.1 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 12.1 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 12.1.

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 12.1.

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 12.1.

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 and a neighbour y_2 of w_2 in D, and we branch again by colouring the uncoloured neighbours of x_2 and y_2, and so on. We repeat this process until there are no more uncoloured vertices. This gives us $2p$ distinct vertices $w_1, \ldots, w_p$, $x_1, \ldots, x_p$, together with vertices $y_1, \ldots, y_p$ (with possibly $y_a = y_b$ for some $1 \leq a < b \leq p$) for some integer $p \geq 1$. 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 $w_jx_jx_1y_1$. 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 $w_jx_jx_1y_1$ 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 $w_jx_jx_1y_1z$ is an induced P_5; see also Figure 3. 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 12.1.

The correctness of our algorithm follows from its description. With regard to its running time, 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 is a polynomial number. 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. □

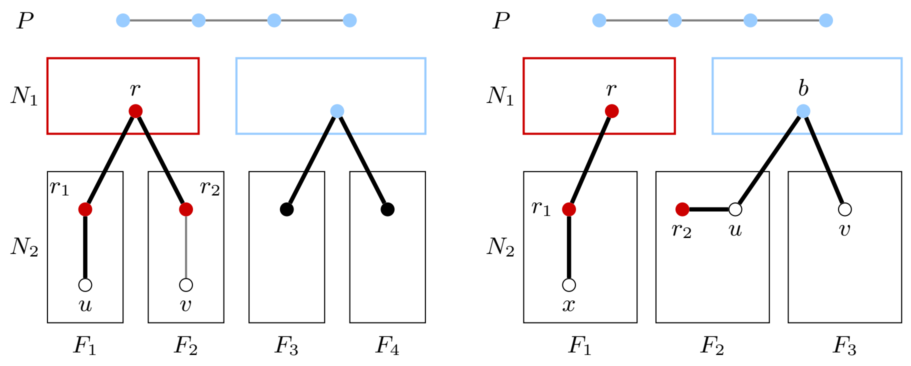

Fig. 4: Illustration of the proof of Theorem 13. The white vertices indicate uncoloured vertices, while for the black vertices we do not know if they are uncoloured or which colour they have.

We now consider the case $H = P_3 + P_4$.

Theorem 13. *For every $d \geq 2$, d-Cut is polynomial-time solvable for (P_3+P_4)-free graphs.*

Proof. Let G be a connected P_4+P_3-free graph. We may assume that G contains an induced P_4, else we apply Theorem 12.

We divide the proof into two cases. First, we decide whether there exists a red-blue d-colouring of G in which some induced P_4 is monochromatic. We then search for a red-blue d-colouring in which every induced P_4 is bichromatic.

Case 1. There exists a monochromatic P_4 in some red-blue d-colouring of G.

We branch over all induced P_4s of G. Note that there are at most $O(n^4)$ induced P_4s in G. Let P be the current induced P_4 of G which we assume is monochromatic. Without loss of generality, every vertex of P is blue. Let $N_1 = N(P)$ and $N_2 = V(G) \setminus N[P]$. Since G is $P_4 + P_3$-free we get that $G[N_2]$ is P_3-free and thus every connected component of $G[N_2]$ is a clique. We branch over all $O(n^{4d})$ possible colourings of N_1. If $V(P) \cup N_1$ is monochromatic, that is N_1 is blue, then it is enough to decide whether there is a connected component F of $G[N_2]$ such that $G[V(P) \cup N_1 \cup V(F)]$ has a red-blue d-colouring. All other components can safely be coloured blue. Since F is a clique, F is monochromatic if $|V(F)| \geq 2d+1$ and has constant size otherwise. In both cases only a constant number of possible colourings exist for F and we can try each of them to decide if it gives a red-blue d-colouring of $G[V(P) \cup N_1 \cup V(F)]$. If we find a red-blue d-colouring we are done, otherwise we discard the branch and continue with the case where $V(P) \cup N_1$ is not monochromatic.

Let S be the set of red vertices in N_1. Since we handled the case where all vertices in N_1 are blue, we may assume that S is not empty. We then branch over all $O(n^{4d^2})$ possible colourings of $N(S)$. Note that every vertex which remains uncoloured belongs to N_2 and that all its neighbours in N_1 are blue. Recall that each connected component of $G[N_2]$ is a clique. Therefore each connected component either has size at most $2d$ or is monochromatic. This implies that there are a constant number of possible red-blue colourings for each component. We colour-process the current red-blue colouring.

Consider a connected component of $G[N_2]$ which contains an uncoloured vertex. We call this component F_1. If F_1 contains no red vertex with a red neighbour in N_1 then all vertices in F_1 have only blue neighbours in N_1, since the neighbourhood of S is coloured. Thus, we may recolour the whole connected component blue.

We now consider the case that there exists a red vertex $r \in N_1$ which has red neighbours in two different connected components of N_2 which contain uncoloured vertices. Let F_1 and F_2 be these connected components and $r_1 \in V(F_1)$ the red neighbour of r in F_1 and $u \in V(F_1)$ an uncoloured vertex in F_1. Similarly let $r_2 \in V(F_2)$ be a red neighbour of r in F_2 and $v \in V(F_2)$ an uncoloured vertex in F_2. Note that neither u nor v is a neighbour of r since $r \in S$ and $N(S)$ is already coloured. Then $Q = u - r_1 - r - r_2$ is an induced P_4. By P_4+P_3-freeness, any induced P_3 in G contains a vertex which is adjacent to at least one vertex of Q. In particular, this implies hat every blue vertex in N_1 with uncoloured neighbours in more than one connected component of $G[N_2]$ which is neither F_1, nor F_2 has a neighbour on Q. See Figure 4 (left). Let T be the set of these vertices, that is the set of blue vertices in N_1 with uncoloured neighbours in at least two connected components, neither of which is F_1 or F_2. Since the vertices of Q are either red or uncoloured, they may have at most d blue neighbours and thus $|T| \leq 4d$.

We branch over all $O(n^{4d^2})$ colourings of $N(T) \cup V(F_1) \cup V(F_2)$. Let $F_3, \dots, F_k$ be the connected components of $G[N_2]$ which contain an uncoloured vertex. Let $x \in V(F_i)$, for $i \in \{3, \dots, k\}$, be a vertex that remains uncoloured. The neighbours of x in N_1 have uncoloured neighbours only in F_i.

Therefore, it is enough to decide whether we can extend the current red-blue colouring to a red-blue d-colouring of $G[V(F_i) \cup N_1]$ for each connected component F_i of $G[N_2]$ with $i \geq 3$. To see that these colourings are independent, recall that any remaining vertex of N_1 with an uncoloured neighbour has uncoloured neighbours in only one component F_i. That is, we branch over each of the constant number of possible red-blue colourings of the uncoloured vertices of F_i and decide whether this leads to a red-blue colouring of $G[V(F_i) \cup N_1]$. If this is the case for every component F_i of $G[N_2]$ then G is a yes-instance. Otherwise we discard the branch.

Next we consider the case where every red vertex r in N_1 has red neighbours in at most one connected component of $G[N_2]$ containing uncoloured vertices. Let F_1 be a connected component of $G[N_2]$ containing an uncoloured vertex x together with a neighbour r_1 of r. Note that x and r are non-adjacent since $r \in S$ and the neighbourhood of S is coloured. Then $J = x - r_1 - r$ is an induced P_3 in G and at most $3d$ blue vertices in G have a neighbour on this path. Let Γ be the set of such vertices and consider each of $O(n^{3d^2})$ colourings of $N[\Gamma]$.

Any remaining blue vertex $b \in N_1$ with an uncoloured neighbour has no neighbours belonging to Γ. If b has uncoloured neighbours u and v in two different components F_2 and F_3 and a red non-neighbour r_2 in one of them, say F_2, we obtain an induced $P_4 + P_3$ consisting of these four vertices together with J, see Figure 4 (right). Therefore, b either has neighbours in at most one connected component of $G[N_2]$ different from F_1 or is adjacent to every red or uncoloured vertex in every connected component different from F_1 in which it has a neighbour.

Note that b has at most $d-1$ red neighbours, else its remaining neighbours would be coloured blue. Since every connected component of $G[N_2]$ containing an uncoloured vertex contains a red vertex, b has uncoloured neighbours in at most $d-1$ components.

Let T be the set of blue vertices in N_1 with uncoloured neighbours. For $t \in T$, let M_t be the set of cliques in N_2 in which t has uncoloured neighbours. For any two vertices t_i and $t_j \in T$, M_{t_i} and M_{t_j} are either disjoint or one is contained in the other. Assume otherwise. Then there exist blue vertices $t_i, t_j \in T$ and distinct integers $a, b, c \geq 2$ such that:

- $x \in F_a$ is adjacent to both t_i and t_j.

- $y \in F_b$ is adjacent to t_i but not t_j.
- $z \in F_c$ is adjacent to t_j but not t_i.

Then t_i and t_j are adjacent by $P_4 + P_3$-freeness. But then $y - b_i - b_j - z$ is an induced P_4 with no neighbours in J a contradiction.

Hence, we obtain a partition $D_1 \dots D_l$ of the cliques of N_2 containing uncoloured vertices such that, for each part D:

- No vertex of T with a neighbour in D_i has neighbours in any other part.
- Each part contains at most $d-1$ cliques, each of size at most $2d$.

Therefore, for each part D in turn, we test all of $2^{2d(d-1)}$ possible colourings of the uncoloured vertices in $G[V(D_i)]$ to decide whether there exists an extension of our red-blue colouring of N_1 to a red-blue d-colouring of $G[V(D) \cup N_1]$. If such a colouring exists, we have found a d-cut otherwise we discard the branch.

Case 2. In any red-blue d-colouring of G, every induced P_4 is bichromatic.

Note that we may assume that both the set B of blue vertices in a red-blue d-colouring of G and the set R of red vertices induce connected subgraphs of G. Otherwise, assume that $G[B]$ is disconnected. Then we may recolour all but one connected component of $G[B]$ red. If then $G[R]$ is disconnected, each of its connected components contains a vertex with a neighbour in B, else G is disconnected. We obtain a red-blue d-colouring where both $G[R]$ and $G[B]$ are connected by recolouring all but one connected component of $G[R]$ blue. Therefore in this case $G[B]$ and $G[R]$ have diameter at most 2, else we have a monochromatic P_4.

Let P be an induced P_4 in G. We can find P in time $O(n^4)$ and we branch over all 16 colourings of P. Further, we branch over all $O(n^{4d})$ colourings of its neighbourhood N_1. Let $N_2 = V(G) \setminus N[P]$. If $N_2 = \emptyset$ we are done. Otherwise, since G is $P_4 + P_3$-free, every connected component of $G[N_2]$ is a clique. Let $F_1, \dots, F_k$, for some integer $k \geq 1$, be the connected components of $G[N_2]$. We colour-process the current colouring.

If all components $F_1, \dots, F_k$ are coloured we are done, otherwise there is a connected component, say F_1, of $G[N_2]$ with an uncoloured vertex x. Then x has at most d red neighbours and at most d blue neighbours in N_1, since otherwise it would have been coloured by colour-processing. Let $B_x \subseteq N_1$ be the set of blue neighbours of x in N_1 and $R_x \subseteq N_1$ the set of red neighbours. We branch over all $O(n^{2d^2})$ colourings of $G[N(B_x \cup R_x) \cup V(F_1)]$.

We colour N_2 as follows. If x is coloured blue then any blue vertex of N_2 not belonging to F_1 must have a neighbour in B_x since $G[B]$ has diameter 2. This implies that any uncoloured vertex must be red since it has no neighbour in B_x. Similarly, if x is red then any uncoloured vertex must be blue. We check in polynomial time if the resulting colouring is a red-blue d-colouring. If yes, we have found a solution, otherwise we discard the branch.

If in any case any branch led to a red-blue d-colouring of G we immediately returned that G has a red-blue d-colouring and thus a d-cut. Otherwise if no branch led to a red-blue d-colouring we conclude that G has no d-cut.

The correctness of the algorithm follows from its description. The maximum number of branches in Case 1 is

$$O\left(n^4 * n^{4d} * \left(n^{2d} + n^{4d^2} * \left(n^{4d^2} * n^{2d} + n^{3d^2}\right)\right)\right)$$

and in Case 2 $O(16 * n^4 * n^{4d} * n^{2d^2})$. In total we have at most $n^{O(d^3)}$ branches and each branch can processed in polynomial time. Thus, the running time of our algorithm is polynomial. □

As our final result in this section, we prove the following.

Theorem 14. *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.*

Proof. Suppose that d-Cut is polynomial-time solvable for H-free graphs. Let G be a connected $(H + P_1)$-free graph. If G is H-free, the result follows by assumption. If G is not H-free, then $V(G)$ contains a set U such that $G[U]$ is isomorphic to H. As G is $(H + P_1)$-free, U dominates G. As H is fixed, $|U| = |V(H)|$ is a constant. Hence, in this case we can apply Lemma 8. □

4 NP-Completeness Results

In this section we show our NP-completeness results for d-Cut for $d \geq 2$. As d-Cut is readily seen to be in NP for each $d \geq 1$, we only show NP-hardness in our proofs.

For the proof of our first result, which is for the case $d = 2$ and $H = K_{1,4}$, we need some additional terminology. A *hypergraph* $H = (V, E)$ consists of a vertex set V and an edge set E, where an edge $e \in E$ is a set of vertices of H. The *size* of an edge in a hypergraph is the number of vertices it contains. The proof uses a reduction of Hypergraph 2-Colourability. This problem takes a hypergraph $H = (V, E)$ as input and asks if there exists a 2-colouring c that uses both colours on every edge of E. This problem is known to be NP-complete even for hypergraphs in which all edges have size 3 [22].

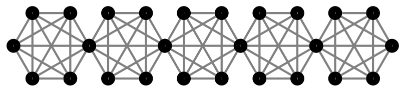
Fig. 5: A chain of cliques.

We call a *chain of cliques* of length k a set of k cliques of the same size, such that every clique, except two, has exactly two neighbouring cliques, with each of which it shares exactly one vertex, see Figure 5 for an example.

We are now ready to show our first result. Its proof generalizes the arguments of the corresponding proof for Matching Cut [7] for $d = 1$.

Theorem 15. *For $d = 2$, d-Cut is NP-complete for $K_{1,4}$-free graphs.*

Proof. We reduce from Hypergraph 2-Colourability. Let H be a hypergraph. We construct a graph G as follows. Figure 6 illustrates the construction. Let $n = |V(H)|$ be the number of vertices of H and let $m = |E(H)|$ be the number of edges of H.

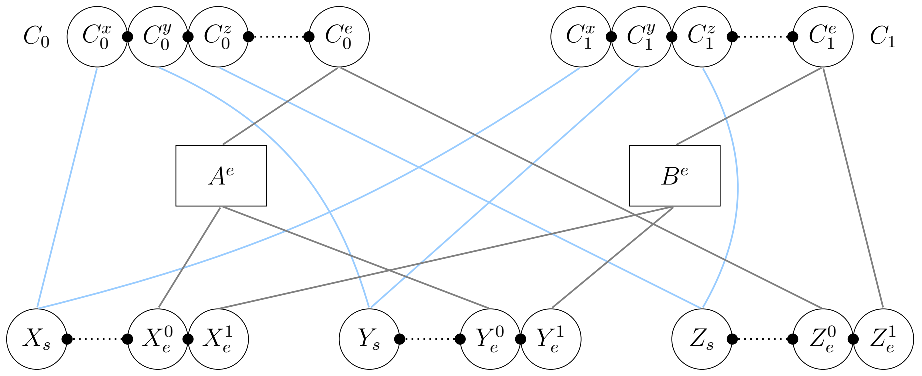

Fig. 6: The construction of G. In this example there is only one edge $e = xyz$. Every cycle represents a clique of size 5. The boxes represent independent sets.

Let C_0 and C_1 each be a chain of $n + m$ cliques of size 5. For every vertex $v \in V(H)$ there are cliques C_0^v in C_0 and C_1^v in C_1 and for every edge $e \in V(H)$ there are cliques C_0^e in C_0 and C_1^e in C_1. For every vertex $v \in V(H)$ we make a chain of cliques of size 5, containing two cliques per edge in which v appears, and one extra clique. We denote the two cliques corresponding to v and the edge e as V_e^0 and V_e^1. We denote the extra clique by V_s. For an edge $e = (u, v, w)$ we add two independent sets, each of two vertices, A_e and B_e. We define $A = \bigcup_{e \in E(H)} A_e$ and $B = \bigcup_{e \in E(H)} B_e$.

We add all edges between A_e and two vertices of U_e^0, all edges between A_e and two vertices of V_e^0, all edges between A_e and two vertices d_1 and d_2 of C_0^e and all edges between d_1 and d_2 and two vertices of W_e^0. We do the same for B_e and the sets U_e^1, V_e^1, W_e^1 and C_1^e. For every vertex $v \in V(H)$ we add all edges between two vertices s_1 and s_2 in V_s and two vertices in C_0^v as well as all edges between s_1 and s_2 and two vertices in C_1^v.

We claim that H is bicolourable if and only if G has a 2-cut. We assume that G has a 2-cut. That is G has a red-blue 2-colouring. Consider such a colouring.

Claim 15.1. *Every chain of cliques is monochromatic.*

Proof of the Claim. Every clique of size at least 5 must be monochromatic in any red-blue 2-colouring. Since the cliques of the chain have size 5 and two consecutive cliques in the chain share vertices, they all have the same colour. $\triangleleft$

Claim 15.2. *C_0 and C_1 have different colours.*

Proof of the Claim. Suppose that C_0 and C_1 are both coloured blue. Then the cliques V_s will all be coloured blue for every $v \in V(H)$, since they have two neighbours in each of C_0 and C_1. Thus, by Claim 15.1 all chains of cliques corresponding to vertices are coloured blue. The only vertices which remain uncoloured are the vertices in $A \cup B$. Since any vertex in $A \cup B$ has 6 blue neighbours, A and B are both coloured blue and thus the whole graph is coloured blue and the colouring is not a red-blue 2-colouring. Hence, C_0 and C_1 are coloured differently. $\triangleleft$

Claim 15.3. *Consider an edge $e = (u, v, w) \in E(H)$. The cliques U_e^0, V_e^0 and W_e^0 do not have all the same colour.*

Proof of the Claim. Suppose that there is an edge e in H, such that U_e^0, V_e^0 and W_e^0 are coloured the same. Without loss of generality we may assume they are all coloured blue. By Claim 15.1 we know that the chains in which U_e^0, V_e^0 and W_e^0 are contained are coloured blue as well and especially that U_e^1, V_e^1 and W_e^1 are blue. Without loss of generality we may further assume that U_e^0 and V_e^0 are connected to A_e and similarly that U_e^1 and V_e^1 are connected to B_e. Then every vertex in A_e and B_e has 4 blue neighbours (the vertices U_e^0 and V_e^0 or the vertices in U_e^1 and V_e^1). Thus, A_e and B_e are both coloured blue. Hence, the connector vertices in C_0^e and C_1^e have each 4 blue neighbours (the vertices in W_e^0 and A_e or the vertices in W_e^1 and B_e) and are also coloured blue. Due to Claim 15.1 this leads to C_0 and C_1 being both blue, a contradiction to Claim 15.2. $\triangleleft$

Since Claim 15.3 holds for every edge in $E(H)$, we get that if G has a red-blue d-colouring, then H is bicolourable, by colouring every vertex of H according to the colour of its corresponding chain in a red-blue 2-colouring.

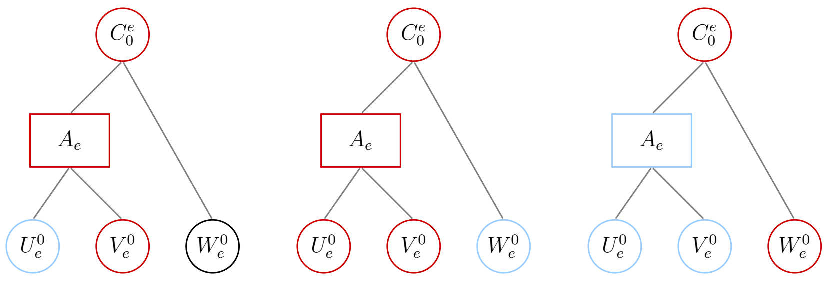

Fig. 7: The three possible red-blue 2-colourings of an edge gadget.

We assume now that H is bicolourable. We consider a bicolouring of the vertices of H. We colour the vertices of G as follows. For each chain of cliques in G corresponding to a vertex $v \in V(H)$, we colour the chain of cliques according to the colour of v in the bicolouring of H. Further we colour C_0 red and C_1 blue. Every vertex in A or B has two neighbours in each of its adjacent three cliques. We colour every vertex in A and B with the colour that at least two of its neighbouring cliques have. This clearly leads to a red-blue 2-colouring of the vertices in A and B. For $v \in V(H)$, every vertex in V_e^0, V_e^1, C_0^v and C_1^v has at most two neighbours outside of its monochromatic chain. Thus, no matter how we colour the vertices in A_e and B_e, the vertices in V_e^0, V_e^1, C_0^v and C_1^v have at most two neighbours of the other colour. The vertices in V_s which are connected to C_0 and C_1 have two neighbours of each colour, so also for them the colouring is a red-blue 2-colouring.

Consider an edge $e = (u, v, w) \in E(H)$. It remains to show that the vertices in C_0^e and C_1^e have at most two neighbours of the other colour. Consider a vertex $x \in C_0^e$. Note that the analysis for vertices in C_1^e works analogously. The vertex x has two neighbours in A_e and two neighbours in W_e^0. Since we consider a bicolouring of H we know that one or two of U_e^0, V_e^0 and W_e^0 are blue and one or two are red. We distinguish several cases. First suppose that U_e^0 and V_e^0 have different colours (see Figure 7 (left)). Say without loss of generality that U_e^0 is blue and V_e^0 is red. Independently of the colour of W_e^0 we colour A_e red, since it has two neighbouring red cliques (C_0^e and V_e^0). Thus, x is red and has two red neighbours in A_e and either 0 or 2 blue neighbours in W_e^0. Now suppose that U_e^0 and V_e^0 are both red (see Figure 7 (middle)). Then, due to the bicolouring of H, A_e is red and W_e^0 is blue. Thus, x has two blue and two red neighbours. The last case we consider is where U_e^0 and V_e^0 are both blue (see Figure 7 (right)). Then A_e is blue and W_e^0 is red. Again, x has two blue and two red neighbours. We get that the colouring is a red-blue 2-colouring and thus the theorem follows. □

For $d \geq 3$, we prove a stronger result than Theorem 15. 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 16. *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 10. 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 8:

- Build a clique $S = \{v_{x_1}^S, \ldots, v_{x_n}^S\} \cup \{v_1^{c_i}, \ldots, v_{d-2}^{c_i}\} \cup \cdots \cup \{v_1^{c_m}, \ldots, v_{d-2}^{c_m}\}$.
- Build a clique $\overline{S} = \{v_{x_1}^{\overline{S}}, \ldots, v_{x_n}^{\overline{S}}\} \cup \{u_1^{c_i}, \ldots, u_{d-2}^{c_i}\} \cup \cdots \cup \{u_1^{c_m}, \ldots, u_{d-2}^{c_m}\}$.
- For every $x \in X$, add cliques $V_x = \{v_1^x, \ldots, v_{d-1}^x\}$ and $V_{\overline{x}} = \{v_1^{\overline{x}}, \ldots, v_{d-1}^{\overline{x}}\}$.
- For every $x \in X$, add a vertex v_x with edges $v_x v_x^S$, $v_x v_x^{\overline{S}}$, $v_x v_1^x, \ldots, v_x v_{d-1}^x$, $v_x v_1^{\overline{x}}, \ldots, v_x v_{d-1}^{\overline{x}}$.
- For every $C \in \mathcal{C}$, add a clause vertex v_c with edges $v_c v_1^c, \ldots, v_c v_{d-2}^c$, $v_c u_1^c, \ldots,$ $v_c u_{d-2}^c$, and if $C = \{x_i, x_j, x_k\}$, also add a vertex $v_c^{x_i}$ to V_{x_i}, a vertex $v_c^{x_j}$ to V_{x_j} and a vertex $v_c^{x_k}$ to V_{x_k}, and add the edges $v_c v_c^{x_i}$, $v_c v_c^{x_j}$, $v_c v_c^{x_k}$.

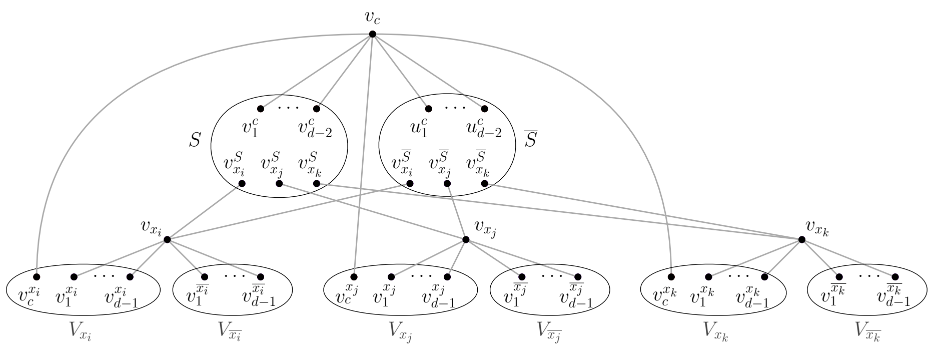

Fig. 8: An example of vertices in the reduction related to clause $C = \{x_i, x_j, x_k\}$.

– Add, if we 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 $L(G)$ has a d-cut. First suppose $L(G)$ has a d-cut. By Observation 7, we find that $L(G)$ has a red-blue d-colouring. Hence, G has a red-blue edge d-colouring c (note that G is the only graph with line graph $L(G)$ due to the Whitney isomorphism theorem). We prove a series of claims:

Claim 16.1 *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. $\triangleleft$

By Claim 16.1, 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 16.2 *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 $|D| \geq 2d+1$. $\triangleleft$

Claim 16.3 *The cliques S and $\overline{S}$ have different colours if and only if V_x and $V_{\overline{x}}$ have different colours for every variable $x \in X$.*

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 16.2, 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 $v_xv_x^S$ 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 16.2, 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. $\triangleleft$

Claim 16.4 *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 16.2, we have that $v_xv_x^S$ is blue. By Claim 16.3, 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 16.2. Consequently, the blue edge $v_xv_x^S$ 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 16.2, 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. $\triangleleft$

Claim 16.5 *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 16.4, S and $\overline{S}$ are coloured differently, say S is red and $\overline{S}$ is blue. By Claim 16.2, we have that the three edges $v_cv_c^{x_i}$, $v_cv_c^{x_j}$ and $v_cv_c^{x_k}$ 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. $\triangleleft$

For each variable x, if the clique V_x is coloured red, then set x to true, and else to false. By Claim 16.5, this yields a satisfying not-all-equal truth assignment.

Now 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 $D \in \{S, \overline{S}, V_{x_1}, V_{\overline{x_1}}, \dots, V_{x_n}, V_{\overline{x_n}}\}$. Colour uv with the same colour as the edges of D.

Now, let $D \in \{S, \overline{S}, V_{x_1}, V_{\overline{x_1}}, \dots, V_{x_n}, V_{\overline{x_n}}\}$. 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, and thus $L(G)$ has a red-blue d-colouring and thus a d-cut. $\square$

We now show that the case $H = 3P_2$ is hard. The gadget in our NP-hardness reduction is neither $2P_4$-free nor P_6-free nor P_7-free.

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

Proof. We reduce from Not-All-Equal Satisfiability. Let $(X, \mathcal{C})$ be an instance of this problem. By Theorem 10 we may assume that each variable occurs as a positive literal in exactly two clauses and as a negative literal in exactly two other clauses. Further, each clause consists of three distinct literals that are either all positive or all negative. Hence, let $X = \{x_1, x_2, \ldots, x_n\}$ for some $n \geq 1$ and $\mathcal{C} = \{C_1, \ldots, C_p, D_1, \ldots, D_q\}$ where each C_j consists of three distinct positive literals, and each D_j consists of three distinct negative literals. We may assume without loss of generality that $p \geq 4$ and $q \geq 4$, as otherwise the problem is trivial to solve by using brute force.

From $(X, \mathcal{C})$, we construct a graph $G = (V, E)$ as follows. We introduce two vertices C and D and let the other vertices of G represent either variables or clauses. That is, we introduce a clique $K = \{C_1, \ldots, C_p, C\}$; a clique $K' = \{D_1, \ldots, D_q, D\}$ and an independent set $I = \{x_1, \ldots, x_n\}$, such that K, K', I are pairwise disjoint and $V = K \cup K' \cup I$. For every $h \in \{1, \ldots, n\}$ and every $i \in \{1, \ldots, p\}$, we add an edge between x_h and C_i if and only if x_h occurs as a literal in C_i. For every $h \in \{1, \ldots, n\}$ and every $j \in \{1, \ldots, q\}$, we add an edge between x_h and D_j if and only if x_h occurs as a literal in D_j. We also add the edge CD. See Figure 9 for an example.

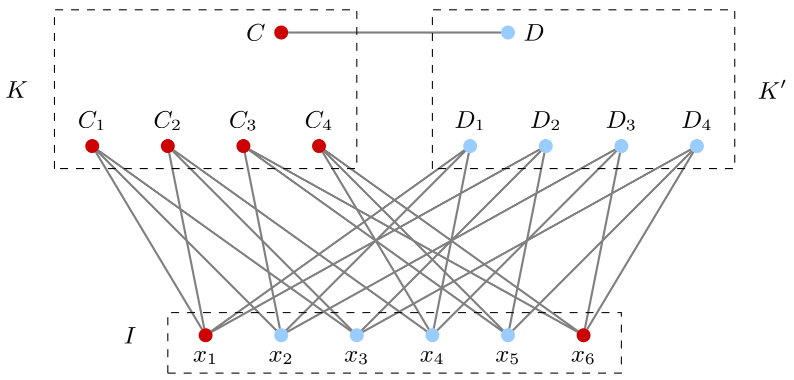

Fig. 9: The graph G for $X = \{x_1, \ldots, x_6\}$ and $\mathcal{C} = \{\{x_1, x_2, x_3\}, \{x_1, x_3, x_4\}, \{x_2, x_5, x_6\}, \{x_4, x_5, x_6\}, \{\overline{x_1}, \overline{x_2}, \overline{x_4}\}, \{\overline{x_1}, \overline{x_3}, \overline{x_5}\}, \{\overline{x_2}, \overline{x_4}, \overline{x_6}\}, \{\overline{x_3}, \overline{x_5}, \overline{x_6}\}\}$. For readability the edges inside the cliques K and K' are not shown.

As every edge must have at least one end-vertex in K or K', and K and K' are cliques, we find that G is $3P_2$-free. Moreover, G has radius 2, as the distance from C or D to any other vertex in G is at most 2. In addition, the distance from a vertex in $I \cup (K \setminus \{C\}) \cup (K' \setminus \{D\})$ to any other vertex in G is at most 3. Hence, G has diameter at most 3.

We claim that $(X, \mathcal{C})$ is a yes-instance of Not-All-Equal-Satisfiability if and only if G has a 2-cut.

First suppose $(X, \mathcal{C})$ is a yes-instance of Not-All-Equal-Satisfiability. Then X has a truth assignment ϕ that sets, in each C_i and in each D_j, at least one literal true and at least one literal false. In I, we colour for $h \in \{1, \dots, n\}$, vertex x_i red if ϕ sets x_i to be true and blue if ϕ sets x_i to be false. We colour all the vertices in K red and the vertices in K' blue.

Consider a vertex x_h in I. First suppose that x_h is coloured red. As each literal appears in exactly two clauses from $\{D_1, \dots, D_q\}$, we find that x_h has only two blue neighbours (which all belong to K'). Now suppose that x_h is coloured blue. As each literal appears in exactly two clauses from $\{C_1, \dots, C_p\}$, we find that x_h has only two red neighbours (which all belong to K). Now consider a vertex C_i in K, which is coloured red. As C_i consists of three distinct positive literals and ϕ sets at least one positive literal of C_i to be true, we find that C_i is adjacent to at most two blue vertices in I. Hence, every C_i is adjacent to at most two blue vertices. Now consider a vertex D_j in K', which is coloured blue. As D_j consists of three distinct negative literals and ϕ sets at least one negative literal of D_j to be true, we find that D_j is adjacent to at most two red vertices in I. Hence, every D_j is adjacent to at most two red vertices. Finally, we note that C, which is coloured red, is adjacent to exactly one blue neighbour, namely D, while D has only one red neighbour, namely C. The above means that we obtained a red-blue 2-colouring of G. By Observation 7, this means that G has a 2-cut.

Now suppose G has a 2-cut. By Observation 7, this means that G has a red-blue 2-colouring c. As $|K| = p + 1 \geq 5$ and $|K'| = q + 1 \geq 5$, both K and K' are monochromatic. Say c colours every vertex of K red. For a contradiction, assume c colours every vertex of K' red as well. As c must colour at least one vertex of G blue, this means that I contains a blue vertex x_i. As each variable occurs as a positive literal in exactly two clauses and as a negative literal in exactly two other clauses, we now find that a blue vertex, x_i, has two red neighbours in K and two red neighbours in K', so four red neighbours in total, a contradiction. We conclude that c must colour every vertex of K' blue.

Recall that every C_i and every C_j consists of three literals. Hence, every vertex in $K \cup K'$ has three neighbours in I. As every vertex C_i in K is red, this means that at least one neighbour of C_i in I must be red. As every vertex D_j in K' is blue, this means that at least one neighbour of D_j in I must be blue. Hence, setting x_i to true if x_i is red in G and to false if x_i is blue in G gives us the desired truth assignment for X.

Now, we consider the case where $d \geq 3$. We adjust G as follows. We first modify K into a larger clique by adding for each x_h, a set L_h of $d-3$ new vertices. We also modify K' into a larger clique by adding for each x_h, a set L'_h of $d-3$ vertices. For each $h \in \{1, \dots, n\}$ we make x_h complete to both L_h and to L'_h. Finally, we add additional edges between vertices in $K \cup L_1 \cup \dots \cup L_n$ and vertices in $K' \cup L'_1 \cup \dots \cup L'_n$, in such a way that every vertex in $K \setminus \{C\}$ has $d-2$ neighbours in $K' \setminus \{D\}$, and vice versa. The modified graph G is still $3P_2$-free, has radius 2 and diameter 3, and also still has size polynomial in m and n. The remainder of the proof uses the same arguments as before. □

Our final theorem is a straightforward generalization of the case $d = 1$ from [25]; see Appendix A where we give its proof for completeness.

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. While for $d = 1$, there still exists an infinite number of non-equivalent open cases, we were able to obtain for every $d \geq 2$, an almost-complete complexity classification of d-Cut for H-free graphs, with only three non-equivalent open cases left if $d \geq 3$. 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 computational complexity of 2-Cut for line graphs and for claw-free graphs? An answer to this question could lead to a jump in complexity from $d = 2$ to $d = 3$. We have not yet seen such a jump for other graph classes.

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 $d = 1$, whereas for $d = 2$ all three cases are still open. We are currently investigating the cases $H = 2P_4$ and $H = P_6$ and believe these cases to be polynomial-time solvable for every $d \geq 1$, whereas for the case $H = P_7$ we do not have any conjecture on its complexity.

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. Darmann, A., Döcker, J.: On simplified NP-complete variants of Monotone 3-Sat. Discrete Applied Mathematics **292**, 45–58 (2021)

9. Farley, A.M., Proskurowski, A.: Networks immune to isolated line failures. Networks **12**, 393–403 (1982)
10. Feghali, C.: A note on Matching-Cut in P_t-free graphs. Information Processing Letters **179**, 106294 (2023)
11. 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)
12. Golovach, P.A., Paulusma, D., Song, J.: Computing vertex-surjective homomorphisms to partially reflexive trees. Theoretical Computer Science **457**, 86–100 (2012)
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. Lovász, L.: Coverings and colorings of hypergraphs. In: Proc. 4th Southeastern Conference of Combinatorics, Graph Theory, and Computing. pp. 3–12. Utilitas Mathematica Publishing (1973)
23. 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)
24. 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)
25. Lucke, F., Paulusma, D., Ries, B.: Finding matching cuts in H-free graphs. Algorithmica **85**, 3290–3322 (2023)
26. Moshi, A.M.: Matching cutsets in graphs. Journal of Graph Theory **13**, 527–536 (1989)
27. Patrignani, M., Pizzonia, M.: The complexity of the Matching-Cut problem. Proc. WG 2001, LNCS **2204**, 284–295 (2001)
28. Schaefer, T.J.: The complexity of satisfiability problems. Proc. STOC 1978 pp. 216–226 (1978)

A The Proof of Theorem 18

In this section, we give the proof of Theorem 18. We recall that it is a (very) straightforward generalization of the proof in [11] for the corresponding result for $d = 1$, and below we just give a full proof for completeness.

Let uv be an edge of a graph G. We define an edge operation as displayed in Figure 10, which when applied on uv will replace uv in G by the subgraph T^i_{uv}. Note that in the new graph, the only vertices from T^i_{uv} that may have neighbours outside T^i_{uv} are u and v.

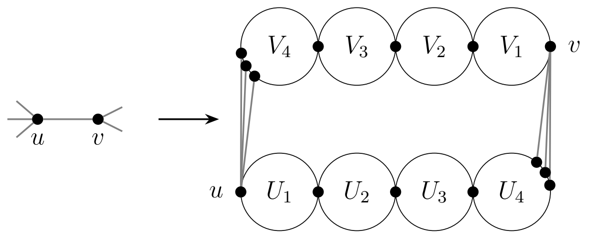

Fig. 10: The edge uv (left) which we replace by the subgraph T^i_{uv} (right), here in the case where $d = 3$ and $i = 4$. Every cycle represents a clique of size $2d + 1$. The vertex u (v, resp.) has d neighbours in V_i (U_i, resp.).

Theorem 18 (restated). *For $d \geq 2$ and $i \geq 1$, d-Cut is NP-complete for $(H^*_1, \ldots, H^*_i)$-free graphs.*

Proof. Fix $i \geq 1$. We reduce from Matching Cut. Let $G = (V, E)$ be a connected graph. Replace every $uv \in E$ by the graph T^i_{uv} (see Figure 10). This yields the graph $G' = (V', E')$.

We claim that G' is $(H^*_1, \ldots, H^*_i)$-free. For a contradiction, assume that G' contains an induced $H^*_{i'}$ for some $1 \leq i' \leq i$. Then G' contains two vertices x and y that are centers of an induced claw, as well as an induced path from x to y of length i'. All vertices in $V' \setminus V$ are not centers of any induced claw. Hence, x and y belong to V. By construction, any shortest path between two vertices of V has length at least $i + 1$ in G', a contradiction.

We claim that G' has a d-cut if and only if G has a matching cut. First suppose G' has a d-cut M', so G' has a red-blue d-colouring c'. We prove a claim for G':

Claim 18.1. *For every edge $uv \in E(G)$ it holds that:*

(a) either $c'(u) = c'(v)$, and then T^i_{uv} is monochromatic, or
(b) $c'(u) \neq c'(v)$, and then c' colours $U_1, \ldots, U_i$ with the same colour as u, while c' colours all vertices of $T^i_{uv} - \{u, U_1, \ldots, U_i\}$ with the same colour as v, and moreover, the d edges from u to V_i and the d edges from v to U_i are in M'.

Proof of the Claim. First assume $c'(u) = c'(v)$, say c' colours u and v red. As any clique of size at least $2d+1$ is monochromatic, all vertices in T^i_{uv} are coloured red, so T^i_{uv} is monochromatic.

Now assume $c'(u) \neq c'(v)$, say u is red and v blue. As before, we find that all cliques $U_1, \ldots, U_i$ have the same colour as u, so are red, while all cliques

$V_1, \ldots, V_i$ have the same colour as v, so are blue. By definition, every edge xy with $c'(x) \neq c'(y)$ belongs to M'. $\lhd$

We construct a subset $M \subseteq E$ in G as follows. We add an edge $uv \in E$ to M if and only if $c'(u) \neq c'(v)$ in G'. We now show that M is a matching in G. Let $u \in V$. For a contradiction, suppose that M contains edges uv and uw for $v \neq w$. Then $c'(u) \neq c'(v)$ and $c'(u) \neq c'(w)$. By Claim 18.1, we find that M' matches u in G' to d vertices in each of T^i_{uv} and T^i_{uw}, contradicting our assumption that M' is a d-cut. Hence, M is a matching.

Now let c be the restriction of c' to V. If c colours every vertex of G with one colour, say red, then c' would also colour every vertex of G' red by Claim 18.1, contradicting that c' is a red-blue d-colouring. Hence, c uses both colours. Moreover, for every $uv \in E$, the following holds: if $c(u) \neq c(v)$, then $c'(u) \neq c'(v)$ and thus $uv \in M$. Hence, as M is a matching, c is a red-blue 1-colouring, and thus M is a matching cut of G.

Conversely, assume that G admits a matching cut, so V has a red-blue 1-colouring c. We construct a red-blue d-colouring c' of V' as follows.

– For every edge $uv \in E$ with $c(u) = c(v)$, we let $c'(x) = c(u)$ for every $x \in V(T^i_{uv})$.
– For every edge $uv \in E$ with $c(u) \neq c(v)$, we let $c'(u) = c(u)$ and $c'(u') = c(u)$, for every $u' \in \bigcup_{j=1}^{i} U_j$ and similarly, $c'(v) = c(v)$ as well as $c'(v') = c(v)$, for every $v' \in \bigcup_{j=1}^{i} V_j$.

As c is a red-blue 1-colouring, c uses both colours and thus by construction, c' uses both colours. Let $u \in V$. Again as c is a red-blue 1-colouring, $c(u) \neq c(v)$ holds for at most one neighbour v of u in G. Hence, by construction, u belongs to at most one non-monochromatic gadget T^i_{uv}. Thus, c' colours in G' at most d neighbours of u with a different colour than u. Let $u \in V' \setminus V$. By construction, we find that c' colours at most d neighbours of u with a different colour than u. Hence, c' is a red-blue d-colouring, and so G' has a d-cut. This completes the proof. $\square$