A Complexity Dichotomy for Finding Disjoint Solutions of Vertex Deletion Problems[1,2]

Michael R. Fellows, Charles Darwin University
Jiong Guo, Universität des Saarlandes
Hannes Moser, Friedrich-Schiller-Universität Jena
Rolf Niedermeier, TU Berlin

We investigate the computational complexity of a general "compression task" centrally occurring in the recently developed technique of iterative compression for exactly solving NP-hard minimization problems. The core issue (particularly but not only motivated by iterative compression) is to determine the computational complexity of the following task: given an already inclusion-minimal solution for an underlying (typically NP-hard) vertex deletion problem in graphs, find a smaller *disjoint* solution. The complexity of this task is so far lacking a systematic study. We consider a large class of vertex deletion problems on undirected graphs and show that a few cases are polynomial-time solvable, and the others are NP-hard. The considered class of vertex deletion problems includes Vertex Cover (where the compression task is polynomial-time) and Undirected Feedback Vertex Set (where the compression task is NP-complete).

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems; G.2.2 [**Discrete Mathematics**]: Graph Theory—*Graph algorithms*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Iterative compression, computational complexity, parameterized complexity, graph algorithms, hereditary graph properties, vertex deletion problems

ACM Reference Format:
Fellows, M. R., Guo, J., Moser, H., Niedermeier, R. 2011. A Complexity Dichotomy for Finding Disjoint Solutions of Vertex Deletion Problems. ACM V, N, Article A (January YYYY), 23 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

[1]A preliminary version of this paper appears in the proceedings of the 34th International Symposium on Mathematical Foundations of Computer Science (MFCS '09), volume 5734 of LNCS, pages 319–330, Springer, 2009.
[2]Main work was done while all authors were staying at the University of Jena.

The first author was supported by the Australian Research Council. Work done while staying in Jena as a recipient of a Research Award from the Alexander von Humboldt Foundation, Bonn, Germany. The second author was partially supported by the DFG, PALG, NI 369/8. The third author was supported by the DFG, project AREG, NI 369/9.
Authors' addresses: M. R. Fellows, School of Engineering and Information Technology, Charles Darwin University, Darwin, Northern Territory 0909, Australia; email: michael.fellows@cdu.edu.au; J. Guo, Universität des Saarlandes, Campus E 1.4, D-66123 Saarbrücken, Germany; email: jguo@mmci.uni-saarland.de; H. Moser, Institut für Informatik, Friedrich-Schiller-Universität Jena, Ernst-Abbe-Platz 2, D-07743 Jena, Germany; email: hannes.moser@uni-jena.de; R. Niedermeier, Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, D-10587 Berlin, Germany; email: rolf.niedermeier@tu-berlin.de

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

The introduction of the iterative compression technique by Reed et al. [2004] provided parameterized algorithm design with a general new tool for showing fixed-parameter tractability results for NP-hard minimization problems (cf. [Guo et al. 2009; Niedermeier 2006]). In 2008, the technique led to major breakthroughs concerning the classification of the parameterized complexity of two important problems. First, Chen and Liu et al. [2008] showed that the NP-complete DIRECTED FEEDBACK VERTEX SET problem is fixed-parameter tractable. Second, Razgon and O'Sullivan [2009] proved that the NP-complete ALMOST 2-SAT problem is fixed-parameter tractable. Refer to the recent survey by Guo et al. [2009] for more on applications of the iterative compression technique to exactly solving NP-hard minimization problems.

The central idea behind iterative compression is to employ a *compression routine*. This is an algorithm that, given a problem instance and a corresponding solution, either calculates a smaller solution or proves that the given solution is of minimum size. Using a compression routine, one finds an optimal solution to a problem by inductively building up the problem instance and iteratively compressing intermediate solutions. Herein, the essential fact from the viewpoint of parameterized complexity is that if the task performed by the compression routine is fixed-parameter tractable, then so is the problem solved by means of iterative compression. The main strength of iterative compression is that it allows us to see the problem from a different angle: the compression routine has as input not only an "intermediate" instance (e.g., a graph), but also an "intermediate" solution (e.g., a vertex cover) that provides valuable structural information.

While embedding the compression routine into the iteration framework is usually straightforward, finding the compression routine itself is not [Guo et al. 2009; Niedermeier 2006]. For many vertex deletion problems, a common approach to designing a compression routine is to branch on the possible subsets of the uncompressed solution that are to be retained in the compressed solution. This leads to the following generic problem that asks for a *disjoint* compressed solution:

COMPRESSION TASK
Input: An instance of the underlying NP-hard problem and a solution X.
Question: Is there a solution X' such that $X' \cap X = \emptyset$ and $|X'| < |X|$?

We study the complexity of this COMPRESSION TASK as it varies according to the underlying NP-hard vertex deletion problem. The computational complexity of COMPRESSION TASK has not yet been systematically investigated—this is our main objective. For example, the fixed-parameter tractability results (using iterative compression) for VERTEX BIPARTIZATION [Reed et al. 2004] or UNDIRECTED FEEDBACK VERTEX SET [Chen et al. 2008; Dehne et al. 2007; Guo et al. 2006] leave open whether the respective COMPRESSION TASK is NP-hard or polynomial-time solvable. By way of contrast, the fixed-parameter tractability result for the NP-complete CLUSTER VERTEX DELETION problem [Hüffner et al. 2010] is based on a polynomial-time algorithm for the COMPRESSION TASK for that problem. Here, extending a framework attributed to Yannakakis [Lewis and Yannakakis 1980], we describe a complete classification of COMPRESSION TASK for a wide class of vertex deletion problems (specified by a graph property Π), including all of the above mentioned specific problems.

A *graph property* Π is a set of graphs; in the following, we say that a graph G *satisfies* Π if $G \in \Pi$. A graph property Π is *hereditary* if it is closed under vertex deletion, and *non-trivial* if it is *satisfied* by infinitely many graphs and it is *not satisfied* by infinitely many graphs.

The classical Π-VERTEX DELETION problem is defined as follows.

Π-VERTEX DELETION
Input: An undirected graph $G = (V, E)$ and a nonnegative integer k.
Question: Is there a vertex subset $S \subseteq V$ of size at most k such that $G - S \in \Pi$?

For example, UNDIRECTED FEEDBACK VERTEX SET corresponds to the case that Π means "being cycle-free". Yannakakis has shown that Π-VERTEX DELETION is NP-complete for any non-trivial hereditary graph property Π in general graphs [Lewis and Yannakakis 1980]. General vertex deletion problems have also been studied in terms of their parameterized complexity, where the parameter is the number of vertices to be deleted [Cai 1996] (studying finite forbidden sets). The family of dual problems, that is, subgraph problems for hereditary graph properties, have been studied with respect to their parameterized complexity as well, where the parameter is the number of vertices in the (induced) subgraph [Khot and Raman 2002].

The COMPRESSION TASK restricted to the vertex deletion problem for a graph property Π, called DISJOINT Π-VERTEX DELETION, can be formulated as follows:

DISJOINT Π-VERTEX DELETION
Input: An undirected graph $G = (V, E)$ and a vertex subset $X \subseteq V$ such that $G[X] \in \Pi$, $G[V \setminus X] \in \Pi$, and X is inclusion-minimal under this property, that is, for every proper subset $X' \subset X$ the graph $G[V \setminus X']$ does not satisfy Π.
Question: Is there a vertex subset $X' \subseteq V$ of size less than $|X|$ such that $X \cap X' = \emptyset$ and $G[V \setminus X']$ satisfies Π?

We demand that $G[X] \in \Pi$ since, otherwise, there cannot exist a solution X' disjoint from X. We also demand that X is inclusion-minimal; any solution can be made inclusion-minimal in polynomial time if Π can be tested in polynomial time. Thus, in these cases, this requirement does not change the complexity.

A graph property Π is *determined by the components* if it holds that if every connected component of the graph satisfies Π, then so does the whole graph. For example, the graph property "cycle-free" is determined by the components, but the graph property "at least five cycle-free components" is obviously not determined by the components. The central result of this work can be informally stated as follows:

MAIN THEOREM: *Let Π be any non-trivial hereditary graph property that is determined by the components and that can be tested in polynomial time.* DISJOINT Π-VERTEX DELETION *is NP-complete unless Π is the set of all graphs whose connected components are cliques or Π is the set of all graphs whose connected components are cliques of at most s vertices, $s \geq 1$—in these cases it is polynomial-time solvable.*[3]

This theorem applies to many natural vertex deletion problems in undirected graphs, including VERTEX COVER, BOUNDED-DEGREE VERTEX DELETION [Nishimura et al. 2005; Moser et al. 2009], UNDIRECTED FEEDBACK VERTEX SET [Guo et al. 2006; Dehne et al. 2007; Chen et al. 2008], VERTEX BIPARTIZATION [Reed et al. 2004], CLUSTER VERTEX DELETION [Hüffner et al. 2010], CHORDAL DELETION [Marx 2010], and PLANAR DELETION [Marx and Schlotter 2007]. Among these problems, except for VERTEX COVER, CLUSTER VERTEX DELETION, and BOUNDED-DEGREE-1 VERTEX DELETION, all other problems have NP-complete DISJOINT Π-VERTEX DELETION associated compression tasks.

Our original motivation for analyzing the complexity of DISJOINT Π-VERTEX DELETION comes from the desire to better understand the limitations of the iterative compression technique. Beyond this, DISJOINT Π-VERTEX DELETION also seems to be

[3]There might exist other polynomial-time solvable cases for *non-hereditary* properties.

a natural and interesting problem on its own: In combinatorial optimization, one often may be confronted with finding *alternative* good solutions to ones already found. In the setting of DISJOINT Π-VERTEX DELETION, this is put to the extreme in the sense that we ask for solutions that are completely unrelated, that is, disjoint. For instance, this demand also naturally occurs in the context of finding quasicliques [Abello et al. 2002]. The computational complexity of finding alternative solutions with less strict demands has been considered in various contexts. Among others, the complexity of finding alternative solutions has been studied with respect to the HAMILTON CYCLE problem [Papadimitriou 1994; Krawczyk 1999]. More generally, local search is a very general technique in combinatorial optimization where one "explores" the space of possible solutions by moving from one solution to a "close" better solution (if possible) [Aarts and Lenstra 1997]. However, this does not ask for disjointness of solutions. Finally, let us mention that there are also ties to the recent framework of reoptimization problems [Böckenhauer et al. 2008]—there, one deals with the recomputation of a solution for a locally modified input instance. In all these settings one asks how the knowledge of a solution can provide structural information that helps in finding another one.

The remainder of this paper provides the proof of the main theorem. It is organized as follows. After some preliminaries (Section 2), we describe a general iterative compression framework for vertex deletion problems and show how the compression task DISJOINT Π-VERTEX DELETION is employed (Section 3). After that, we show the announced complexity dichotomy (Section 4). Section 4 is roughly organized as follows. We first address the polynomial-time solvable cases (corresponding to certain graph properties Π) of DISJOINT Π-VERTEX DELETION (Section 4.1). Then, extending a framework of Lewis and Yannakakis [1980], we provide several different constructions for showing the NP-completeness of *all* other DISJOINT Π-VERTEX DELETION problems, yielding our complexity dichotomy (Sections 4.2 and 4.3). We end with some challenges for future work (Section 5).

2. PRELIMINARIES

We only consider undirected graphs $G = (V, E)$ with $n := |V|$ and $m := |E|$. We write $V(G)$ and $E(G)$ to denote, respectively, the vertex and edge set of a graph G. For $v \in V$, let $N_G(v) := \{u \in V \mid \{u, v\} \in E\}$ and let $\deg_G(v) := |N_G(v)|$. For $S \subseteq V$, let $N_G(S) := \bigcup_{v \in S} N(v) \setminus S$. For $S \subseteq V$, let $G[S]$ be the subgraph of G induced by S and $G - S := G[V \setminus S]$. For $v \in V$, let $G - v := G[V \setminus \{v\}]$. For a connected graph G, a *cut-vertex* is a vertex $v \in V$ such that $G - v$ is not connected. A K_t for $t \geq 3$ is a complete graph on t vertices. A P_t for $t \geq 2$ is a path on t vertices. For $s \geq 1$, the graph $K_{1,s} = (\{u, v_1, \ldots, v_s\}, \{\{u, v_1\}, \ldots, \{u, v_s\}\})$ is a *star*. The vertex u is the *center* of the star and the vertices $v_1, \ldots, v_s$ are the *leaves* of the star.

If a graph H does not satisfy some hereditary property Π, then any supergraph of H does not satisfy Π. We call H a *forbidden subgraph* for Π. For any hereditary property Π there exists a set $\mathcal{H}$ of "minimal" forbidden induced subgraphs, that is, forbidden graphs for which every induced subgraph satisfies Π [Greenwell et al. 1973]. For this work, we restrict our attention to non-trivial hereditary properties that are determined by the components. For the corresponding characterization of Π by forbidden induced subgraphs, this means that the set of forbidden subgraphs contains only *connected* graphs.

Since the number of sets of graphs is uncountable, but the set of algorithms can be enumerated and is therefore countable, it follows that there exist DISJOINT Π-VERTEX DELETION problems that are not in NP. As in Lewis and Yannakakis [1980], we add the stipulation that Π can be tested in polynomial time, hence the corresponding DIS-

Algorithm: ITERATE (G, k)
Input: An undirected graph G and a nonnegative integer k.
Output: A Π-deletion set of size at most k, or "no-instance".

```
1 V' ← ∅
2 S ← ∅
3 while V' ≠ V do
4       select a vertex v ∈ V \ V'
5       V' ← V' ∪ {v}
6       S ← S ∪ {v}
7       S ← COMPRESS(G[V'], S)
8       if |S| > k then return "no-instance"
9 return S
```

Fig. 1: Pseudo-code of the algorithm to solve Π-VERTEX DELETION via iterative compression. The pseudo-code of the algorithm COMPRESS is given in Fig. 2.

JOINT Π-VERTEX DELETION problem is in NP, and our NP-hardness results to come will thus show NP-completeness.

A parameterized problem (I, k) is *fixed-parameter tractable* with respect to the parameter k if it can be solved in $f(k) \cdot \mathrm{poly}(|I|)$ time, where I is the input instance and f is some computable function. The corresponding algorithm is called a *fixed-parameter algorithm*.

3. ITERATIVE COMPRESSION AND THE COMPRESSION TASK

In this section, we give a general iterative compression framework for Π-VERTEX DELETION, where Π is a non-trivial hereditary property that is determined by the components.

First, we show how to employ the compression routine.

Iteration. In the following, we call a solution for Π-VERTEX DELETION a Π-*deletion set*. The pseudo-code of the iteration algorithm is given in Fig. 1. We start with empty vertex subsets $V' = \emptyset$ and $S = \emptyset$ (lines 1 and 2); clearly, an empty set is a Π-deletion set for an empty graph. Iterating over all graph vertices, step by step we add one vertex $v \in V \setminus V'$ to both V' and S (lines 4–6). Then S is still a Π-deletion set for $G[V']$. In each step we try to find a smaller Π-deletion set for $G[V']$ by applying a compression routine (line 7). It takes the graph $G[V']$ and the Π-deletion set S for $G[V']$, and returns a smaller Π-deletion set for $G[V']$, or proves that S is optimal (by returning a Π-deletion set of the same size). Note that a smaller Π-deletion set for $G[V']$ is always a minimum one and has size $|S| - 1$, since $S \setminus \{v\}$ is a minimum Π-deletion set for $G[V' \setminus \{v\}]$, and a minimum Π-deletion set for $G[V']$ cannot be smaller than a minimum Π-deletion set for $G[V' \setminus \{v\}]$.[4] Therefore, it is a loop invariant that the compressed Π-deletion set S is a minimum-size Π-deletion set for $G[V']$. If $|S| > k$ (line 8), then we can conclude that G does not have a Π-deletion set of size at most k. Since eventually $V' = V$, we obtain a Π-deletion set of size at most k for G once the algorithm returns S (line 9).

Compression. It remains to describe the compression routine. Given an undirected graph G and a solution S for Π-VERTEX DELETION, the compression routine finds a smaller solution for G or proves that the solution S is of minimum size. To this end, we consider all partitions of S into one part to keep in the solution and one part to

[4]In other words, Π-VERTEX DELETION behaves *monotonically* with respect to adding vertices. This follows from the fact that the graph property Π is hereditary.

Algorithm: COMPRESS (G, X)
Input: An undirected graph G and a solution S
Output: A smaller solution S', if it exists, otherwise S.

```
1 for each Y ⊊ S
2       X ← S \ Y
3       G' ← G − Y
4       if G[X] ∈ Π then
5           X' ← CompressDisjoint (G', X)
6           if |X'| < |X| then return X' ∪ Y
7 return S
```

Fig. 2: Pseudo-code of the compression routine for Π-VERTEX DELETION. Algorithm COMPRESSDISJOINT solves DISJOINT Π-VERTEX DELETION.

exchange. The compression routine works as follows. See Fig. 2 for the corresponding pseudo-code. Consider a smaller Π-deletion set S' as a modification of the larger Π-deletion set S for the graph $G = (V, E)$. This modification retains some vertices $Y \subsetneq S$ as part of the solution set (that is, the vertices to be deleted), while the other vertices $X := S \setminus Y$ are replaced by new vertices from $V \setminus S$. The idea is to try by brute force all $2^{|S|} - 1$ nontrivial partitions of S into these two sets Y and X. For each such partition, the vertices from Y are immediately deleted from S (line 2) and G (line 3), since we have already decided to take them into the Π-deletion set. If $G[X] \notin \Pi$, then we know that there can be no solution S' with $S' \cap S = Y$; hence, we only proceed if $G[X] \in \Pi$ (line 4). In the remaining instance $G' := G - Y$, it remains to find a smaller Π-deletion set X' that is disjoint from X (line 5). This task, DISJOINT Π-VERTEX DELETION, is solved by a *disjoint compression routine* (COMPRESSDISJOINT). If such a smaller solution is found, it is returned (line 6). Otherwise, after trying all possible partitions without finding a smaller solution, we know that the solution S is optimal and return it (line 7).

The running time of the whole algorithm can be stated as follows.

LEMMA 3.1. *If* DISJOINT Π-VERTEX DELETION *can be solved in* t_1 *time and the property* Π *can be tested in* t_2 *time, then* Π-VERTEX DELETION *can be solved in* $O(2^k \cdot (t_1 + t_2)n)$ *time.*

PROOF. The loop of algorithm ITERATE in Fig. 1 (lines 3–8) can be executed at most n times. In each iteration, the algorithm COMPRESS in Fig. 2 is called (line 7). After that, ITERATE aborts in line 8 if $|S| > k$. Therefore, in the next iteration of the loop, S has size at most k before adding v to it in line 6 of ITERATE. Thus, S has size at most $k + 1$ if COMPRESS is called. Therefore, COMPRESS executes its loop (lines 2–6 in Fig. 2) at most $2^{k+1} - 1$ times. COMPRESS needs t_2 time to test whether $G[X] \in \Pi$ (line 4) and t_1 time to compress the solution X (line 5). The remaining instructions in lines 2, 3, and 6, can be executed in linear time in the worst case. Thus, each execution of COMPRESS needs $O(2^k \cdot (t_1 + t_2))$ time. In total, this sums up to a running time of $O(2^k \cdot (t_1 + t_2)n)$ for algorithm ITERATE. □

The idea to try by brute force all nontrivial partitions of a given solution S into two sets Y and X is applied for all known applications of iterative compression for Π-VERTEX DELETION problems, that is, VERTEX BIPARTIZATION [Reed et al. 2004], UNDIRECTED FEEDBACK VERTEX SET [Guo et al. 2006; Dehne et al. 2007; Chen et al. 2008], VERTEX COVER [Guo 2006], and CLUSTER VERTEX DELETION [Hüffner et al. 2010]. The main difference between these problems lies in the disjoint compres-

sion routine. The disjoint compression routines for the first two problems have exponential worst-case running time; for VERTEX BIPARTIZATION DISJOINT Π-VERTEX DELETION is solved via a brute-force approach combined with maximum flow techniques [Reed et al. 2004], and for UNDIRECTED FEEDBACK VERTEX SET it is solved with a bounded search tree approach combined with data reduction rules [Chen et al. 2008]. The disjoint compression routine for CLUSTER VERTEX DELETION runs in polynomial time and is based on matching techniques, and the disjoint compression routine for VERTEX COVER is a trivial algorithm applying the simple observation that a disjoint solution X' must contain all neighbors of X. From Lemma 3.1 it follows directly that for VERTEX COVER and CLUSTER VERTEX DELETION there exists an algorithm with running time $2^k \cdot \mathrm{poly}(n)$. The results in this paper are driven by the following question: *For which other vertex deletion problems is* DISJOINT Π-VERTEX DELETION *polynomial-time solvable?*

In the next section, we answer this question by establishing a complexity dichotomy for DISJOINT Π-VERTEX DELETION for any non-trivial hereditary property Π that is determined by the components.

4. COMPLEXITY DICHOTOMY FOR THE COMPRESSION TASK

This section is dedicated to the proof of the following theorem. This is the main theorem stated in the introduction with the difference that here a graph property Π is characterized via its corresponding set of forbidden induced subgraphs. In Section 4.1 we argue that the formulations used here and in the introduction are equivalent.

THEOREM 4.1. *Let Π be any non-trivial hereditary graph property that is determined by the components and that can be tested in polynomial time, and let $\mathcal{H}$ be the set of minimal forbidden induced subgraphs corresponding to* Π. DISJOINT Π-VERTEX DELETION *is NP-complete unless $\mathcal{H}$ contains P_2 or P_3, and in these cases it is polynomial-time solvable.*

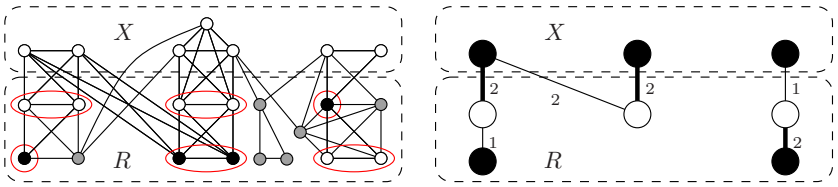
4.1. The Polynomial-Time Solvable Cases

This section covers all cases of DISJOINT Π-VERTEX DELETION that can be solved in polynomial time. These correspond to each graph property Π whose set of minimal forbidden induced subgraphs $\mathcal{H}$ contains P_2 (a single edge) or P_3 (a path on three vertices). Recall that we restricted our attention to hereditary graph properties that are determined by the components, that is, all graphs in H are connected.

We shortly describe the "structure" of the corresponding graph properties. If $\mathcal{H}$ contains P_2, then this is the only forbidden induced subgraph, that is, $\mathcal{H} = \{P_2\}$, since any other connected graph contains P_2 (recall that $\mathcal{H}$ can be assumed to be a set of minimal forbidden induced subgraphs). Hence, Π is the set of all graphs with no edges, that is, it is the graph property "being edgeless". If $\mathcal{H}$ contains no P_2 but P_3, then $\mathcal{H}$ can additionally contain exactly one clique, since any other connected graph on at least four vertices contains P_3 as induced subgraph. If $\mathcal{H}$ contains only P_3, then Π is the set of all *cluster graphs*, that is, graphs whose connected components form cliques. If $\mathcal{H}$ contains P_3 and a clique K_t, then the corresponding graph property Π is the set of all graphs whose connected components form cliques of size at most $t-1$.

In the following, we name these graph properties according to the following definition.

Definition 4.2. Let Π_s, for $s \geq 1$, be the graph property that consists of all graphs whose connected components are cliques of at most s vertices. Furthermore, let Π_∞ be the graph property that consists of all graphs whose connected components of G are cliques (of arbitrary size).



(a) DISJOINT Π_∞-VERTEX DELETION instance (b) The assignment problem

Fig. 3: (a) Data reduction in the disjoint compression routine. The gray vertices in the input instance are deleted by the data reduction rules. The black vertices correspond to a minimum solution X_2' (determined by a solution for the assignment problem, see below). Each circled group of vertices corresponds to an edge in the assignment problem (b). If a circled group of vertices is in X or if no vertex in this group has a neighbor in X, then the corresponding vertex is black, otherwise, it is white. The bold edges show the maximum matching that corresponds to the minimum solution X_2'.

For instance, Π_1, Π_2, and Π_∞ are the properties "being edgeless", "being a graph of maximum degree one", and "being a cluster graph", respectively. As described above, the corresponding sets of minimal forbidden induced subgraphs consist of: P_2 (Π_1), P_3 and K_3 (Π_2), and P_3 (Π_∞). In general, the set of minimal forbidden induced subgraphs of Π_s for $s \geq 2$ consists of P_3 and K_{s+1}. Summarizing, for each property Π_s, $s \geq 1$, and Π_∞, the corresponding set of minimal forbidden induced subgraphs contains a star with at most two leaves (in other words, P_2 or P_3), and these are the only properties whose sets of minimal forbidden induced subgraphs contain a star with at most two leaves.

THEOREM 4.3. DISJOINT Π-VERTEX DELETION *can be solved in polynomial time if* $\Pi = \Pi_s$*, for some* $s \geq 1$*, or if* $\Pi = \Pi_\infty$.

For property Π_1, the disjoint solution X' must contain every endpoint of each edge that has one endpoint in the given solution X and the other endpoint in $V \setminus X$. Hence, the input is a yes-instance if and only if X forms an independent set and $|N_G(X)| < |X|$. This condition can be tested in linear time by scanning through all edges incident to each vertex in X.

LEMMA 4.4. DISJOINT Π_1-VERTEX DELETION *can be solved in* $O(n+m)$ *time,*

DISJOINT Π_∞-VERTEX DELETION is equivalent to the disjoint compression task of CLUSTER VERTEX DELETION, that can be solved in $O(m\sqrt{n} \cdot \log n)$ time [Hüffner et al. 2010]. In order to make the description of the complexity dichotomy self-contained, we provide a very short version of the proof of the following lemma by Hüffner et al. [2010]; this proof exhibits the main technique, which we will adapt for the proof of Lemma 4.6, that shows the polynomial-time solvability for the remaining properties Π_s for $s \geq 2$.

LEMMA 4.5 ([HÜFFNER ET AL. 2010]). DISJOINT Π_∞-VERTEX DELETION *can be solved in* $O(m\sqrt{n} \cdot \log n)$ *time.*

PROOF. Recall that the set of minimal forbidden induced subgraphs corresponding to Π_∞ only contains one element, namely P_3. We describe an algorithm that solves DISJOINT Π_∞-VERTEX DELETION. An example for a DISJOINT Π_∞-VERTEX DELETION instance is shown in Fig. 3a. The algorithm begins by computing all vertices that necessarily have to be in X'; if there exists an induced P_3 in G such that exactly one vertex v of that P_3 is in $R := V \setminus X$ (thus, the remaining two vertices of the P_3 are

in X), then we call v a *necessary vertex*. Obviously, all necessary vertices have to be in X'. Thus, a first data reduction rule computes the set of necessary vertices, adds them to an initially empty set X'_1, and deletes the necessary vertices from the graph G and from the set R. After that, a second data reduction rule deletes connected components that are cliques from G, from R, and from X; the soundness of this rule is obvious. In the following, let (G, X) be the remaining instance after deleting all necessary vertices and isolated cliques. The set X'_1 contains all necessary vertices, thus it remains to compute an optimal solution X'_2 for (G, X); $X' := X'_1 \cup X'_2$ is then a minimum-size solution for our input instance.

The reduced instance (G, k) is much simplified: In each clique of $G[R]$, we can divide the vertices into equivalence classes according to their neighborhood in X; each class then contains either vertices adjacent to all vertices of a particular clique in $G[X]$, or the vertices adjacent to no vertex in X (see Fig. 3a), otherwise, there would be a necessary vertex and the first data reduction rule above would apply. This classification is useful because of the following:

Claim. *If there exists a solution for* DISJOINT Π_∞-VERTEX DELETION*, then in the cluster graph resulting by this solution, each clique in $G[R]$ consists of vertices of at most one equivalence class.*

Proof of Claim. Clearly, inside a clique, it is never useful to delete only some, but not all vertices of an equivalence class, since if that led to a solution, we could always re-add the deleted vertices without introducing new induced P_3's. Further, assume that for a clique C in $G[R]$ the vertices of two equivalence classes are present. Let $u \in C$ and $v \in C$ be a vertex from each equivalence class, respectively. Since u and v are in different equivalence classes, they must have a different neighborhood with respect to the cliques in $G[X]$. Assume without loss of generality that v is adjacent to all vertices of a clique C' in $G[X]$. Since u is in an other equivalence class than v, u is not adjacent to any vertex of C'. Let $w \in C'$. The path uvw forms an induced P_3, contradicting our assumption and showing the claim.

Due to this claim, the remaining task for solving DISJOINT Π_∞-VERTEX DELETION is to assign each clique in $G[R]$ to one of its equivalence classes (corresponding to the preservation of this class, and the deletion of all vertices from the other classes within the clique) or to do nothing (corresponding to the complete deletion of the clique). However, we cannot do this independently for each clique; we must not choose two classes from different cliques in $G[R]$ such that these two classes are adjacent to the same clique in $G[X]$ since that would create an induced P_3. This assignment problem can be modeled as a weighted bipartite matching problem in an auxiliary graph H, where each edge corresponds to a possible choice. The graph H is constructed as follows (see Fig. 3b):

(1) Add a vertex for every clique in $G[R]$ (white vertices).
(2) Add a vertex for every clique in $G[X]$ (black vertices in X).
(3) For a clique C_X in $G[X]$ and a clique C_R in $G[R]$, add an edge between the vertex for C_X and the vertex for C_R if there is an equivalence class in C_R containing a vertex adjacent to a vertex in C_X. This edge corresponds to choosing this class for C_R and one assigns the number of vertices in this class as its weight.
(4) Add a black vertex for each class in a clique C_R that is not adjacent to the cliques in $G[X]$ (black vertices outside X), and connect it to the vertex representing C_R. Again, this edge corresponds to choosing this class for C_R and is weighted with the number of vertices in this class.

Since we only added edges between black and white vertices, H is bipartite. The task is now to find a *maximum-weight bipartite matching*, that is, a set of edges of maxi-

mum weight where no two edges have an endpoint in common. To solve this matching instance, we can use an algorithm for integer-weighted matching [Gabow and Tarjan 1989] with a maximum weight of n (since a class can contain at most n vertices), yielding a running time of $O(m\sqrt{n}\log n)$. The set X_2' can be directly constructed from a maximum matching; it contains all vertices in equivalence classes in $G[R]$ that correspond to edges not chosen by the matching in H (see Fig. 3). If we apply the data reduction rules in their given order, we can execute them in $O(m)$ time. Obviously, the input instance is a yes-instance if and only if $|X_1'| + |X_2'| < |X|$. Thus, we can solve DISJOINT Π_∞-VERTEX DELETION in $O(m\sqrt{n}\log n)$ time. □

It remains to show the polynomial-time solvability for the remaining properties Π_s. The technique is similar.

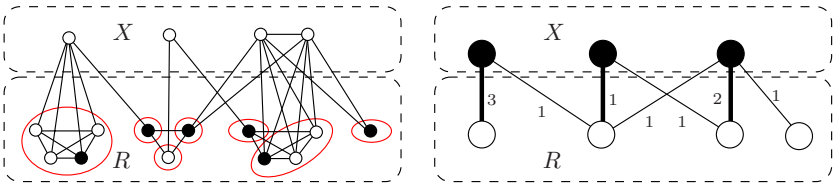
LEMMA 4.6. *For each $s \geq 2$,* DISJOINT Π_s-VERTEX DELETION *can be solved in $O(m\sqrt{n}\log n)$ time.*

PROOF. Let (G, X) be the input instance for DISJOINT Π_s-VERTEX DELETION. Recall that $G[X]$ is a collection of cliques and the set of minimal forbidden induced subgraphs corresponding to Π_s is P_3 and a clique of $s+1$ vertices; hence, every connected component of $G - X$ is a clique of at most s vertices.

We describe an algorithm that finds a minimum-size vertex set X' such that $X \cap X' = \emptyset$ and such that $G - X' \in \Pi_s$, or returns "no-instance". This algorithm is similar to the one for DISJOINT Π_∞-VERTEX DELETION in the proof of Lemma 4.5, but additionally takes into account the forbidden clique of $s+1$ vertices. Moreover, every vertex in $V \setminus X$ that is adjacent to one vertex in a connected component in $G[X]$ is already adjacent to all vertices in this connected component.

The algorithm starts by computing all vertices that necessarily have to be in X': if there exists a forbidden induced subgraph F in G such that exactly one vertex v of F is in $R := V \setminus X$ (thus, all remaining vertices of F are in X), then we call v a *necessary vertex*. Obviously, all necessary vertices have to be in X'. Thus, a first data reduction rule computes the set of necessary vertices, adds them to an initially empty set X_1', and deletes the necessary vertices from G. This can be easily accomplished by first finding and deleting all necessary vertices due to the forbidden P_3 (as in the proof of Lemma 4.5), and then finding and deleting all necessary vertices due to the forbidden clique of $s+1$ vertices (clearly, the neighborhood of every s-vertex clique in $G[X]$ is a set of necessary vertices due to the forbidden clique). Then, consider the connected components C in the graph reduced by the first rule. For each C that is a clique, a second data reduction rule adds $|V(C)| - s$ arbitrary vertices from $V(C) \cap R$ to X_1' if $|V(C)| > s$ (there are always sufficiently many vertices to choose from, since $G[X]$ only contains cliques of size at most s), and deletes C from the graph. This reduction rule is correct, because for a connected component that is a clique of more than s vertices it does not matter which vertices are deleted, as there is no connection to the rest of the graph.

In the following, let (G, X) be the remaining instance after exhaustively applying the two data reduction rules. As in the proof of Lemma 4.5, a minimum-size set X_2', $X_2' \cap X = \emptyset$, containing a vertex of every induced path on three vertices and every clique of $s+1$ vertices, can be obtained with matching techniques. The main difference is that if a clique in $G[R]$ is present in the cluster graph $G - X'$, then it is not necessarily the case that all vertices of that clique are present due to the size constraint s. This size constraint, however, can be encoded in the edge weights of the corresponding assignment problem. The construction of H is the same as in the proof of Lemma 4.5 except for the weight of an edge between the vertex for a clique C_X in $G[X]$ and the vertex for a clique C_R in $G - X$ (assuming that there is an equiva-



(a) DISJOINT Π_4-VERTEX DELETION instance (b) The assignment problem

Fig. 4: (a) Instance (G, X), preprocessed as described in the proof of Lemma 4.6, for $s = 4$. The black vertices are in a minimum solution X_2'. Each group of encircled vertices corresponds to an edge in the assignment problem (b). The edges in the assignment problem are weighted with the number of vertices that do not have to be taken into X_2' if the corresponding edge is chosen to be in the matching. The bold edges show the maximum matching that corresponds to the minimum solution X_2'.

lence class in C_R that contains a vertex adjacent to C_X): the weight of the edge is set to $\min\{s - |V(C_X)|, t\}$, where t is the number of vertices in the corresponding equivalence class in C_R. If the edge is in a maximum matching of H, and if $t \leq s - |V(C_X)|$, then the argument is as in the proof of Lemma 4.5 and no vertex of the corresponding class in C_R is in X_2'; however, if $t > s - |V(C_X)|$, then this still means that the corresponding class of C_R is chosen, but since together with the vertices in C_X there are more than s vertices, one has to add all but $s - |V(C_X)|$ vertices of this class to X_2'. Consider Fig. 4 for an example of such an instance (G, X) and the corresponding assignment problem.

The data reduction rules can be performed in $O(m)$ time, and the matching algorithm needs $O(m\sqrt{n}\log n)$ time (see the proof of Lemma 4.5). □

4.2. NP-Hardness Framework and Simple Proofs

Lewis and Yannakakis [1980] showed that Π-VERTEX DELETION for any non-trivial hereditary property Π is NP-complete. Due to the similarity of Π-VERTEX DELETION to DISJOINT Π-VERTEX DELETION, in some simple cases we can adapt the framework from Lewis and Yannakakis [1980].[5] This section is mainly devoted to this framework and how it can be modified to partially address the complexity of DISJOINT Π-VERTEX DELETION.

There are cases, however, where our adaption fails; this happens when there is a star with at least three leaves among the family $\mathcal{H}$ of minimal forbidden induced subgraphs corresponding to Π. For this case, we have to devise other NP-hardness proofs (if there is a star with at most two leaves, then the problem is polynomial-time solvable). Summarizing, we have to distinguish the following three cases (recall that each graph in $\mathcal{H}$ is connected, because Π is determined by the components):

(1) $\mathcal{H}$ does not contain a star (NP-hard, this section, Theorem 4.7), and
(2) $\mathcal{H}$ contains a star with at least three leaves (NP-hard, Section 4.3, Theorem 4.12), and
(3) $\mathcal{H}$ contains a star with at most two leaves (that is, P_2 or P_3; polynomial-time solvable, Section 4.1, Theorem 4.3).

[5]As made explicit in Lewis and Yannakakis' paper [Lewis and Yannakakis 1980], the parts of it we are referring to in our work have been contributed by Yannakakis. That is why we refer to it in the following as "the framework of Yannakakis".

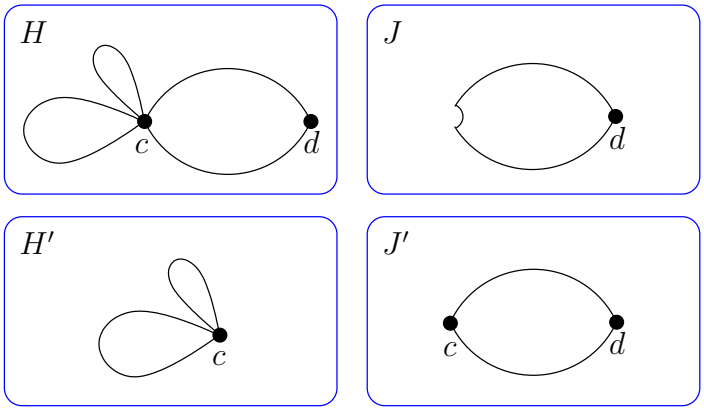


Fig. 5: Connected graph H with cut-vertex c and some vertex d in a largest connected component J of $H - c$. Moreover, the graphs $H' = H - V(J)$ and $J' = H[V(J) \cup \{c\}]$ are illustrated.

The main result of this section covers all cases that can be proven by adapting the framework of Yannakakis.

THEOREM 4.7. *Let Π be a non-trivial hereditary property that is determined by the components and let $\mathcal{H}$ be the corresponding set of all minimal forbidden induced subgraphs. If $\mathcal{H}$ contains no star, then* DISJOINT Π-VERTEX DELETION *is NP-hard.*

4.2.1. The Framework of Yannakakis. In the following, we briefly describe the reduction by Yannakakis [Lewis and Yannakakis 1980], showing that any vertex deletion problem for a non-trivial hereditary graph property is NP-hard. Since the hereditary graph properties we consider are assumed to be determined by the components, we present a variant that is restricted to such properties, that is, the forbidden induced subgraphs are connected.

Preliminaries. Let $\mathcal{H}$ be the set of minimal forbidden induced subgraphs that correspond to the non-trivial hereditary property Π that is determined by its components. In the following, we call a vertex subset X such that $G - X \in \Pi$ a *$\mathcal{H}$-obstruction set* in G (since it obstructs every forbidden induced subgraph in $\mathcal{H}$). An important concept for the framework is the notion of α-sequences [Lewis and Yannakakis 1980].

Definition 4.8 (*(α-sequence)*). For a connected graph $H \in \mathcal{H}$, if H is 1-connected, then take a cut-vertex c; otherwise, then let c be an arbitrary vertex (in this case, $H - c$ has just one connected component). Sorting the connected components of $H - c$ decreasingly with respect to their sizes gives a sequence $\alpha = (n_1, \ldots, n_i)$, where $n_1 \geq \ldots \geq n_i$. The sequence depends on the choice of c. The *α-sequence* of H, $\alpha(H)$, is a sequence which is lexicographically smallest among all such sequences α.

Let $H \in \mathcal{H}$ be a graph with the lexicographically smallest α-sequence among all graphs in $\mathcal{H}$, and let $c \in V(H)$ be a vertex that yields such a lexicographically smallest α-sequence. Note that every proper induced subgraph of H has a lexicographically smaller α-sequence than H. Since Π is satisfied by all edge-less graphs (Π is determined by the components, thus it contains all edge-less graphs), the connected graph H must contain at least two vertices, thus a largest component J of $H - c$ contains at least one vertex. Let d be an arbitrary vertex in J, and let H' be the graph resulting by re-

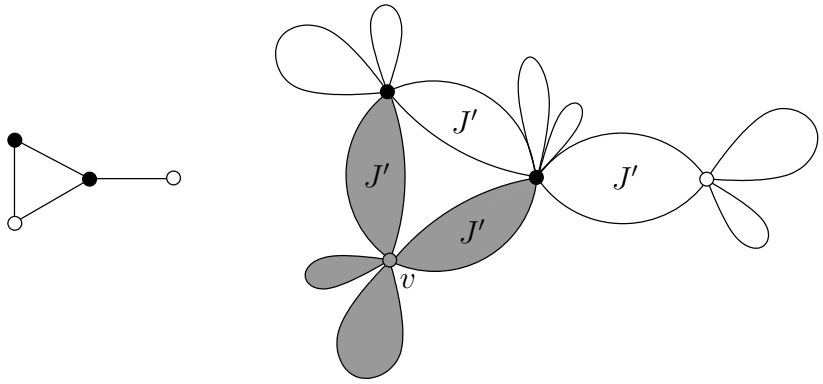


Fig. 6: Example for a reduction from VERTEX COVER. Left: VERTEX COVER instance with a vertex cover A (black vertices). Right: Corresponding Π-VERTEX DELETION instance G' constructed by the framework of Yannakakis, together with a solution X (black vertices) corresponding to the vertex cover A. The gray area marks a connected component in $G'-X$ corresponding to case (2) in the correctness proof with cut-vertex v.

moving all vertices in J from H, and let J' be the subgraph of H induced by $V(J) \cup \{c\}$. See Fig. 5 for an example.

Reduction. The reduction by Yannakakis [Lewis and Yannakakis 1980] from the NP-complete VERTEX COVER problem works as follows. Let G be an instance of VERTEX COVER. For every vertex v in G create a copy of H' and identify c and v. Replace every edge $\{u, v\}$ in G by a copy of J', identifying c with u and d with v. Let G' be the resulting graph. See Fig. 6 for an example of the reduction.

Correctness. The graph G has a size-k vertex cover if and only if G' has a size-k vertex set that obstructs the set of forbidden induced subgraphs $\mathcal{H}$ in G':

($\Rightarrow$) If A is a vertex cover of G, then $X' := A$ also obstructs all graphs in $\mathcal{H}$: Every connected component of $G' - X'$ is either

(1) a connected component of a copy of $H' - c$ or
(2) a copy of H' together with several copies of J', each with c or d deleted.

Let C be a connected component of $G' - X'$. In case (1), $\alpha(H' - c)$ is lexicographically smaller than $\alpha(H)$ since $H' - c$ is a subgraph of H. In case (2), the copy of H' and the copies of J' intersect exactly in one vertex v of $V(G)$. Hence, v is a cut-vertex and the components of $C - v$ can be divided into a copy of $H' - c$ and several copies of J with one vertex deleted. Since the latter type of components has less than $|V(J)|$ vertices, the cut-vertex v gives an α-sequence for C which is lexicographically smaller than the α-sequence of H (see also Fig. 6). As a consequence, the connected components in $G' - X'$ have a smaller α-sequence than H, and because H is a forbidden induced subgraph with lexicographically smallest α-sequence, these connected components do not contain forbidden induced subgraphs.

($\Leftarrow$) If X' is a solution for Π-VERTEX DELETION, then one can determine a vertex cover A for G as follows: for each $w \in X'$, if w is in a copy of H' (possibly $w \in V(G)$), then add vertex c of that copy of H' to A, and if w is in a copy of J' (where $w \notin V(G)$), then add vertex c of that copy of J' to A. Obviously, $|A| \leq |X'|$. Suppose that there exists an edge $\{u, v\}$ in $G - A$. Then, by construction of A, X' neither contains any vertex from the two copies of H' corresponding to the vertices u and v nor from the copy of J' that

replaced the edge $\{u, v\}$ in the construction of G'. Hence $G' - X'$ contains a copy of H, a contradiction. Therefore, A is a vertex cover for G.

Limitations. In order to modify the reduction from Section 4.2.1 for DISJOINT Π-VERTEX DELETION, the main difficulty lies in the construction of an old solution X such that X does not prevent the "equivalence argument" of the reduction. In other words, one has to construct an $\mathcal{H}$-obstruction set X in G' with the restriction that X does not contain any vertex from $V(G)$. In some cases the set X can be constructed in a straight-forward way and, thus, the same argument as in Section 4.2.1 can be used.

However, if J' is a clique and some graph H of $\mathcal{H}$ is contained in G, then H is also contained in G' and can only be obstructed by deleting vertices from $V(G)$ and there is no obvious construction of such a set X. The worst case of this problem occurs when we have a star as the graph H in $\mathcal{H}$ that has the smallest α-sequence. In this case, J' is simply an edge and each connected component of $H - c$ is an isolated vertex (c is the center of the star). Thus, the vertex d has to be one of these vertices, and G and, therefore, G' might contain a forbidden induced subgraph with lexicographically higher α-sequence than H. This induced subgraph cannot be obstructed by a set X with $X \cap V(G) = \emptyset$. To cope with this problem, we distinguish cases based on whether H contains a star or not.

If there is no star in H, then we slightly modify the reduction in Section 4.2.1, namely, reducing from VERTEX COVER on K_3-free graphs, and using the graph with the smallest α-sequence among all K_3-free graphs in $\mathcal{H}$. This case is accomplished in two steps, first the simpler case that all graphs in $\mathcal{H}$ contain K_3 as subgraph (Lemma 4.10) and, then, the case that some graph in $\mathcal{H}$ does not contain K_3 as subgraph (Lemma 4.11).

If there is a star in H, then we again distinguish two cases, H containing a large star, that is, a star with at least four leaves, or not. If yes, then we again reduce from a special NP-hard version of VERTEX COVER, namely, VERTEX COVER on graphs with maximum degree three. Note that the upper bound on the maximum degree excludes the existence of large stars in G, which makes the argument in Section 4.2.1 (using an adapted gadget construction) work again (Lemma 4.13). Thus, it remains the case that H contains a star of exactly three leaves. Note that the case of stars with at most two leaves is polynomial-time solvable and has been considered in Section 4.1. In the case of stars with three leaves, we could not find a way to adapt the framework of Yannakakis, since any reduction from an NP-hard variant of VERTEX COVER did not avoid leaving some stars with three leaves in the new instance. Therefore, we apply a completely new reduction from 3-CNF-SAT which has two variants, differing in how they deal with long paths in $\mathcal{H}$ (Lemmas 4.14 and 4.15).

4.2.2. Proofs Based on the Reduction Framework of Yannakakis. First, we introduce a new variant of DISJOINT Π-VERTEX DELETION, where the size of the new solution is given as a parameter, and show how it can be reduced to DISJOINT Π-VERTEX DELETION. In all proofs that follow, we show the NP-hardness of that variant, because all hardness proofs need a *size gadget* that is employed in the reduction from the variant to DISJOINT Π-VERTEX DELETION.

SIZE DISJOINT Π-VERTEX DELETION
Input: An undirected graph $G = (V, E)$, a parameter k, and a vertex subset $X \subseteq V$ such that $G[X] \in \Pi$, $G - X \in \Pi$, and X is inclusion-minimal under this property.
Question: Is there a vertex subset $X' \subseteq V$ with $|X'| \leq k$ such that $X \cap X' = \emptyset$ and $G - X' \in \Pi$?

LEMMA 4.9. SIZE DISJOINT Π-VERTEX DELETION *can be reduced to* DISJOINT Π-VERTEX DELETION *in polynomial time.*

PROOF. Let (G, X, k) be an instance of SIZE DISJOINT Π-VERTEX DELETION. We construct an instance $(\hat{G}, \hat{X})$ of DISJOINT Π-VERTEX DELETION as follows (recall that DISJOINT Π-VERTEX DELETION asks for a solution $\hat{X}'$ such that $|\hat{X}'| < |\hat{X}|$). First, suppose that $k = |X| - 1$. Then, obviously, (G, X, k) is a yes-instance for SIZE DISJOINT Π-VERTEX DELETION if and only if $(\hat{G}, \hat{X}) := (G, X)$ is a yes-instance for DISJOINT Π-VERTEX DELETION. It remains to deal with the cases $k < |X| - 1$ and $k > |X| - 1$, where we employ a padding trick using a *size gadget*. The basic idea is to enforce that, in the constructed graph G' containing the size gadget, a certain number of vertices of the new solution have to be in the size gadget, such that there are exactly k vertices left to obstruct all forbidden induced subgraphs in G.

The size gadget for $k < |X| - 1$: In this case, informally speaking, we have to force that only k vertices out of the $|X| - 1$ available vertices can be used to obstruct all forbidden induced subgraphs. Let H, c, J, J', and d be defined as in the reduction scheme (see Fig. 5). We create a new graph $\hat{G}$ by using a copy of G and adding a padding gadget C constructed as follows. Add a new vertex w and $|X| - k$ copies of H, identify the vertex d of each newly added copy of H with w, and let $\hat{X} := X \cup \{w\}$. The gadget C is obviously connected and w is a cut-vertex in C. The vertex w obstructs all forbidden induced subgraphs in C, because deleting w (and, thus, d) from each copy of H in C leaves a graph with lexicographically smaller α-sequence (witnessed by c in each copy of H). Hence, $\hat{X}$ is a minimal $\mathcal{H}$-obstruction set for $\hat{G}$.

An $\mathcal{H}$-obstruction set $\hat{X}'$ for $\hat{G}$ with $\hat{X}' \cap \hat{X} = \emptyset$ must contain at least one vertex in each copy of H in C, thus $\hat{X}'$ must contain at least $|X| - k$ vertices of C; putting into $\hat{X}'$ the vertex c of each copy of H in C obstructs every forbidden induced subgraph in H: every connected component of $C - \hat{X}'$ either is a connected component of a copy of $H - c$ or consists of $|X| - k$ copies of J that pairwise overlap in the vertex w. In the latter case, w is a cut-vertex witnessing that each remaining connected component has size smaller than J, yielding a lexicographically smaller α-sequence. This shows that $\hat{X}'$, in order to obstruct all forbidden induced subgraphs in C, needs to contain at least $|X| - k$ vertices of C. Recall that one demands that $|\hat{X}'| < |\hat{X}|$. Since $\hat{X} = X \cup \{w\}$, there remain at most $|\hat{X}| - |X| + k - 1 = k$ vertices to obstruct all forbidden induced subgraphs in $G = \hat{G} - V(C)$. Hence, (G, X, k) is a yes-instance for SIZE DISJOINT Π-VERTEX DELETION if and only if $(\hat{G}, \hat{X})$ is a yes-instance for DISJOINT Π-VERTEX DELETION.

The size gadget for $k > |X| - 1$: In this case, we construct $\hat{G}$ in the same manner using a gadget C with $k - |X| + 2$ copies of H overlapping in vertex w and let $\hat{X}$ be the union of X and the vertex c of each copy of H. Then, $|\hat{X}| = k + 2$. A new solution $\hat{X}'$ of size at most $|\hat{X}| - 1 = k + 1$ with $\hat{X}' \cap \hat{X} = \emptyset$ for $\hat{G}$ can obstruct all forbidden induced subgraphs in C with the vertex w, and there are k vertices left to obstruct all forbidden induced subgraphs in $G = \hat{G} - V(C)$. Hence, (G, X, k) is a yes-instance for SIZE DISJOINT Π-VERTEX DELETION if and only if $(\hat{G}, \hat{X})$ is a yes-instance for DISJOINT Π-VERTEX DELETION.

Obviously, in both cases the size gadget C can be constructed in polynomial time. □

Recall that, for the following two proofs, we assume that the set $\mathcal{H}$ of forbidden induced subgraphs corresponding to Π contains no star. We have to distinguish between the cases that

(1) all forbidden induced subgraphs in $\mathcal{H}$ contain K_3 (see Lemma 4.10), and that
(2) not all forbidden induced subgraphs in $\mathcal{H}$ contain K_3 (see Lemma 4.11).

LEMMA 4.10. *If the set $\mathcal{H}$ of minimal forbidden induced subgraphs corresponding to Π only consists of graphs that contain K_3, then* DISJOINT Π-VERTEX DELETION *is NP-hard.*

PROOF. The proof is by reduction from the NP-complete VERTEX COVER problem on K_3-free graphs [Garey and Johnson 1979] to SIZE DISJOINT Π-VERTEX DELETION. Let (G, k) be an instance of VERTEX COVER, where G is K_3-free. First, construct a graph G' using the reduction scheme of Yannakakis. Greedily compute a minimal $\mathcal{H}$-obstruction set X for G' such that $X \cap V(G) = \emptyset$. Such a set X always exists, since G is K_3-free and, therefore, does not contain any forbidden induced subgraph. By these arguments and the reduction scheme, G has a size-k vertex cover if and only if (G', X, k) is a yes-instance for SIZE DISJOINT Π-VERTEX DELETION. The NP-hardness of DISJOINT Π-VERTEX DELETION then follows from Lemma 4.9. □

In the following, assume that not all forbidden induced subgraphs contain K_3.

LEMMA 4.11. *If the set $\mathcal{H}$ of minimal forbidden induced subgraphs corresponding to Π contains no stars, but does contain other graphs that do not contain K_3, then* DISJOINT Π-VERTEX DELETION *is NP-hard.*

PROOF. The reduction from the NP-complete VERTEX COVER on K_3-free graphs is very similar to the one for Lemma 4.10. The difference is that G might now contain a forbidden subgraph in $\mathcal{H}$, and we have to show that we can greedily compute a minimal $\mathcal{H}$-obstruction set X for G' such that $X \cap V(G) = \emptyset$ (as in the proof of Lemma 4.10). To this end, we first set the encoding forbidden subgraph H used in the reduction scheme by Yannakakis equal to a K_3-free subgraph in $\mathcal{H}$ which has the lexicographically smallest α-sequence among all K_3-free graphs in $\mathcal{H}$. Second, by setting H in this way, a largest connected component in $H - c$ contains at least one edge, due to the fact that H is not a star. Moreover, since H does not contain K_3, at most one endpoint of this edge is adjacent to c. Then, we select an endpoint of this edge that is not adjacent to c as the vertex d used in the construction of G'. Now, we can observe that in the resulting G' the vertices in $V(G)$ induce an independent set. Therefore, removing all vertices $V(G') \setminus V(G)$ gives an $\mathcal{H}$-obstruction set for G' and we can easily compute an inclusion-minimal solution X for G' with $X \cap V(G) = \emptyset$. Since the graphs G and H are K_3-free and so is G', forbidden induced subgraphs with a smaller α-sequence than H, that contain K_3, do not have to be considered. The correctness of the reduction for this case follows from the same arguments as in the proof of Lemma 4.10. □

4.3. Refined Reduction Strategies

Here, we present NP-hardness proofs for the cases where we have a star with at least three leaves as a forbidden subgraph. The main result of this section is as follows.

THEOREM 4.12. *Let Π be a non-trivial hereditary graph property that is determined by the components and let $\mathcal{H}$ be the corresponding set of all minimal forbidden induced subgraphs. If $\mathcal{H}$ contains a star with at least three leaves, then* DISJOINT Π-VERTEX DELETION *is NP-hard.*

Note that a star has a smaller α-sequence than any other forbidden induced subgraph that is not a star, and there is only one star in $\mathcal{H}$, since the graphs in $\mathcal{H}$ are inclusion-minimal. Therefore, if $\mathcal{H}$ contains a star, then the graph with smallest α-sequence is necessarily the star in $\mathcal{H}$. Let H be the star in $\mathcal{H}$.

The proof of Theorem 4.12 is based on the following case distinctions.

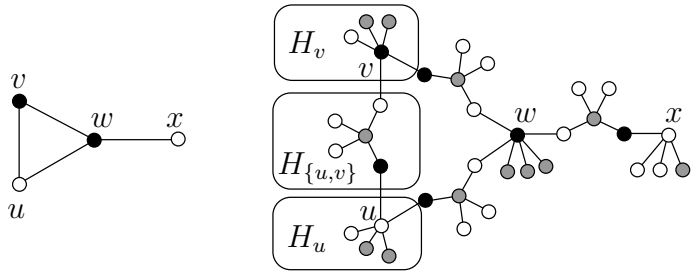


Fig. 7: Example for the reduction in the proof of Lemma 4.13 if H is a star with four leaves. Left: VERTEX COVER instance with a vertex cover C (black vertices). Right: Corresponding SIZE DISJOINT Π-VERTEX DELETION instance with the given solution X (gray vertices) and a solution X' corresponding to the vertex cover C (black vertices). For illustration, the gadgets H_u, H_v, and $H_{\{u,v\}}$ are labeled.

(1) H is a star with at least four leaves (Lemma 4.13).
(2) H is a star with three leaves.
    (a) $\mathcal{H}$ contains P_4 (Lemma 4.14).
    (b) $\mathcal{H}$ does not contain P_4 (Lemma 4.15).

LEMMA 4.13. *If the set $\mathcal{H}$ of minimal forbidden induced subgraphs corresponding to property Π contains a star H with at least four leaves, then* DISJOINT Π-VERTEX DELETION *is NP-hard.*

PROOF. The proof is by reduction from the NP-complete VERTEX COVER on graphs of maximum degree three [Garey and Johnson 1979] to SIZE DISJOINT Π-VERTEX DELETION. Let (G, k) be a corresponding input instance of VERTEX COVER. Let $l \geq 4$ be the number of leaves of H. An example of the following construction is given in Fig. 7. Starting with an empty graph G' and an empty solution set X, for each vertex v in G, create a copy H_v of a star with $l-1$ leaves (vertex gadget), identify its center vertex with v, and add any $\deg_G(v)$ of H_v's leaves to X. For each edge $\{u, v\}$ in G, create a copy $H_{\{u,v\}}$ of H (edge gadget), add $H_{\{u,v\}}$'s center vertex to X, select two arbitrary leaves u', v' of $H_{\{u,v\}}$ and insert the edges $\{u, u'\}$ and $\{v, v'\}$.

Obviously, the graph $G' - X$ only contains connected components that are either isomorphic to a star with $l-1$ leaves or isolated vertices. An isolated vertex as well as a star with $l-1$ leaves both have lexicographically smaller α-sequences than H. Hence, since the star H with at least four leaves has a lexicographically smallest α-sequence among all graphs in $\mathcal{H}$, $G' - X \in \Pi$. Moreover, X is minimal, because $G - (X \setminus \{v\})$ does contain a star with l leaves for any $v \in X$. It remains to show that there is a size-k vertex cover for G if and only if there is a vertex set X', $X' \cap X = \emptyset$, of size $k' := k + |E(G)|$, obstructing all forbidden induced subgraphs in G'. In other words, (G, k) is a yes-instance for VERTEX COVER if and only if (G', X, k') is a yes-instance for SIZE DISJOINT Π-VERTEX DELETION, which shows that SIZE DISJOINT Π-VERTEX DELETION is NP-hard. The NP-hardness of DISJOINT Π-VERTEX DELETION then follows from Lemma 4.9.

($\Rightarrow$) Let C be a size-k vertex cover for G. The set X' is constructed as follows. Beginning with $X' := C$, for each copy $H_{\{u,v\}}$ with the two leaves u', v' (see construction), if $u \in C$ and $v \notin C$, then add v' to X', if $u \notin C$ and $v \in C$, then add u' to X', and if $u \in C$ and $v \in C$, then add either u' or v' to X'. Clearly, $|X'| = k + |E(G)|$, since X' contains a vertex for each edge in G and k vertices from the size-k vertex cover C (also see Fig. 7).

The connected components in $G' - X'$ are either isolated vertices or stars with $l-1$ leaves: the leaves of the components H_v are isolated vertices in $G' - X'$ if $v \in X'$, and if $v \notin X'$, then its neighbors on the adjacent edge-gadgets are in X' by construction of X'. Hence, the vertex-gadget H_v, a star with $l-1$ leaves, forms a connected component in $G' - X'$. Concerning an edge-gadget $H_{\{u,v\}}$, we observe that exactly one of the two vertices u', v' is in X'. Without loss of generality assume that $v' \in X'$ and $u' \notin X'$. Then, by the construction of X', $u \in X'$. Thus, $H_{\{u,v\}} - v'$, a star with $l-1$ leaves, is a connected component in $G' - X'$. Since H is the forbidden subgraph with the lexicographically smallest α-sequence, X' obstructs all forbidden induced subgraphs in G'.

($\Leftarrow$) Let X' be a size-$(k + |E(G)|)$ vertex set that obstructs every forbidden induced subgraph in G'. We may assume that X' does not contain any degree-one vertex of G' (since a degree-one vertex in X' of a vertex gadget could be simply replaced by its neighbor, and a degree-one vertex in X' of an edge gadget $H_{\{u,v\}}$ could be simply replaced by either u' or v'). Observe that for each edge-gadget $H_{\{u,v\}}$ at least one of u', v' must be in X', since $H_{\{u,v\}}$ is a forbidden induced subgraph. Hence, X' contains at least $|E(G)|$ vertices of the edge gadgets. Let $\{u, v\}$ be an edge in G. We distinguish two cases:

(1) If only one of u', v' of $H_{\{u,v\}}$ is in X', then u or v is in X': assume without loss of generality that $u' \in X'$ and $v' \notin X'$. Then, v' together with the vertices of H_v induce a star with l leaves (which is forbidden) in G', and since we assumed that the leaves of H_v are not in X', $v \in X'$.
(2) If both u' and v' are in X', and $u, v \notin X'$, then we can simply remove u' from X' and add u instead. After that, X' still obstructs all forbidden induced subgraphs, and case (1) applies.

Hence, for each edge $\{u, v\}$ in G at least one of its endpoints is in X'. In other words, $X' \cap V(G)$ is a vertex cover for G. Since X' contains at least $|E(G)|$ vertices of the edge gadgets, $X' \cap V(G)$ has size at most k. □

Next, we show the NP-hardness of the case that the forbidden subgraph with the lexicographically smallest α-sequence is a star with three leaves. In this case, a reduction from VERTEX COVER seems less promising, since the VERTEX COVER instance we reduce from contains vertices of degree three and therefore copies of the forbidden induced star with three leaves, which would have to be obstructed by the solution X in the reduction. This makes it difficult to translate a solution for SIZE DISJOINT Π-VERTEX DELETION back to a vertex cover in the VERTEX COVER instance.

We reduce from 3-CNF-SAT. Our proofs rely heavily on the simple structure of a star. First, we consider the case that the path on four vertices is also forbidden.

LEMMA 4.14. *If the set $\mathcal{H}$ of minimal forbidden induced subgraphs corresponding to property Π contains a star H with three leaves and $\mathcal{H}$ also contains the path on four vertices, then* DISJOINT Π-VERTEX DELETION *is NP-hard.*

PROOF. The proof is by reduction from 3-CNF-SAT to SIZE DISJOINT Π-VERTEX DELETION. We assume without loss of generality that each variable appears in each clause at most once. Let $F = c_1 \wedge \cdots \wedge c_q$ be a 3-CNF formula over a variable set $Y = \{y_1, \ldots, y_p\}$. We denote the kth literal in clause c_j by l_j^k, for $1 \leq k \leq 3$. An example of the following construction is given in Fig. 8. Starting with an empty graph G and $X := \emptyset$, construct an instance (G, X) for DISJOINT Π-VERTEX DELETION as follows. For each variable y_i, introduce a cycle Y_i of $4q$ vertices (variable gadget), add every second vertex on Y_i to X, and label all the other vertices on the cycle alternately with "+" and "−". For each clause c_j, add a star C_j with three leaves (clause gadget) and add its center

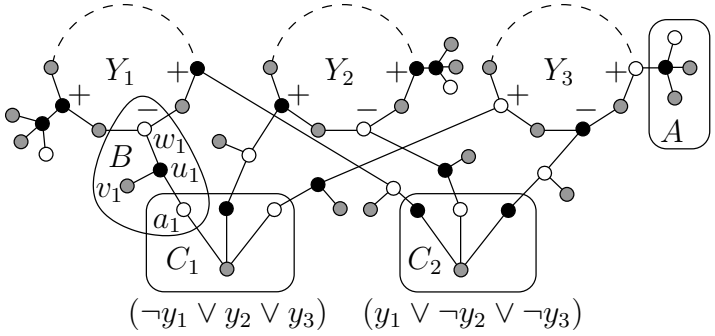


Fig. 8: Example for the reduction in the proof of Lemma 4.14 for the 3-CNF-SAT formula $(\neg y_1 \vee y_2 \vee y_3) \wedge (y_1 \vee \neg y_2 \vee \neg y_3)$. For illustration, one minimality gadget is labeled with A and one connection gadget is labeled with B. The vertices of the connection gadget B are named according to the definitions of a_k, u_k, v_k, and w_k in the proof of Lemma 4.14 for $k = 1$. The vertices in the given solution X are gray, the vertices in the disjoint solution X', corresponding to the satisfying truth assignment $y_1 =$ true, $y_2 =$ true, $y_3 =$ false, are black.

vertex to X. Each of the three leaves of C_j corresponds to a literal in c_j, and each leaf is connected to a variable gadget as follows. Suppose that l_j^k is a literal y_i or $\neg y_i$, and let a_k be the leaf of C_j corresponding to l_j^k. Add a star with three leaves (connection gadget), identify one leaf with a_i, identify another leaf with an *unused vertex*[6] on Y_i with label "+" if l_j^k is positive and with an unused vertex on Y_i with label "−" if l_j^k is negative, and add the remaining leaf to X. Finally, for each remaining unused vertex v with label "+" or "−" in G, add a star with three leaves (minimality gadget), add two of its leaves to X, and add an edge connecting the center of the star with v. This completes the construction.

Obviously, $G - X$ only contains paths on three vertices as connected components (cf. Fig. 8), that is, $G - X \in \Pi$. Moreover, X is minimal, that is, for any $v \in X$, $G - (X \setminus \{v\})$ does not satisfy Π. Let r be the number of minimality gadgets. We show that formula F has a satisfying truth assignment if and only if there exists a size-$(r + 3pq + 3q)$ set X', $X' \cap X = \emptyset$, that obstructs all forbidden induced subgraphs in G. In other words, F has a satisfying truth assignment if and only if $(G, X, r+3pq+3q)$ is a yes-instance of SIZE DISJOINT Π-VERTEX DELETION. The NP-hardness of DISJOINT Π-VERTEX DELETION then follows from Lemma 4.9.

($\Rightarrow$) Assume that a satisfying truth assignment for F is given. Based on this truth assignment, we construct the disjoint solution X', beginning with $X' := \emptyset$, as follows. For each variable y_i, $1 \leq i \leq p$, if $y_i =$ true, then add every vertex on Y_i with label "+" to X', and if $y_i =$ false, then add every vertex on Y_i with label "−" to X'. Add the center vertex of each minimality gadget to X'. For each literal l_j^k of each clause c_j, if $l_j^k =$ true, then add a_k of the corresponding clause gadget C_j to X', and if $l_j^k =$ false, then add the center of the corresponding connection gadget (which is adjacent to a_k) to X'. Clearly, $|X'| = r + 3pq + 3q$. The connected components of $G - X'$ are either isolated vertices or paths on at most three vertices (thus, $G - X' \in \Pi$): the set X' contains the center of each minimality gadget, hence the leaves of the minimality gadgets are

[6]This means that no vertex of another connection gadget has been identified with this vertex on Y_i, that is, it is of degree two.

isolated vertices in $G - X'$. Concerning a variable gadget Y_i, observe that every fourth vertex on the cycle is in X', and, if a vertex labeled with "+" or "−" is not in X', then its neighbor outside of Y_i (which belongs either to a minimality gadget or to a connection gadget) is in X'. Thus, in $G - X'$, the remaining vertices of Y_i induce connected components that are paths on three vertices. For a connection gadget, observe that either the center vertex is in X' or two of its leaves (which are identified with vertices on other gadgets) are in X', so there remains either an isolated vertex or a single edge in $G - X'$. Concerning a clause gadget C_j, if there is a satisfying truth assignment, then at least one literal is true; therefore, at least one leaf of C_j is in X'. If a leaf is not in X', then its neighbor on the corresponding connection gadget is in X'. Hence, the connected component in $G - X'$ that includes the center of C_j is either a path on three vertices, a single edge, or an isolated vertex (depending on how many literals of c_j are true).

($\Leftarrow$) Let X', $X' \cap X = \emptyset$, be a size-$(r + 3pq + 3q)$ vertex set that obstructs every forbidden induced subgraph in G. We may assume that X' does not contain any degree-one vertex in G (since a degree-one vertex in X' could simply be replaced by its neighbor). Recall that the set of minimal forbidden induced subgraphs contains the star with three leaves and the path on four vertices. Each minimality gadget is a star with three leaves, and since we assumed that no degree-one vertex is in X', its center vertex must be in X'. Hence, X' contains exactly r vertices of the minimality gadgets. Since P_4s are forbidden, at least every fourth vertex on the cycle of each variable gadget has to be in X'. However, we will see that X' contains exactly three vertices for each clause (thus, $3q$ vertices for all clauses), and these vertices cannot be vertices on any variable gadget. Therefore, for each variable gadget Y_i, the set X' must contain *exactly* every fourth vertex of Y_i (in order to obtain a total number of $3qp$ vertices in X' for all p variable gadgets). Thus X' either contains all vertices labeled "+" or all vertices labeled "−". If X' contains all vertices labeled "+", then we set $y_i :=$ true. If X' contains all vertices labeled "−", then we set $y_i :=$ false. It remains to show that the assignment defined in this way is a satisfying truth assignment for the formula F.

For a clause gadget C_j, and for each leaf a_k of C_j corresponding to literal l_j^k, let u_k be the center of the corresponding connection gadget, v_k be the degree-one neighbor of u_k, and w_k be the neighbor of u_k on the variable gadget Y_i, for some $1 \leq i \leq p$ (cf. Fig. 8). There is a P_4 containing the center of C_j, together with a_k, u_k, and v_k. Since the center of C_j is in X, the set X' has to contain at least three vertices to obstruct the three P_4s corresponding to C_j (one for each leaf). Thus, for all clauses, there are at least $3q$ vertices in X' that obstruct these P_4s. In total, X' contains $r + 3pq + 3q$ vertices. Therefore, there are *exactly* $3q$ vertices in X' that obstruct these P_4s. Thus, for a clause gadget C_j, for each leaf a_k, either $a_k \in X'$ or $u_k \in X'$. Which case applies depends on which vertices from Y_i are in X': if $w_k \notin X'$, then w_k together with u_k and its two neighbors on Y_i induce a star with three leaves, thus $u_k \in X'$. If $w_k \in X'$, then either $a_k \in X'$ or $u_k \in X'$. If $w_k \in X'$ and $u_k \in X'$, however, then one can simply remove u_k from X' and add a_k instead. After that, X' still obstructs all forbidden induced subgraphs. Since X' obstructs all forbidden induced subgraphs, at least one leaf a_k of C_j must be in X', which implies that $w_k \in X'$. Let Y_i be the variable gadget that contains w_k. If w_k has label "+", then $y_i =$ true by the definition of the assignment, and by construction $l_j^k = y_i$ is a positive literal, hence c_j is satisfied. If w_k has label "−", then $y_i =$ false, and, by construction, $l_j^k = \neg y_i$ is a negative literal, hence c_j is satisfied. Summarizing, for every clause there is at least one true literal and thus the constructed truth assignment satisfies F. □

Finally, we consider the case that the path on four vertices is not forbidden.

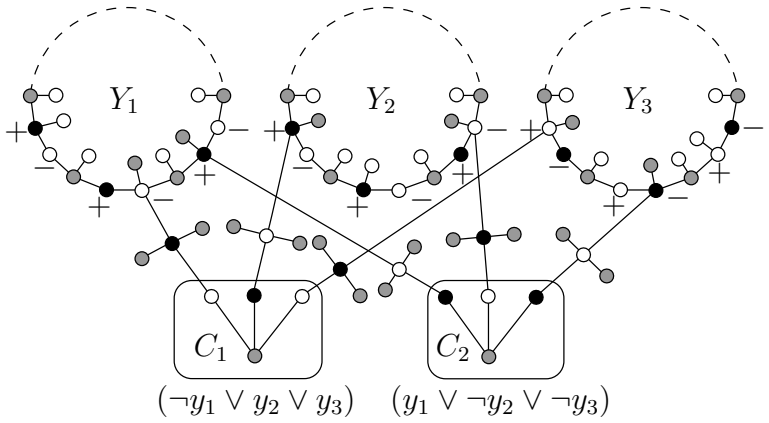


Fig. 9: Example for the reduction in the proof of Lemma 4.15 for the 3-CNF-SAT formula $(\neg y_1 \vee y_2 \vee y_3) \wedge (y_1 \vee \neg y_2 \vee \neg y_3)$. The vertices in the given solution X are gray, the vertices in the disjoint solution X' corresponding to the satisfying truth assignment $y_1 =$ true, $y_2 =$ true, $y_3 =$ false, are black.

LEMMA 4.15. *If the set $\mathcal{H}$ of minimal forbidden induced subgraphs corresponding to property Π contains a star H with three leaves and $\mathcal{H}$ does not contain the path on four vertices, then* DISJOINT Π-VERTEX DELETION *is NP-hard.*

PROOF. As for Lemma 4.14, the proof is by reduction from 3-CNF-SAT to SIZE DISJOINT Π-VERTEX DELETION, and we use the same notation for the 3-CNF-SAT formula as employed there. The proof principle is similar, but the gadgets differ. In the following, we only describe the particularities of this construction and omit straightforward details that can directly be adapted from the proof of Lemma 4.14. An example of the construction is given in Fig. 9. The basic structure of a variable gadget is a cycle of $3q$ vertices (in the following, further vertices and edges will be added to each such cycle). Add every third vertex on that cycle to X, and for each such vertex $v \in X$ add a new vertex and make it adjacent to v. Then, label the remaining vertices on the cycle, that is, vertices on the cycle that are not in X, alternately with "$+$" and "$-$". For each clause c_j introduce a star with three leaves C_j (clause gadget). Each of the three leaves of C_j corresponds to a literal in c_j. As in the proof of Lemma 4.14, we connect the leaves of C_j with vertices labeled with "$+$" or "$-$" on the corresponding variable gadgets, depending on whether the corresponding literal is positive or negative, respectively. Herein, for a pair of adjacent labeled vertices, one with label "$+$" and one with label "$-$", at most one of them is connected to a clause gadget. The connection gadget is a star with four leaves, one leaf is identified with a leaf of a clause gadget, one leaf is identified with a vertex labeled "$+$" or "$-$" on a variable gadget, and the remaining two leaves are added to X. After adding all connection gadgets, for each pair u, v of adjacent labeled vertices, one with label "$+$" (say, u) and one with label "$-$" (say, v), if both u and v have not been connected to a clause gadget, add a new vertex and make it adjacent to u. Moreover, for each labeled vertex u on a variable gadget that has been connected to a clause gadget, add a new vertex, add it to X, and make it adjacent to u.

For the resulting graph G, observe that $G - X$ contains only isolated vertices, paths on three vertices, and paths on four vertices as connected components. Only the paths on four vertices have a higher α-sequence than the star with three leaves, but, by

the preconditions of Lemma 4.15, these are not forbidden. Therefore, X obstructs all forbidden induced subgraphs in G, and thus $G - X \in \Pi$. Moreover, X is minimal, since for each $v \in X$, $G - (X \setminus \{v\})$ contains a star with three leaves. We claim that F has a satisfying assignment if and only if there exists a size-$(3q+3pq)$ set X', $X' \cap X = \emptyset$, that obstructs all forbidden induced subgraphs in G. The proof of the claim is very similar to the proof of Lemma 4.14. For this reason, we omit the details. Note that for each variable gadget (together with the degree-one vertices that have been added) the only possibilities are that all vertices labeled "$+$" or all vertices labeled "$-$" can be in X', and for each clause, X' must contain exactly three vertices, one for each literal. For each clause gadget in G, at least one leaf must be in X', and thus the clause gadgets are obstructed. This guarantees the satisfiability of the corresponding formula F if X' has size $3q + 3pq$. □

Clearly, Lemmas 4.13–4.15 yield Theorem 4.12.

5. OUTLOOK

As indicated in the introductory section, there are important problems amenable to iterative compression that do not fall into the problem class studied here. Among these, in particular, we have DIRECTED FEEDBACK VERTEX SET and ALMOST 2-SAT. Hence, it would be interesting to further generalize our results to other problem classes, among these also being vertex deletion problems on directed graphs or bipartite graphs, and edge deletion problems. Moreover, our work here has left open the case where a forbidden subgraph may consist of more than one connected component.

6. ACKNOWLEDGEMENTS

We thank three anonymous referees of *Transactions on Computation Theory* for helpful remarks improving the presentation of this paper.

REFERENCES

AARTS, E. AND LENSTRA, J. K. 1997. *Local Search in Combinatorial Optimization*. Wiley.

ABELLO, J., RESENDE, M. G. C., AND SUDARSKY, S. 2002. Massive quasi-clique detection. In *Proceedings of the 5th Latin American Symposium on Theoretical Informatics (LATIN '02)*. LNCS Series, vol. 2286. Springer, 598–612.

BÖCKENHAUER, H.-J., HROMKOVIČ, J., MÖMKE, T., AND WIDMAYER, P. 2008. On the hardness of reoptimization. In *Proceedings of the 34th Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM '08)*. LNCS Series, vol. 4910. Springer, 50–65.

CAI, L. 1996. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters 58,* 4, 171–176.

CHEN, J., FOMIN, F. V., LIU, Y., LU, S., AND VILLANGER, Y. 2008. Improved algorithms for feedback vertex set problems. *Journal of Computer and System Sciences 74,* 7, 1188–1198.

CHEN, J., LIU, Y., LU, S., O'SULLIVAN, B., AND RAZGON, I. 2008. A fixed-parameter algorithm for the directed feedback vertex set problem. *Journal of the ACM 55,* 5. Article 21, 19 pages.

DEHNE, F. K. H. A., FELLOWS, M. R., LANGSTON, M. A., ROSAMOND, F. A., AND STEVENS, K. 2007. An $O(2^{O(k)}n^3)$ FPT algorithm for the undirected feedback vertex set problem. *Theory of Computing Systems 41,* 3, 479–492.

GABOW, H. N. AND TARJAN, R. E. 1989. Faster scaling algorithms for network problems. *SIAM Journal on Computing 18,* 5, 1013–1036.

GAREY, M. R. AND JOHNSON, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman.

GREENWELL, D. L., HEMMINGER, R. L., AND KLERLEIN, J. B. 1973. Forbidden subgraphs. In *Proceedings of the 4th Southeastern Conference on Combinatorics, Graph Theory and Computing*. 389–394.

GUO, J. 2006. Algorithm design techniques for parameterized graph modification problems. Ph.D. thesis, Institut für Informatik, Friedrich-Schiller-Universität Jena.

GUO, J., GRAMM, J., HÜFFNER, F., NIEDERMEIER, R., AND WERNICKE, S. 2006. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *Journal of Computer and System Sciences 72,* 8, 1386–1396.

GUO, J., MOSER, H., AND NIEDERMEIER, R. 2009. Iterative compression for exactly solving NP-hard minimization problems. In *Algorithmics of Large and Complex Networks*. LNCS Series, vol. 5515. Springer, 65–80.

HÜFFNER, F., KOMUSIEWICZ, C., MOSER, H., AND NIEDERMEIER, R. 2010. Fixed-parameter algorithms for cluster vertex deletion. *Theory of Computing Systems 47,* 1, 196–217.

KHOT, S. AND RAMAN, V. 2002. Parameterized complexity of finding subgraphs with hereditary properties. *Theoretical Computer Science 289,* 2, 997–1008.

KRAWCZYK, A. 1999. The complexity of finding a second Hamiltonian cycle in cubic graphs. *Journal of Computer and System Sciences 58,* 3, 641–647.

LEWIS, J. M. AND YANNAKAKIS, M. 1980. The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences 20,* 2, 219–230.

MARX, D. 2010. Chordal deletion is fixed-parameter tractable. *Algorithmica 57,* 4, 747–768.

MARX, D. AND SCHLOTTER, I. 2007. Obtaining a planar graph by vertex deletion. In *Proceedings of the 33rd International Workshop on Graph-Theoretic Concepts in Computer Science (WG '07)*. LNCS Series, vol. 4769. Springer, 292–303.

MOSER, H., NIEDERMEIER, R., AND SORGE, M. 2009. Algorithms and experiments for clique relaxations—finding maximum s-plexes. In *Proceedings of the 8th International Symposium on Experimental Algorithms (SEA '09)*. LNCS Series, vol. 5526. Springer, 233–244.

NIEDERMEIER, R. 2006. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press.

NISHIMURA, N., RAGDE, P., AND THILIKOS, D. M. 2005. Fast fixed-parameter tractable algorithms for nontrivial generalizations of Vertex Cover. *Discrete Applied Mathematics 152,* 1–3, 229–245.

PAPADIMITRIOU, C. H. 1994. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences 48,* 3, 498–532.

RAZGON, I. AND O'SULLIVAN, B. 2009. Almost 2-SAT is fixed-parameter tractable. *Journal of Computer and System Sciences 75,* 8, 435–450.

REED, B., SMITH, K., AND VETTA, A. 2004. Finding odd cycle transversals. *Operations Research Letters 32,* 4, 299–301.