Reducing the Vertex Cover Number via Edge Contraction

Paloma T. Lima, Vinicius Fernandes dos Santos, Ignasi Sau, Uéverton dos Santos Souza, Prafullkumar Tale

Reducing the Vertex Cover Number via Edge Contractions

Paloma T. Lima
Computer Science Department, IT University of Copenhagen, Denmark

Vinicius F. dos Santos
Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Belo Horizonte, Brazil

Ignasi Sau
LIRMM, Université de Montpellier, CNRS, Montpellier, France

Uéverton S. Souza
Instituto de Computação, Universidade Federal Fluminense, Niterói, Brazil
Institute of Informatics, University of Warsaw, Warsaw, Poland

Prafullkumar Tale
CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Abstract

The Contraction(vc) problem takes as input a graph G on n vertices and two integers k and d, and asks whether one can contract at most k edges to reduce the size of a minimum vertex cover of G by at least d. Recently, Lima et al. [MFCS 2020, JCSS 2021] proved, among other results, that unlike most of the so-called *blocker problems*, Contraction(vc) admits an XP algorithm running in time $f(d) \cdot n^{\mathcal{O}(d)}$. They left open the question of whether this problem is FPT under this parameterization. In this article, we continue this line of research and prove the following results:

- Contraction(vc) is W[1]-hard parameterized by $k + d$. Moreover, unless the ETH fails, the problem does not admit an algorithm running in time $f(k + d) \cdot n^{o(k+d)}$ for any function f. In particular, this answers the open question stated in Lima et al. [MFCS 2020] in the negative.
- It is NP-hard to decide whether an instance (G, k, d) of Contraction(vc) is a Yes-instance even when $k = d$, hence enhancing our understanding of the classical complexity of the problem.
- Contraction(vc) can be solved in time $2^{\mathcal{O}(d)} \cdot n^{k-d+\mathcal{O}(1)}$. This XP algorithm improves the one of Lima et al. [MFCS 2020], which uses Courcelle's theorem as a subroutine and hence, the $f(d)$-factor in the running time is non-explicit and probably very large. On the other hand, this shows that when $k = d$, the problem is FPT parameterized by d (or by k).

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Blocker problems, edge contraction, vertex cover, parameterized complexity.

Digital Object Identifier 10.4230/LIPIcs.MFCS.2022.20

Related Version The full version of this article is available at `https://arxiv.org/abs/2202.03322`.

Funding *Vinicius F. dos Santos*: Grant APQ-01707-21 Minas Gerais Research Support Foundation (FAPEMIG) and Grant 312069/2021-9 National Council for Scientific and Technological Development (CNPq).
Ignasi Sau: CAPES-PRINT Institutional Internationalization Program, process 88887.371209/2019-00, and projects DEMOGRAPH (ANR-16-CE40-0028), ESIGMA (ANR-17-CE23-0010), ELIT (ANR-20-CE48-0008-01), and UTMA (ANR-20-CE92-0027).
Uéverton S. Souza: This research has received funding from Rio de Janeiro Research Support Foundation (FAPERJ) under grant agreement E-26/201.344/2021, National Council for Scientific and Technological Development (CNPq) under grant agreement 309832/2020-9, and the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement CUTACOMBS (No. 714704).

Prafullkumar Tale: The author has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement SYSTEMATICGRAPH (No. 725978).

Acknowledgements The last author would like to thank Roohani Sharma for insightful discussions.

1 Introduction

Graph modification problems have been extensively studied in theoretical computer science, in particular for their vast expressive power and their applicability in a number of scenarios. Such problems can be generically defined as follows. For a fixed graph class $\mathcal{F}$ and a fixed set $\mathcal{M}$ of allowed graph modification operations, such as vertex deletion, edge deletion, edge addition, edge editing or edge contraction, the $\mathcal{F}$-$\mathcal{M}$-Modification problem takes as input a graph G and a positive integer k, and the goal is to decide whether at most k operations from $\mathcal{M}$ can be applied to G so that the resulting graph belongs to the class $\mathcal{F}$. For most natural graph classes $\mathcal{F}$ and modification operations $\mathcal{M}$, the $\mathcal{F}$-$\mathcal{M}$-Modification problem is NP-hard [15, 16]. To cope up with this hardness, these problems have been examined via the lens of parameterized complexity [1, 3]. With an appropriate choice of $\mathcal{F}$ and the allowed modification operations $\mathcal{M}$, $\mathcal{F}$-$\mathcal{M}$-Modification can encapsulate well-studied problems like Vertex Cover, Feedback Vertex Set (FVS), Odd Cycle Transversal (OCT), Chordal Completion, or Cluster Editing, to name just a few.

The most natural and well-studied modification operations are, probably in this order, vertex deletion, edge deletion, and edge addition. In recent years, the *edge contraction* operation has begun to attract significant scientific attention. (When contracting an edge uv in a graph G, we delete u and v from G, add a new vertex and make it adjacent to vertices that were adjacent to u or v.) In parameterized complexity, $\mathcal{F}$-Contraction problems, i.e., $\mathcal{F}$-$\mathcal{M}$-Modification problems where the only modification operation in $\mathcal{M}$ is edge contraction, are usually studied with the number of edges allowed to contract, k, as the parameter. A series of more than 15 recent papers studied the parameterized complexity of $\mathcal{F}$-Contraction for various graph classes $\mathcal{F}$ (see [14] and the references therein, as well as the full version of this article, for the precise list of these classes $\mathcal{F}$).

For all the $\mathcal{F}$-$\mathcal{M}$-Modification problems mentioned above, a typical definition of the problem contains a description of the target graph class $\mathcal{F}$. For example, Vertex Cover, FVS, and OCT are $\mathcal{F}$-$\mathcal{M}$-Modification problems where $\mathcal{F}$ is the collection of edgeless graphs, forests, and bipartite graphs, respectively, and $\mathcal{M}$ contains only vertex deletion. Recently, a different formulation of these graph modification problems, called *blocker problems*, has been considered. In this formulation, the target graph class is defined in a *parametric way* from the input graph. To make the statement of such problems precise, consider an invariant $\pi : \mathcal{G} \mapsto \mathbb{N}$, where $\mathcal{G}$ is the collection of all graphs. For a fixed invariant π, a typical input of a blocker problem consists of a graph G, a budget k, and a threshold value d, and the question is whether G can be converted into a graph G' using at most k allowed modifications such that $\pi(G') \leq \pi(G) - d$. This is the same as determining whether (G, k, d) is a Yes-instance of $\mathcal{F}^{\pi}_{G,d}$-$\mathcal{M}$-Modification where $\mathcal{F}^{\pi}_{G,d} = \{G' \in \mathcal{G} \mid \pi(G') \leq \pi(G) - d\}$.

Consider the following examples of this formulation. For the invariant $\pi(G) = |E(G)|$, threshold $d = |E(G)|$, and vertex deletion as the modification operation in $\mathcal{M}$, $\mathcal{F}^{\pi}_{G,d}$-$\mathcal{M}$-Modification is the same as Vertex Cover. Setting the threshold d to a fixed integer p leads to Partial Vertex Cover. In a typical definition of this problem, the input is a graph G and two integers k, p, and the objective is to decide whether there is a set of vertices of size at most k that has at least p edges incident on it. Consider another example when

$\pi(G) = \mathtt{vc}(G)$, the size of a minimum vertex cover of G, the threshold value $d = \mathtt{vc}(G) - 1$, and the allowed modification operation is edge contraction. To reduce the size of a minimum vertex cover from $\mathtt{vc}(G)$ to 1 by k edge contractions, we need to find a connected vertex cover of size $k+1$. Hence, in this case $\mathcal{F}^{\pi}_{G,d}$-$\mathcal{M}$-Modification is the same as Connected Vertex Cover. In all these cases, we can think of the set of vertices or edges involved in the modifications as 'blocking' the invariant π, that is, preventing π from being smaller.

With 'vertex deletion' or 'edge deletion' as the allowed graph modification operation, blocker problems have been investigated for numerous graph invariants (see the full version of this article for an exhaustive list of these invariants, along with the appropriate references). Blocker problems for edge contraction have already been studied with respect to the chromatic number, clique number, independence number [6, 13], the domination number [8, 10], total domination number [9], and the semitotal domination number [7].

This article is strongly motivated by the results in [12]. They proved, in particular, that it is coNP-hard to test whether we can reduce the size of a minimum feedback vertex set or of a minimum odd cycle transversal of a graph by one, i.e., $d = 1$, by performing one edge contraction, i.e., $k = 1$. This is consistent with earlier results, as blocker problems are generally very hard, and become polynomial-time solvable only restricted to specific graph classes. However, the notable exception is when the invariant is the size of a minimum vertex cover of the input graph. We define the problem before mentioning existing results and our contribution (G/F denotes the graph obtained from G by contracting the edges in F).

Contraction($\mathtt{vc}$)
Input: An undirected graph G and two non-negative integers k and d.
Question: Does there exist a set $F \subseteq E(G)$ such that $|F| \leq k$ and $\mathtt{vc}(G/F) \leq \mathtt{vc}(G) - d$?

Our results. A simple reduction, briefly mentioned in [12], shows that the above problem is NP-hard for *some* k in $\{d, d+1, \ldots, 2d\}$. In our first result, we enhance our understanding of the classical complexity of the problem and prove that the problem is NP-hard even when $k = d$. As any edge contraction can decrease $\mathtt{vc}(G)$ by at most one, if $k < d$ then the input instance is a trivial No-instance. To state our first result, we introduce the notation of $\mathsf{rank}(G)$, which is the number of vertices of G minus its number of connected components (or equivalently, the number of edges of a set of spanning trees of each of the connected components of G). Note that it is sufficient to consider values of k that are at most $\mathsf{rank}(G)$, as otherwise it is possible to transform G to an edgeless graph with at most k contractions, and therefore in this case G is a yes-instance for Contraction($\mathtt{vc}$) if and only if $\mathtt{vc}(G) \geq d$.

▶ **Theorem 1.** *To decide whether an instance (G, k, d) of* Contraction($\mathtt{vc}$) *is a* Yes*-instance is*

- coNP-hard *if $k = \mathsf{rank}(G)$,*
- coNP-hard *if $k < \mathsf{rank}(G)$ and $2d \leq k$, and*
- NP-hard *if $k < \mathsf{rank}(G)$ and $k = d + \frac{\ell-1}{\ell+3} \cdot d$ for any integer $\ell \geq 1$ such that k is an integer.*

As one needs to contract at least d edges to reduce the size of a minimum vertex cover by d, the above theorem, for $\ell = 1$, implies that the problem is para-NP-hard when parameterized by the 'excess over the lower bound', i.e., by $k - d$. Since we can assume that $d \leq k$, d is a 'stronger' parameter than k. One of the main results of [12] is an XP algorithm for Contraction($\mathtt{vc}$) with running time $f(d) \cdot n^{\mathcal{O}(d)}$. Here, and in the rest of the article, we denote by n the number of vertices of the input graph. The authors explicitly asked whether the problem admits an FPT algorithm parameterized by d. As our next result, we answer

this question in the negative by proving that such an algorithm is highly unlikely, even when parameterized by the larger parameter $d + k$ (or equivalently, just k, as discussed above).

▶ **Theorem 2.** Contraction(vc) *is* W[1]-hard *parameterized by* $k + d$. *Moreover, unless the* ETH *fails*[1]*, it does not admit an algorithm running in time* $f(k + d) \cdot n^{o(k+d)}$ *for any computable function* $f : \mathbb{N} \mapsto \mathbb{N}$. *The result holds even if we assume that the input* (G, k, d) *is such that* $k < \mathsf{rank}(G)$ *and* $d \leq k < 2d$, *and* G *is a bipartite graph with a bipartition* $\langle X, Y \rangle$ *such that* X *is a minimum vertex cover of* G.

For the XP algorithm in [12], the authors did not explicitly mention an upper bound on the corresponding function f, but it is expected to be quite large since it uses Courcelle's theorem [2] as a subroutine. Our next result provides a concrete upper bound on the running time, and distinguishes in a precise way the contribution of k and d.

▶ **Theorem 3.** *There exists an algorithm that solves* Contraction(vc) *in time* $2^{\mathcal{O}(d)} \cdot n^{k-d+\mathcal{O}(1)}$. *Moreover, for an input* (G, k, d), *the algorithm runs in time* $2^{\mathcal{O}(d)} \cdot n^{\mathcal{O}(1)}$ *unless* $k < \mathsf{rank}(G)$ *and* $d \leq k < 2d$.

Note that the above result implies, in particular, that the problem is FPT parameterized by d when $k - d$ is a constant.

Our methods. A central tool in both our negative and positive results is Lemma 6, which allows us to reformulate the problem as follows. As discussed later, by applying appropriate FPT reductions to the input graph G, it is possible to assume that we have at hand a minimum vertex cover X of G. We say that a set of edges F is a *solution* of (G, k, d) if $|F| \leq k$ and $\mathsf{vc}(G/F) \leq \mathsf{vc}(G) - d$. Lemma 6 implies that there exists such a solution (i.e., an edge set) if and only if there exist vertex subsets $X_s \subseteq X$ and $Y_s \subseteq V(G) \setminus X$ such that the pair $\langle X_s, Y_s \rangle$, which we call a *solution pair*, satisfies the technical conditions mentioned in its statement (and which we prefer to omit here). This reformulation allows us to convert the problem of finding a subset F of edges to the problem of modifying the given minimum vertex cover X to obtain another vertex cover $X_{\mathsf{el}} = (X \setminus X_s) \cup Y_s$ such that $|X_{\mathsf{el}}| \leq |X| + (k - d)$ and $\mathsf{rank}(X_{\mathsf{el}}) \geq k$. Here, we define $\mathsf{rank}(X_{\mathsf{el}}) := \mathsf{rank}(G[X_{\mathsf{el}}])$. See the full version for an illustration.

In our hardness reductions, another simple, yet critical, tool is Lemma 7, which states that if there is a vertex which is adjacent to a pendant vertex (i.e., a vertex of degree one), then there is a solution pair that does not contain this vertex. We present overviews of the reductions in Section 3 and in the full version to demonstrate the usefulness of these two lemmas in the respective hardness results. The reduction that we use to prove the third item in the statement of Theorem 1 (which is the most interesting case) is from a variant of Multicolored Independent Set, while the one in the proof of Theorem 2 is from Edge Induced Forest, a problem that we define and that we prove to be W[1]-hard in Theorem 8, by a parameter preserving reduction from, again, Multicolored Independent Set. It is worth mentioning that the W[1]-hardness in Theorem 8 holds even if we assume that the input graph G is a bipartite graph with a bipartition $\langle X, Y \rangle$ such that X is a minimum vertex cover of G, and such that $k < \mathsf{rank}(G)$ and $d \leq k < 2d$. This case is the crux of the difficulty of the problem. This becomes clear in the XP algorithm of Theorem 3 that we proceed to discuss.

[1] The Exponential Time Hypothesis (ETH) is a conjecture stating that N-variable 3-SAT cannot be solved in time $2^{o(N)}$. We refer the reader to [4, Chapter 14] for the formal definition and related literature.

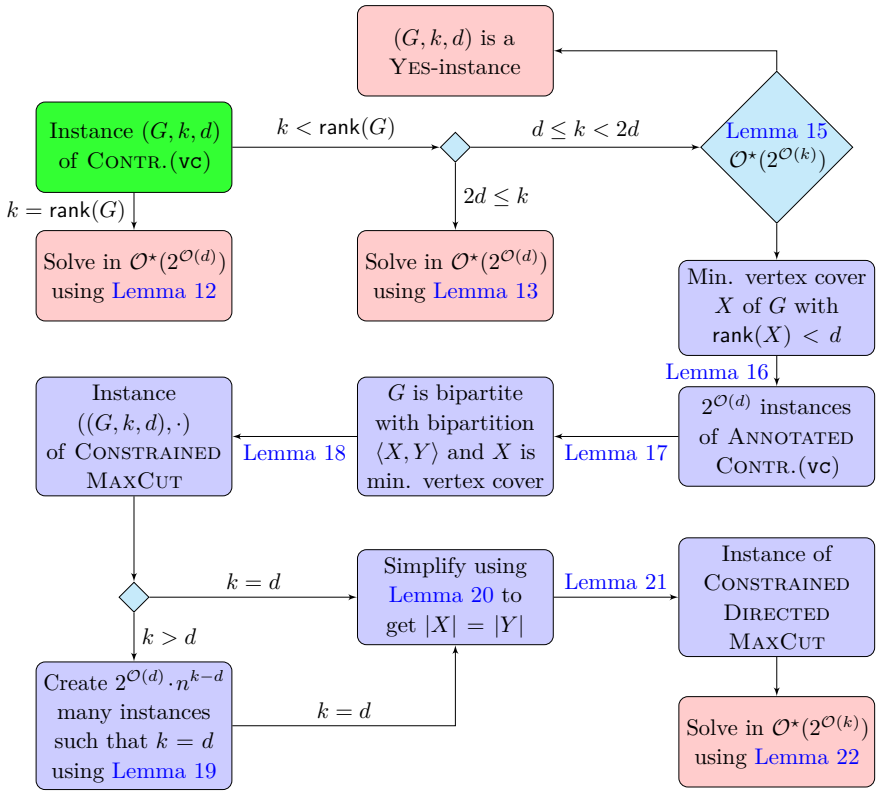

Figure 1 Diagram of the algorithm for CONTRACTION(vc) given by Theorem 3. Recall that we can assume that $d \le k \le \mathsf{rank}(G)$, hence the case distinction considered in the beginning is exhaustive. Note also that, in the case where $d \le k < 2d$, it holds that $\mathcal{O}^\star(2^{\mathcal{O}(k)}) = \mathcal{O}^\star(2^{\mathcal{O}(d)})$.

The algorithm for CONTRACTION(vc), which is our main technical contribution, is provided in Section 4 and summarized in the diagram of Figure 1. By a standard Knapsack-type dynamic programming, mentioned in [12], we can assume that the input graph G is connected. We distinguish three cases depending on the relation between k, d, and $\mathsf{rank}(G)$. The first two cases are easy, and can be solved in time $2^{\mathcal{O}(d)} \cdot n^{\mathcal{O}(1)}$, by essentially running an FPT algorithm to determine whether $\mathsf{vc}(G) < d$; see Lemma 12 and Lemma 13. We now present an overview of the algorithm for the third case, namely when its input (G,k,d) is with guarantees that $k < \mathsf{rank}(G)$ and $d \le k < 2d$ (cf. Lemma 14). Inspired by Lemma 6, we introduce an annotated version of the problem called ANNOTATED CONTRACTION(vc). We first argue (cf. Lemma 15) that there is a $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ algorithm that either correctly concludes that (G,k,d) is a YES-instance of CONTRACTION(vc) or finds a minimum vertex cover X of G such that $\mathsf{rank}(X) < d$. Using X, we can construct $2^{\mathcal{O}(d)}$ many instances of ANNOTATED CONTRACTION(vc) such that (G,k,d) is a YES-instance of CONTRACTION(vc) if and only if at least one of these new instances is a YES-instance of ANNOTATED CONTRACTION(vc) (cf. Lemma 16). Hence, it suffices to design an algorithm to solve ANNOTATED CONTRACTION(vc). We show that we can apply a simple reduction rule (cf. Lemma 17) that allows us to assume that the input graph G of ANNOTATED CONTRACTION(vc) is bipartite with bipartition $\langle X,Y\rangle$ such that X is a minimum vertex cover of G, as mentioned above.

A solution of an instance of ANNOTATED CONTRACTION(vc) is a solution pair $\langle X_s, Y_s\rangle$ as stated in Lemma 6. We find convenient to present an algorithm that finds a partition

$\langle V_L, V_R \rangle$ of $V(G)$ instead of a solution pair $\langle X_s, Y_s \rangle$. To formalize this, we introduce the problem called Constrained MaxCut and we show it to be equivalent to Annotated Contraction(vc) (cf. Lemma 18). We partition the input instances of Constrained MaxCut into the following two types: (1) $k = d$, and (2) $k > d$. For the instances of the second type, we construct $2^{\mathcal{O}(d)} \cdot n^{k-d}$ many instances of the first type such that the input instance is a Yes-instance if and only if at least one of these newly created instances is a Yes-instance (cf. Lemma 19). We remark that this is the only step in the whole algorithm where an n^{k-d}-factor appears (note that this is unavoidable by Theorem 8).

Finally, to handle the instances of the first type (i.e., with $k = d$), we apply a simplification based on the existence of a matching saturating X (cf. Lemma 20), we introduce a directed variation of the problem called Constrained Directed MaxCut, and we prove it to be equivalent to its undirected version (cf. Lemma 21). We then present a dynamic programming algorithm, with running time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$, that critically uses the fact that $k = d$ (cf. Lemma 22), in particular to "merge" appropriately some directed cycles in order to obtain a directed *acyclic* graph, whose topological ordering gives a natural way to process the vertices of the input graph in a dynamic programming fashion. At the end of Section 4 we present an overview of this algorithm.

Organization. Due to space limitations, in this extended abstract most of the technical contents of the paper can only be found in the full version, in particular the proofs of the statements marked with '($\star$)'. In Section 2 we present some notations and preliminary results about Contraction(vc). The proof of Theorem 1 can be found in the full version. In Section 3 we present the proof of Theorem 2. Section 4 is the most technical part of the paper, and contains the description of the algorithm to solve Contraction(vc) mentioned in Theorem 3. We conclude the article in Section 5 with some open problems.

2 Preliminaries

We use standard graph-theoretic notation, and we refer the reader to [5] for any undefined notation. Similarly, we use standard terminology of parameterized complexity, and we refer the reader to [4]. For completeness, we present in the full version the required basic preliminaries about graph theory and parameterized complexity, and we provide here some non-standard definitions and useful properties of the Contraction(vc) problem.

For a positive integer q, we denote the set $\{1, 2, \ldots, q\}$ by $[q]$. We use $\mathbb{N}$ to denote the collection of all non-negative integers. A *spanning forest* of a graph is a collection of spanning trees of its connected components. As already mentioned in Section 1, the *rank* of a graph G, denoted by $\mathsf{rank}(G)$, is the number of edges of a spanning forest of G. The *rank* of a set $X \subseteq V(G)$ of vertices, denoted by $\mathsf{rank}(X)$, is the rank of $G[X]$. The *rank* of a set $F \subseteq V(G)$ of edges, denoted by $\mathsf{rank}(F)$, is the rank of $V(F)$. Note that an edge contraction decreases the rank of a graph G by exactly one. We present a couple of observations regarding an instance (G, k, d) of the Contraction(vc) problem. Later, we present a lemma that helps us to characterize the problem as finding a vertex cover with special properties.

▶ **Observation 4** ($\star$). *Consider an instance (G, k, d) of* Contraction(vc) *such that $k = \mathsf{rank}(G)$. Then, (G, k, d) is a* Yes*-instance if and only if $d \leq \mathsf{vc}(G)$.*

▶ **Observation 5** ($\star$). *Consider an instance (G, k, d) of* Contraction(vc) *such that G is a connected graph, $k < \mathsf{rank}(G)$, and $2d \leq k$. Then, (G, k, d) is a* Yes*-instance if and only if $d < \mathsf{vc}(G)$.*

Suppose that (G, k, d) is a YES-instance of CONTRACTION(vc). We say that a set $F \subseteq E(G)$ is a *solution* of (G, k, d) if $|F| \leq k$ and $\mathtt{vc}(G/F) \leq \mathtt{vc}(G) - d$. Fix a minimum vertex cover X of G. As X is a vertex cover, for every edge in F, *at least one* of its endpoints is in X. We argue that one can construct an *enlarged vertex cover* X_{el} of G such that for every edge in F, *both* of its endpoints are in X_{el}. Also, X_{el} is not much larger than X. In order to construct X_{el} from X, one needs to remove and add some vertices to X. We denote the removed and added vertices by X_s and Y_s, respectively, and call $\langle X_s, Y_s \rangle$ a *solution pair*. See the corresponding figure in the full version for an illustration. The following lemma relates a solution (a set of edges) to a solution pair (a tuple of disjoint vertex sets).

▶ **Lemma 6** ($\star$). *Consider a connected graph G, a minimum vertex cover X of G, a proper subset F of edges of a spanning forest of G (i.e., $|F| < \mathsf{rank}(G)$), and a non-negative integer d. Then, $\mathtt{vc}(G/F) \leq \mathtt{vc}(G) - d$ if and only if there exists subsets $X_s \subseteq X$ and $Y_s \subseteq V(G) \setminus X$ such that (i) $X_{\mathsf{el}} := (X \setminus X_s) \cup Y_s$ is a vertex cover of G, (ii) $\mathsf{rank}((X \setminus X_s) \cup Y_s) \geq |F|$, and (iii) $|Y_s| - |X_s| \leq |F| - d$, i.e., $|X_{\mathsf{el}}| \leq |X| + |F| - d$.*

In the following lemma, we argue that there exists a solution pair $\langle X_s, Y_s \rangle$ such that X_s does not contain any vertex in X which is adjacent to a pendant vertex. For example, in the aforementioned figure in the full version, there exists a solution pair $\langle X_s, Y_s \rangle$ such that $x_1 \notin X_s$.

▶ **Lemma 7** ($\star$). *Consider a connected graph G, a minimum vertex cover X of G, and two integers ℓ and d. Suppose that there exists a vertex x° in X which is adjacent to a pendant vertex. Suppose that there are subsets $X_s \subseteq X$ and $Y_s \subseteq V(G) \setminus X$ such that (i) $(X \setminus X_s) \cup Y_s$ is a vertex cover of G, (ii) $\mathsf{rank}((X \setminus X_s) \cup Y_s) \geq \ell$, and (iii) $|Y_s| - |X_s| \leq \ell - d$. Then, there are subsets $X'_s \subseteq X$ and $Y'_s \subseteq V(G) \setminus X$ that satisfy these three conditions and $x^\circ \notin X'_s$.*

3 W[1]-hardness results

In this section we prove Theorem 2. To do so, we introduce the EDGE INDUCED FOREST problem, defined below, and present a parameter preserving reduction from MULTICOLORED INDEPENDENT SET to it. This reduction, along with known results about MULTICOLORED INDEPENDENT SET (cf. the preliminaries in the full version), imply the corresponding result for EDGE INDUCED FOREST. We then present a parameter preserving reduction from EDGE INDUCED FOREST to CONTRACTION(vc). This reduction, along with Theorem 8, yield Theorem 2.

EDGE INDUCED FOREST
Input: A graph G and an integer ℓ.
Question: Is there a set F of at least ℓ edges in G such that $G[V(F)]$ is a forest?

We note that a similar problem called INDUCED FOREST has already been studied. In this problem, the input is the same but the objective is to find a subset X of *vertices* of G of size at least ℓ such that $G[X]$ is a forest. The general result of Khot and Raman [11] implies that INDUCED FOREST is W[1]-hard when parameterized by the size of the solution ℓ. As expected, we can prove a similar result for EDGE INDUCED FOREST.

▶ **Theorem 8** ($\star$). EDGE INDUCED FOREST, *parameterized by the size of the solution ℓ, is* W[1]-hard. *Moreover, unless the* ETH *fails, it does not admit an algorithm running in time $f(\ell) \cdot n^{o(\ell)}$ for any computable function $f : \mathbb{N} \mapsto \mathbb{N}$.*

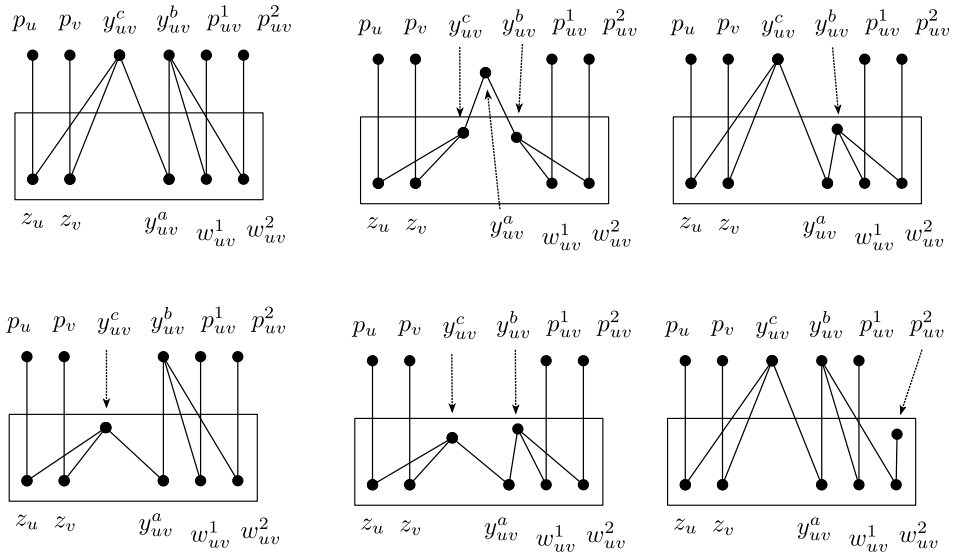

Figure 2 The top-left figure illustrates an encoding of edge uv in G while reducing from an instance of Edge Induced Forest to an instance of Contraction(vc). The remaining five figures correspond to the partition of Y_s mentioned in the proof of Lemma 11.

We now present a parameter preserving reduction from Edge Induced Forest to Contraction(vc), which takes an instance (G, ℓ) of Edge Induced Forest and returns an instance (G', k, d) of Contraction(vc). It constructs a graph G' from G as follows:

- Initialize $V(G') = E(G') = \emptyset$.
- For every vertex u in $V(G)$, add two vertices z_u, p_u to $V(G')$ and the edge $z_u p_u$ to $E(G')$.
- For every edge uv in $E(G)$, add the vertex set $\{y^a_{uv}, y^b_{uv}, y^c_{uv}, w^1_{uv}, w^2_{uv}, p^1_{uv}, p^2_{uv}\}$ to $V(G')$. Add edges $\{z_u y^c_{uv}, z_v y^c_{uv}\}$ to $E(G')$. These edges encode adjacency relations in G. Add also the edges $\{y^a_{uv} y^b_{uv}, y^a_{uv} y^c_{uv}, y^b_{uv} w^1_{uv}, y^b_{uv} w^2_{uv}, w^1_{uv} p^1_{uv}, w^2_{uv} p^2_{uv}\}$ to $E(G')$. These edges are part of a gadget which is private to edge uv.

This completes the construction of G'. The reduction sets $k = 4 \cdot \ell$, $d = 3 \cdot \ell$, and returns (G', k, d) as the constructed instance. Note that, indeed, $k < \mathsf{rank}(G')$ and $d \leq k < 2d$ (more precisely, $k - d = \frac{d}{3}$). See Figure 2 for an illustration. Before proving the correctness of the reduction, we first note some properties of the graph G'. We define the following sets:

- $Z := \{z_u \in V(G') \mid u \in V(G)\}$,
- $Y^{abc} := Y^a \cup Y^b \cup Y^c$ where $Y^a := \{y^a_{uv} \in V(G') \mid uv \in E(G)\}$, $Y^b := \{y^b_{uv} \in V(G') \mid uv \in E(G)\}$, and $Y^c := \{y^c_{uv} \in V(G') \mid uv \in E(G)\}$,
- $W := \{w^1_{uv}, w^2_{uv} \in V(G') \mid uv \in E(G)\}$, and
- $P := \{p_u \in V(G') \mid u \in V(G)\} \cup \{p^1_{uv}, p^2_{uv} \mid uv \in E(G)\}$.

Note that $\langle Z, Y^{abc}, W, P\rangle$ is a partition of $V(G')$ where each vertex in P is a pendant vertex and each vertex in $Z \cup W$ is adjacent to a pendant vertex in P. Moreover, $\mathsf{rank}(G') > k$. Note that $X := Z \cup W \cup Y^a$ is an independent set in G'. In the next lemma, we argue that it is also a minimum vertex cover of G', which implies that, as claimed in the statement of Theorem 2, G' is a bipartite graph with a bipartition $\langle X, Y\rangle$ such that X is a minimum vertex cover of G'.

▶ **Lemma 9** (⋆). *The set $X = Z \cup W \cup Y^a$ is a minimum vertex cover of G'.*

The following lemma corresponds to the "easy" direction of the reduction.

▶ **Lemma 10** (⋆). *If* (G, ℓ) *is a* Yes*-instance of* Edge Induced Forest*, then* (G', k, d) *is a* Yes*-instance of* Contraction(vc).

We now present a brief overview of the proof of the correctness in the backward direction, corresponding to Lemma 11, whose full proof can be found in the full version. By Lemma 6, there is a solution F of (G', k, d) if and only if there exists a solution pair $\langle X_s, Y_s \rangle$ such that (i) $X_{\mathsf{el}} = (X \setminus X_s) \cup Y_s$ is a vertex cover of G', (ii) $\mathsf{rank}(X_{\mathsf{el}}) \geq |F| = k = 4 \cdot \ell$, and (iii) $|Y_s| - |X_s| \leq k - d = \ell$. Note that as X and $Y = V(G') \setminus X$ are independent sets in G', every edge in $E(X_{\mathsf{el}})$ is incident on exactly one vertex in Y_s. We can interpret the second condition as a *value function* and the third condition as a *cost function*. In other words, our objective is to find sets X_s, Y_s such that their cost, i.e., $|Y_s| - |X_s|$, is at most ℓ whereas their value, i.e., the rank of edges in $E(X_{\mathsf{el}})$ that are incident on Y_s, is at least $4 \cdot \ell$. Lemma 7 implies that the vertices of the form z_u, w^1_{uv}, and w^2_{uv} are in X_{el}. The first condition implies that only the five configurations shown in Figure 2 are possible (the top-left is *not* a configuration). Starting from top-middle and moving row-wise, the individual value and cost of these configurations are $(4, 1)$, $(3, 1)$, $(3, 1)$, $(6, 2)$, and $(1, 1)$, respectively. To meet both the value and budget constraints, every vertex in X_s, Y_s needs to be the of first type. This implies there are ℓ vertices in X_s that are of the form y^a_{uv}, and Y_s contains the corresponding vertices of the form y^b_{uv} and y^c_{uv}. We argue that the edges corresponding to vertices in Y^c_{uv} form a solution of (G, ℓ) and formalize these ideas in the next lemma.

▶ **Lemma 11** (⋆). *If* (G', k, d) *is a* Yes*-instance of* Contraction(vc)*, then* (G, ℓ) *is a* Yes*-instance of* Edge Induced Forest.

We are ready to present the proof of Theorem 2.

Proof of Theorem 2. Consider the reduction presented in this subsection. Lemma 10 and Lemma 11 imply that the reduction is safe. By the description of the reduction, it outputs the constructed instance in polynomial time. The W[1]-hardness of Contraction(vc) follows from Theorem 8. As $k = 4 \cdot \ell$ and $d = 3 \cdot \ell$, if Contraction(vc) admits an algorithm with running time $f(k + d) \cdot n^{o(k+d)}$, then Edge Induced Forest also admits an algorithm with running time $f(\ell) \cdot n^{o(\ell)}$, which contradicts Theorem 8. ◀

4 Algorithm for Contraction(vc)

In this section we present the main ideas of the proof of Theorem 3, following the sketch provided in Section 1 and summarized in Figure 1. Due to space limitations, most of the technical content can be found in the full version Namely, we present an algorithm that takes as input an instance (G, k, d) of Contraction(vc) and returns either Yes or No, and whose high-level description is as follows (cf. Figure 1):

- If $k = \mathsf{rank}(G)$, then it uses the algorithm mentioned in Lemma 12.
- If $k < \mathsf{rank}(G)$ and $2d \leq k$, then it uses the algorithm mentioned in Lemma 13.
- If $k < \mathsf{rank}(G)$ and $d \leq k < 2d$, then it uses the algorithm mentioned in Lemma 14.

Note that, since we can safely assume that $d \leq k \leq \mathsf{rank}(G)$, the above three cases are exhaustive. The first two cases turn out to be quite easy to handle.

▶ **Lemma 12** (⋆). *There exists an algorithm that, given as input an instance* (G, k, d) *of* Contraction(vc) *with a guarantee that* $k = \mathsf{rank}(G)$*, runs in time* $1.2738^d \cdot n^{\mathcal{O}(1)}$*, and correctly determines whether it is a* Yes*-instance.*

▶ **Lemma 13** (⋆). *There exists an algorithm that, given as input an instance (G, k, d) of* Contraction(vc) *with guarantees that $k < \mathsf{rank}(G)$ and $2d \leq k$, runs in time $1.2738^d \cdot n^{\mathcal{O}(1)}$, and correctly determines whether it is a* Yes*-instance.*

We may assume that G is a connected graph; we justify this assumption in the full version. In order to deal with the third case above, our objective is to prove the following lemma.

▶ **Lemma 14.** *There exists an algorithm that, given as input an instance (G, k, d) of* Contraction(vc) *with guarantees that $k < \mathsf{rank}(G)$ and $d \leq k < 2d$, runs in time $2^{\mathcal{O}(d)} \cdot n^{k-d+\mathcal{O}(1)}$, and correctly determines whether it is a* Yes*-instance.*

We start with the following result, which will allow us to assume henceforth that we are equipped with a minimum vertex cover of the input graph with small rank.

▶ **Lemma 15** (⋆). *There exists an algorithm that, given as input an instance (G, k, d) of* Contraction(vc) *with guarantees that $k < \mathsf{rank}(G)$ and $d \leq k < 2d$, runs in time $2.6181^k \cdot n^{\mathcal{O}(1)}$, and either correctly concludes that (G, k, d) is a* Yes*-instance, or computes a minimum vertex cover X of G such that $\mathsf{rank}(X) < d$.*

Our next step is to provide an FPT-reduction from Contraction(vc) to the Annotated Contraction(vc) problem, defined as follows.

Annotated Contraction(vc)
Input: An instance (G, k, d) of Contraction(vc), a minimum vertex cover X of G, and a tuple $\langle X_L, X_R \rangle$ such that X_L, X_R are disjoint subsets of X.
Question: Do there exist sets $X_s \subseteq X$ and $Y_s \subseteq Y$ $(= V(G) \setminus X)$ such that (i) $(X \setminus X_s) \cup Y_s$ is a vertex cover of G, (ii) $\mathsf{rank}((X \setminus X_s) \cup Y_s) \geq k$, (iii) $|Y_s| - |X_s| \leq k - d$, and (iv) $X_L \cap X_s = \emptyset$ and $X_R \subseteq X_s$?

The first three conditions correspond to the three conditions mentioned in Lemma 6. We remark that there is a small technical caveat while using Lemma 6. Consider an instance (G, k, d) of Contraction(vc), and let F be a solution. Lemma 6 implies that there are subsets $X_s \subseteq X$ and $Y_s \subseteq V(G) \setminus X$ such that (i) $(X \setminus X_s) \cup Y_s$ is a vertex cover of G, (ii) $\mathsf{rank}((X \setminus X_s) \cup Y_s) \geq |F|$, and (iii) $|Y_s| - |X_s| \leq |F| - d$. However, the statement of Annotated Contraction(vc) specifies the integer k and not the actual size of a minimum solution F. For example, if there exists a solution F of size, say, $k/2$, then Lemma 6 ensures that $\mathsf{rank}((X \setminus X_s) \cup Y_s) \geq k/2$, however $\mathsf{rank}((X \setminus X_s) \cup Y_s)$ can be smaller than k. To overcome this, we assume that $(G, k-1, d)$ is a No-instance of Contraction(vc). This implies that if there is a subset F of $E(G)$ of size *at most* k such that $\mathtt{vc}(G/F) \leq \mathtt{vc}(G) - d$, then F is of size *exactly* k. We summarize below all the assumptions on the input instance.

▶ **Guarantee 4.1.** *Consider an instance (G, k, d) of* Contraction(vc) *that satisfies the following conditions.*

- *G is a connected graph, $k < \mathsf{rank}(G)$, and $d \leq k$.*
- *A minimum vertex cover X of G is provided as an additional part of the input.*
- *$\mathsf{rank}(X) < d$.*
- *$(G, k-1, d)$ is a* No*-instance of* Contraction(vc).

Unless stated otherwise, we denote the independent set $V(G) \setminus X$ by Y.

Consider an instance (G, k, d) of Contraction(vc) with Guarantee 4.1. Using Lemma 6, we construct $2^{\mathcal{O}(d)}$ many instances of Annotated Contraction(vc) such that (G, k, d) is a Yes-instance if and only if at least one of these newly created instances is a Yes-instance. Informally, let F be the set of edges in a spanning forest of $G[X]$. As $\mathsf{rank}(X) < d$, we have

$|F| < d$. We iterate over all 'valid' partitions $\langle X_L, X_R\rangle$ of $V(F)$. We construct an instance of Annotated Contraction(vc) for each such a partition. We formalize this intuition and prove its correctness in the following lemma.

▶ **Lemma 16** (⋆). *Suppose that there is an algorithm that solves* Annotated Contraction(vc) *in time $f(n,k,d)$. Then, there exists an algorithm that given as input an instance (G,k,d) of* Contraction(vc) *with Guarantee 4.1, runs in time $3^d \cdot n^{\mathcal{O}(1)} \cdot f(n,k,d)$, and correctly determines whether it is a* Yes*-instance.*

To solve an instance of Annotated Contraction(vc), we reduce it to an equivalent instance of the Constrained MaxCut problem. To present such a reduction, it is convenient to work with an instance $((G,k,d),X,\langle X_L,X_R\rangle)$ of Annotated Contraction(vc) where X is an independent set. This is guaranteed by the following reduction rule.

▶ **Reduction Rule 4.1.** *Let $((G,k,d),X,\langle X_L,X_R\rangle)$ be an instance of* Annotated Contraction(vc)*, $F_1 = E(X_L,X_R)$, and F_2 be the set of edges in a spanning forest of $G[X_L]$.*

- *Delete the edges in F_1.*
- *Contract the edges in F_2 and reduce both k and d by $|F_2|$.*

Return the instance $((G',k',d'),X',\langle X'_L,X_R\rangle)$ where $G' = (G-F_1)/F_2$, $k' = k-|F_2|$, $d' = d-|F_2|$, $X' = V(G[X]/F_2)$, and $X'_L = V(G[X_L]/F_2)$.

▶ **Lemma 17** (⋆). *Reduction Rule 4.1 is safe. Therefore, it is safe to assume that we are given an instance $((G,k,d),X,\langle X_L,X_R\rangle)$ of* Annotated Contraction(vc) *such that X is an independent set and a minimum vertex cover of G.*

We find the following reformulation of Annotated Contraction(vc) convenient to present an algorithm to solve it.

Constrained MaxCut
Input: An instance (G,k,d) of Contraction(vc), a minimum vertex cover X of G, and a tuple $\langle X_L,X_R\rangle$ such that X_L, X_R are disjoint subsets of X.
Question: Does there exist a partition $\langle V_L,V_R\rangle$ of $V(G)$ such that (i) $E(V_L\cap Y, V_R\cap X) = \emptyset$, (ii) $\mathsf{rank}(E(V_L\cap X, V_R\cap Y)) \geq k$, (iii) $|V_R\cap Y| - |V_R\cap X| \leq k-d$, and (iv) $X_L \subseteq V_L$ and $X_R \subseteq V_R$?

Note that in Annotated Contraction(vc) we are seeking for a pair of subsets, whereas in Constrained MaxCut we are looking for a partition of $V(G)$. Such a formulation allows us to handle vertices that we have decided to keep out of a solution pair. Note that the input instances for both of these problems are the same. Hence, due to Lemma 17, it is safe to assume that X is a minimum vertex cover and an independent set in G. In the next lemma we show that both problems are in fact equivalent.

▶ **Lemma 18** (⋆). *An instance $((G,k,d),X,\langle X_L,X_R\rangle)$ is a* Yes*-instance of* Annotated Contraction(vc) *if and only if it is a* Yes*-instance of* Constrained MaxCut*.*

Consider an instance $((G,k,d),X,\langle X_L,X_R\rangle)$ of Constrained MaxCut. We consider the following two cases: (1) $k = d$, and (2) $d < k < 2d$. (Recall that we are in the case where $k < 2d$.) The first case, as we will see, allows us to impose additional restrictions on the vertices that are in V_R. It also helps us to set up some conditions such that, if they are satisfied while running the algorithm, then it can terminate and safely conclude that the input is a Yes-instance. In the second case, even for $k = d+1$, we do not have these privileges. We deal with each of the two cases separately. Namely, Lemma 19 states that if an input instance is of the second type, then we can construct a collection of $2^{\mathcal{O}(d)} \cdot n^{k-d}$

many instances of the first type such that the input instance is a YES-instance if and only if at least one of these newly created instances is a YES-instance. We remark that this is the only place, in the whole algorithm, where an n^{k-d}-factor appears in the running time. Recall that Theorem 2 implies that this factor is unavoidable. After this lemma we present an algorithm to solve the instances that are of the first type.

▶ **Lemma 19** (⋆). *Suppose that there is an algorithm that, given an instance* $((G,k,d),X,\langle X_L,X_R\rangle)$ *of* CONSTRAINED MAXCUT *with a guarantee that* $k=d$*, runs in time* $f(n,k,d)$ *and correctly determines whether it is a* YES*-instance. Then, there is an algorithm that solves* CONSTRAINED MAXCUT *in time* $f(n,k,d)\cdot 2^{\mathcal{O}(d)}\cdot n^{k-d+1}$.

We proceed to present an algorithm to solve an instance $((G,k,d),X,\langle X_L,X_R\rangle)$ of CONSTRAINED MAXCUT with a guarantee that $k=d$. We first present a reduction rule to simplify these instances under the presence of a matching saturating X.

▶ **Reduction Rule 4.2.** *Consider an instance* $((G,k,d),X,\langle X_L,X_R\rangle)$ *of* CONSTRAINED MAXCUT *such that* $k=d$ *and* X *is an independent set in* G*. Let* M *be a matching in* G *saturating* X*.*

- *If there exists* $x\in X\setminus X_L$ *such that* $N(x)\setminus V(M)\neq\emptyset$*, then add* x *to* X_L*.*
- *If there exists* $x\in X_L$ *such that* $N(x)\setminus V(M)\neq\emptyset$*, then delete all vertices in* $N(x)\setminus V(M)$*.*

Return instance $((G',k,d),X,\langle X'_L,X_R\rangle)$ *where* $G'=G-(N(x)\setminus V(M))$ *and* $X'_L=X_L\cup\{x\}$*.*

▶ **Lemma 20** (⋆). *Reduction Rule 4.2 is safe.*

The full version contains an informal description of the algorithm for CONSTRAINED MAXCUT with a guarantee that $k=d$. We now briefly describe the algorithm formally.

We consider directed graphs that can have parallel arcs. We define the *rank* of a digraph, and the rank of a subset of its vertices or arcs using its underlying undirected graph.

CONSTRAINED DIGRAPH MAXCUT
Input: A digraph D, a tuple $\langle X_L,X_R\rangle$ of disjoint subsets of X, and an integer k.
Question: Does there exist a partition (V_L,V_R) of $V(G)$ such that (i) $A(V_R,V_L)=\emptyset$, (ii) $\mathsf{rank}(A(V_L,V_R))\geq k$, and (iii) $X_L\subseteq V_L$ and $X_R\subseteq V_R$?

We say that a partition $\langle V_L,V_R\rangle$ is a *solution* of $(D,\langle X_L,X_R\rangle,k)$ if it satisfies all the three conditions in the statement of the problem. We present a reduction that, given an instance $((G,k,d),X,\langle X_L,X_R\rangle)$ of CONSTRAINED MAXCUT, returns an instance $(D,\langle X_L,X_R\rangle,k)$ of CONSTRAINED DIRECTED MAXCUT. The reduction takes as input an instance $((G,k,d),X,\langle X_L,X_R\rangle)$ of CONSTRAINED MAXCUT on which Reduction Rule 4.2 is not applicable. It starts with a copy of the graph G and constructs a digraph D. The reduction finds (in polynomial time) a matching M in G that saturates all vertices in X. For every $xy\in E(G)$, where $x\in X$ and $y\in Y$, it deletes edge xy and adds arc (x,y) (i.e., it directs edges from X to Y). For every arc (x,y) in M, it adds arc (x_1,x) for every in-neighbour x_1 of y, and then deletes vertex y. This completes the construction of D. The reduction returns $(D,\langle X_L,X_R\rangle,k)$ as the instance of CONSTRAINED DIGRAPH MAXCUT.

▶ **Lemma 21** (⋆). $((G,k,d),X,\langle X_L,X_R\rangle)$ *is a* YES*-instance of* CONSTRAINED MAXCUT *if and only if* $(D,\langle X_L,X_R\rangle,k)$ *is a* YES*-instance of* CONSTRAINED DIGRAPH MAXCUT.

We present a dynamic programming algorithm for CONSTRAINED DIRECTED MAXCUT.

▶ **Lemma 22** (⋆). *There is an algorithm that, given an instance* $(G,\langle X_L,X_R\rangle,k)$ *of* CONSTRAINED DIRECTED MAXCUT*, runs in time* $2^{\mathcal{O}(k)}\cdot n^{\mathcal{O}(1)}$ *and correctly determines whether it is a* YES*-instance.*

We finally have all the ingredients to prove Lemma 14, and consequently Theorem 3 as well. The proofs are provided in the full version.

5 Conclusion

In this article we considered the problem of reducing the size of a minimum vertex cover of a graph G by at least d using at most k edge contractions. Note that the problem is trivial when $k < d$. A few simple observations prove that when $d \leq 2k$, the problem is coNP-hard and FPT when parameterized by $k + d$. Almost all of our technical work is to handle the case when $d \leq k < 2d$. We proved that the problem is NP-hard when $k = d + \frac{\ell-1}{\ell+3} \cdot d$ for any integer $\ell \geq 1$ such that k is an integer (in particular, $\ell = 1$). This implies that the problem is hard for various values of $k - d$ in the set $\{0, 1, \ldots, d-1\}$. We were able to prove that if $(k - d)$ is a constant then the problem is FPT when parameterized by $k + d$. However, if no such a condition is imposed, then the problem is W[1]-hard. More precisely, we presented an algorithm with running time $2^d \cdot n^{k-d+\mathcal{O}(1)}$ and proved that the problem is W[1]-hard when parameterized by $k + d$ in the case where $k - d = \frac{d}{3}$ (see the proof of Theorem 2).

We believe that it should be possible to prove that the problem is NP-hard for *every* value of $k - d$ in the set $\{0, 1, \ldots, d-1\}$. Such a reduction has the potential to sharpen the distinction between FPT and W[1]-hard cases as $k - d$ varies in this range. It might also simplify the analysis of our XP algorithm or lead to a simpler algorithm. It would be interesting to analyze the parameterized complexity of the problem with respect to structural parameters like the vertex cover number or the treewidth of the input graph. Note that the problem is trivially FPT when parameterized by the vertex cover number. Finally, it is worth mentioning that we did not focus on optimizing the degree of the polynomial term $n^{\mathcal{O}(1)}$ in our XP algorithm, although it is reasonably small.

References

1. Hans L. Bodlaender, Pinar Heggernes, and Daniel Lokshtanov. Graph Modification Problems (Dagstuhl Seminar 14071). *Dagstuhl Reports*, 4(2):38–59, 2014. doi:10.4230/DagRep.4.2.38.
2. Bruno Courcelle. The Monadic Second-Order Logic of Graphs. I. Recognizable Sets of Finite Graphs. *Information and Computation*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.
3. Christophe Crespelle, Pål Grønås Drange, Fedor V. Fomin, and Petr A. Golovach. A survey of parameterized algorithms and the complexity of edge modification, 2020. arXiv:2001.06867.
4. Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
5. Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012. URL: https://dblp.org/rec/books/daglib/0030488.bib.
6. Öznur Yaşar Diner, Daniël Paulusma, Christophe Picouleau, and Bernard Ries. Contraction and deletion blockers for perfect graphs and H-free graphs. *Theoretical Computer Science*, 746:49–72, 2018. doi:https://doi.org/10.1016/j.tcs.2018.06.023.
7. Esther Galby, Paloma T. Lima, Felix Mann, and Bernard Ries. Using edge contractions to reduce the semitotal domination number, 2021. arXiv:2107.03755.
8. Esther Galby, Paloma T. Lima, and Bernard Ries. Reducing the domination number of graphs via edge contractions and vertex deletions. *Discrete Mathematics*, 344(1):112169, 2021. doi:10.1016/j.disc.2020.112169.
9. Esther Galby, Felix Mann, and Bernard Ries. Blocking total dominating sets via edge contractions. *Theoretical Computer Science*, 877:18–35, 2021. doi:https://doi.org/10.1016/j.tcs.2021.03.028.

10 Esther Galby, Felix Mann, and Bernard Ries. Reducing the domination number of (P_3+kP_2)-free graphs via one edge contraction. *Discrete Applied Mathematics*, 305:205–210, 2021. doi:https://doi.org/10.1016/j.dam.2021.09.009.

11 Subhash Khot and Venkatesh Raman. Parameterized complexity of finding subgraphs with hereditary properties. *Theoretical Computer Science*, 289(2):997–1008, 2002. doi:10.1016/S0304-3975(01)00414-5.

12 Paloma T. Lima, Vinícius Fernandes dos Santos, Ignasi Sau, and Uéverton S. Souza. Reducing graph transversals via edge contractions. *Journal of Computer and System Sciences*, 120:62–74, 2021. doi:https://doi.org/10.1016/j.jcss.2021.03.003.

13 Daniël Paulusma, Christophe Picouleau, and Bernard Ries. Critical vertices and edges in H-free graphs. *Discrete Applied Mathematics*, 257:361–367, 2019. doi:10.1016/j.dam.2018.08.016.

14 Saket Saurabh, Uéverton dos Santos Souza, and Prafullkumar Tale. On the parameterized complexity of grid contraction. In *Proc. of the 17th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, volume 162 of *LIPIcs*, pages 34:1–34:17, 2020. doi:10.4230/LIPIcs.SWAT.2020.34.

15 Toshimasa Watanabe, Tadashi Ae, and Akira Nakamura. On the NP-hardness of edge-deletion and -contraction problems. *Discrete Applied Mathematics*, 6(1):63–78, 1983. doi:10.1016/0166-218X(83)90101-4.

16 Mihalis Yannakakis. Node- and Edge-Deletion NP-Complete Problems. In *Proc. of the 10th Annual ACM Symposium on Theory of Computing (STOC)*, pages 253–264, 1978. doi:10.1145/800133.804355.