On the Parameterized Complexity of Maximum Degree Contraction Problem

Saket Saurabh
The Institute Of Mathematical Sciences, HBNI, Chennai, India
University of Bergen, Norway
saket@imsc.res.in

Prafullkumar Tale
CISPA - Helmholtz Center for Information Security, Saarbrücken, Germany
prafullkumar.tale@cispa.saarland

Abstract

In the Maximum Degree Contraction problem, input is a graph G on n vertices, and integers k, d, and the objective is to check whether G can be transformed into a graph of maximum degree at most d, using at most k edge contractions. A simple brute-force algorithm that checks all possible sets of edges for a solution runs in time $n^{\mathcal{O}(k)}$. As our first result, we prove that this algorithm is asymptotically optimal, upto constants in the exponents, under Exponential Time Hypothesis (ETH).

Belmonte, Golovach, van't Hof, and Paulusma studied the problem in the realm of Parameterized Complexity and proved, among other things, that it admits an FPT algorithm running in time $(d+k)^{2k} \cdot n^{\mathcal{O}(1)} = 2^{\mathcal{O}(k \log(k+d))} \cdot n^{\mathcal{O}(1)}$, and remains NP-hard for every constant $d \geq 2$ (Acta Informatica (2014)). We present a different FPT algorithm that runs in time $2^{\mathcal{O}(dk)} \cdot n^{\mathcal{O}(1)}$. In particular, our algorithm runs in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$, for every fixed d. In the same article, the authors asked whether the problem admits a polynomial kernel, when parameterized by $k+d$. We answer this question in the negative and prove that it does not admit a polynomial compression unless NP $\subseteq$ coNP/*poly*.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability

Keywords and phrases Graph Contraction Problems, FPT Algorithm, Lower Bound, ETH, No Polynomial Kernel

Digital Object Identifier 10.4230/LIPIcs.IPEC.2020.26

Related Version Full version: `http://arxiv.org/abs/2009.11793`.

Funding *Saket Saurabh*: This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 819416), and Swarnajayanti Fellowship (No DST/SJF/MSA01/2017-18).
Prafullkumar Tale: This research is a part of a project that 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 We want to thank the anonymous reviewers for their valuable feedback.

1 Introduction

For any graph class $\mathcal{H}$, the $\mathcal{H}$-Modification problem takes as input a graph G and an integer k, and asks whether one can make at most k modifications in G such that the resulting graph is in $\mathcal{H}$. These types of modification problems are one of the central problems in graph theory and have received a considerable attention in algorithm design. With appropriate choice of $\mathcal{H}$ and allowed modification operations, $\mathcal{H}$-Modification can encapsulate well studied problems like Vertex Cover, Chordal Completion, Cluster Editing, Hadwinger Number, etc. Some natural and well-studied graph modification operations are vertex

deletion, edge deletion, edge addition, and edge contraction. The focus of the vast majority of papers on graph modification problems has been to the first three operations. Consider an example of $\mathcal{H}_{\leq d}$-Modification problem where $\mathcal{H}_{\leq d}$ is the collection of all graphs that has maximum degree at most d. If allowed modification operation is vertex deletion then we know the problem as Bounded Degree Deletion (BDD) and if it is edge contraction then as Maximum Degree Contraction (MDC). The complexity of BDD and several of its variants has been extensively studied [7, 9, 10, 13, 15, 16, 18, 25, 31] whereas, to the best of our knowledge, only [8] addressed MDC. In this article, we enhance our understanding of the second problem and answer an open question stated in [8].

The *contraction* of edge uv in simple graph G deletes vertices u and v from G, and replaces them by a new vertex, which is made adjacent to vertices that were adjacent to either u or v. For a set of edges F in $E(G)$, we denote the graph obtained from G by contracting all edges in F by G/F. In the $\mathcal{H}$-Contraction problem, an input is a graph G and an integer k, and the aim is to decide whether there is a set F of at most k edges in G such that G/F is in $\mathcal{H}$. Early papers by Watanabe et al. [33, 34] and Asano and Hirata [6] showed that $\mathcal{H}$-Contraction is NP-Hard for simple graph classes like trees, paths, stars, etc. Brouwer proved that it is NP-Hard even to decide whether a graph can be contracted to a path of length four [11]. Note that this problem admits a simple polynomial time algorithm if we consider any other modification operation. This has been a recurring theme in graph modification problems. For the same target graph class, edge contraction problem tends to more difficult than their counterparts where modification operation is vertex/edge addition/deletion. This difficulty is evident even in the realm of the Parameterized Complexity and Exact Exponential Algorithms.

In Parameterized Complexity, $\mathcal{H}$-Contraction problems are studied with the number of edges allowed to contract, k, as parameter. Heggernes et al. [24] proved that if $\mathcal{H}$ is the set of acyclic graphs then $\mathcal{H}$-Contraction is FPT but does not admit a polynomial kernel unless NP $\subseteq$ coNP/*poly*. The vertex deletion version of the problem, known as Feedback Vertex Set, admits a polynomial kernel. Series of papers studied the parameterized complexity for various graph classes like generalization and restrictions of trees [1, 3], cactus [26], bipartite graphs [21, 23], planar graphs [20], grids [32], cliques [12], split graphs [4], chordal graphs [29], bi-cliques [30], degree constrained graph classes [8, 19], etc. Krithika et al. [27] and Gunda et al. [22] studied $\mathcal{H}$-Contraction problems from the lenses of FPT approximation and lossy kernelization. Agarwal et al. [2] broke the 2^n-barrier for Path Contraction whereas Fomin et al. [17] showed that brute-force algorithms for Hadwinger Number problem and various other $\mathcal{H}$-Contraction problem are optimal under ETH.

Belmonte et al. [8] studied the parameterized complexity of $\mathcal{H}$-Contraction for three different classes $\mathcal{H}$: the class of graphs with maximum degree at most d, the class of d-regular graphs, and the class of d-degenerate graphs. They classified the parameterized complexity of all three problems with respect to the parameters k, d, and $d + k$. The first problem, also known as MDC, is defined as follows.

Maximum Degree Contraction	**Parameter:** $k + d$
Input: Graph G, integers k, d	
Question: Does there exist a subset F of $E(G)$ of size at most k such that every vertex in G/F has degree at most d?	

The authors proved that MDC is FPT when parameterized by $k + d$, W[2]-Hard when parameterized by k (even when restricted to split graphs), and para-NP-Hard when parameterized by d. Note that the problem is trivially solvable in polynomial time when $d \leq 1$ and NP-Hard for every constant $d \geq 2$.

Consider brute-force algorithm for MDC that given an instance (G, k, d), where graph G has n vertices, enumerates all subsets of edges of size at most k in G and for each subset contracts all edges in it to check whether the resulting graph has degree at most d. This algorithm runs in time $n^{\mathcal{O}(k)}$. Our first results states that this algorithm is optimal, up to constants in the exponents, under ETH.

▶ **Theorem 1.** *Unless ETH fails, there is no algorithm that given any instance (G, k, d) of* Maximum Degree Contraction *runs in time $n^{o(k)}$ and correctly determines whether it is a* Yes *instance.*

Belmonte et al. [8] presented an FPT algorithm for MDC that runs in time $(d+k)^{2k} \cdot n^{\mathcal{O}(1)}$. As for any non-trivial instance $d + k$ is smaller than n, we can conclude that there is no algorithm that given any instance (G, k, d) of MDC runs in time $(d + k)^{o(k)} \cdot n^{\mathcal{O}(1)}$ and correctly determines whether it is a Yes instance, unless ETH fails.

We remark that that the lower bound in Theorem 1 does not hold when d is a fixed constant and not a part of input. Hence, it is possible that MDC admits an algorithm that runs in time $k^{o(k)} \cdot n^{\mathcal{O}(1)}$ for a constant value of d. Belmonte et al. [8] proved that MDC problem admits linear vertex kernels on connected graphs when $d = 2$. This linear kernel leads to an FPT algorithm[1] running in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$. This hints that it is possible to design a better FPT algorithm for small values of d. Our second result shows that this is indeed the case.

▶ **Theorem 2.** *There is an algorithm that given an instance (G, k, d) of* Maximum Degree Contraction *runs in time $2^{\mathcal{O}(dk)} \cdot n^{\mathcal{O}(1)}$ and correctly determines whether it is a* Yes *instance.*

We note that the reduction used in [8] to prove that MDC is NP-Hard for any constant $d \geq 2$ implies that there is no $2^{o(dk)}$ algorithm for this problem.

Next, we look at the kernelization of MDC. Belmonte et al. [8] left it as an open question to determine whether MDC admits a polynomial kernel when parameterized by $k + d$. Our last result answers this question in negative.

▶ **Theorem 3.** *Unless* NP $\subseteq$ coNP/*poly, Maximum Degree Contraction, parameterized by $k + d$, does not admit a polynomial compression.*

It is known that the Bounded Degree Deletion problem admits a kernel with $\mathcal{O}(d^3k)$ vertices [16]. Hence, $\mathcal{H}_{\leq d}$-Modification is another example for which changing the modification operations from vertex deletion to edge contraction changes the compressibility drastically.

Due to space constraints, we omit formal proofs of Theorem 1 and 3. **These proofs can be found in the full version of the paper.** In Section 2, we present some preliminaries. In Section 3, we give a reduction from $(k \times k)$-Permutation Independent Set to MDC. We present an FPT algorithm using universal sets and branching techniques in Section 4. In Section 5, we present a sketch of reduction from Red Blue Dominating Set to MDC. We conclude this article with an open question in Section 6.

2 Preliminaries

For a positive integer q, we denote set $\{1, 2, \ldots, q\}$ by $[q]$.

[1] The algorithm colors vertices in the reduced instance with two colors and contracts each connected component in the colored subgraphs.

Graph Theory. In this article, we consider simple graphs with a finite number of vertices. For an undirected graph G, sets $V(G)$ and $E(G)$ denote its set of vertices and edges, respectively. Unless otherwise specified, we use n to denote the number of vertices in the input graph G. We denote an edge with two endpoints u, v as (u, v). Two vertices u, v in $V(G)$ are *adjacent* to each other if there is an edge (u, v) in $E(G)$. The open neighborhood of a vertex v, denoted by $N_G(v)$, is the set of vertices adjacent to v and its degree $\deg_G(v)$ is $|N_G(v)|$. The closed neighborhood of a vertex v, denoted by $N_G[v]$, is the set $N(v) \cup \{v\}$. We omit the subscript in the notation for neighborhood and degree if the graph under consideration is clear. For a subset S of $V(G)$, we define $N[S] = \bigcup_{v \in S} N[v]$ and $N(S) = N[S] \setminus S$. For a subset F of edges, a subset of vertices $V(F)$ denotes the collection of endpoints of edges in F. We say a set of edges F *spans* a set of vertices S if $S \subseteq V(F)$. For a subset S of $V(G)$, we denote the graph obtained by deleting S from G by $G - S$ and the subgraph of G induced on the set S by $G[S]$. For two subsets S_1, S_2 of $V(G)$, edge set $E(S_1, S_2)$ denotes the edges with one endpoint in S_1 and another one in S_2. We say S_1, S_2 are adjacent if $E(S_1, S_2)$ is non empty. For an integer q, a q-coloring of graph G is a function $\phi : V(G) \to [q]$. A *proper coloring* of G is a q-coloring ϕ of $V(G)$ for some integer q such that for any edge (u, v), $\phi(u) \neq \phi(v)$. There is a proper coloring of the graph with $\Delta(G) + 1$ many colors which can found in polynomial time. A set of vertices S is said to be *independent set* if no two vertices in S are adjacent to each other. A set of edges F is called *matching* if no two edges in F share an endpoint. A graph is called *connected* if there is a path between every pair of distinct vertices. A subset S of $V(G)$ is said to be a *connected set* if $G[S]$ is connected. A *spanning tree* of a connected graph is its connected acyclic subgraph, which includes all the vertices of the graph.

Graph Contraction. The *contraction* of an edge uv in G deletes vertices u and v from G, and adds a new vertex which is adjacent to vertices that were adjacent to either u or v. This process does not introduce self-loops or parallel edges. The resulting graph is denoted by G/e. For a graph G and edge $e = uv$, we formally define G/e in the following way: $V(G/e) = (V(G) \cup \{w\}) \setminus \{u, v\}$ and $E(G/e) = \{xy \mid x, y \in V(G) \setminus \{u, v\}, xy \in E(G)\} \cup \{wx \mid x \in N_G(u) \cup N_G(v)\}$. Here, w is a new vertex. An edge contraction reduces the number of vertices in a graph by exactly one. Several edges might disappear because of one edge contraction. For a subset of edges F in G, graph G/F denotes the graph obtained from G by contracting each connected component in the sub-graph $G' = (V(F), F)$ to a vertex.

We now formally define a contraction of graph G to another graph H.

▶ **Definition 4** (Graph Contraction). *A graph G is said to be* contractible *to graph H if there is a function $\psi : V(G) \to V(H)$ such that following properties hold.*

1. *For any vertex h in $V(H)$, set $W(h) := \{v \in V(G) \mid \psi(v) = h\}$ is not empty and graph $G[W(h)]$ is connected.*
2. *For any two vertices h, h' in $V(H)$, edge hh' is present in H if and only if $E(W(h), W(h'))$ is not empty.*

We say graph G is contractible to H via mapping ψ. For a vertex h in H, set $W(h)$ is called a *witness set* associated with or corresponding to h. We define the *H-witness structure* of G, denoted by $\mathcal{W}$, as a collection of all witness sets. Formally, $\mathcal{W} = \{W(h) \mid h \in V(H)\}$. A witness structure $\mathcal{W}$ is a partition of vertices in G. If a *witness set* contains more than one vertex, then we call it *big* witness set, otherwise it is *small* witness set.

If graph G has a H-witness structure, then graph H can be obtained from G by a series of edge contractions. For a fixed H-witness structure, let F be the union of spanning trees of all witness sets. By convention, the spanning tree of a singleton set is the empty set. To obtain

graph H from G, it is sufficient to contract edges in F. Hence, $H = G/F$. For a G/F-witness structure $\mathcal{W}$ of G, there is a unique function $\psi : V(G) \to V(G/F)$ corresponding to it. We say graph G is *k-contractible* to H if the cardinality of F is at most k. In other words, H can be obtained from G by at most k edge contractions.

Maximum Degree Contraction. In this subsection, we state an observation related to MDC. We say a set of edges F is a *solution* to instance (G, k, d) if the number of edges in F is at most k and the maximum degree of graph G/F is at most d. The number of edges that we are allowed to contract, k, is also called *solution size*. The following observation specifies how a solution behaves locally.

▶ **Observation 5.** *Consider a* Yes *instance (G, k, d) of MDC and let v be a vertex of degree at least $d+1$ in G. Then, for any solution F to (G, k, d), there are at least two vertices in $N[v]$ that are in the same witness set in the G/F-witness structure of G.*

Proof. Let G is contractible to a graph G/F, via mapping ψ. Assume, for the sake of contradiction, that no two vertices in $N[v]$ are in the same witness set. This implies $|N[v]| = |\psi(N[v])|$, where $\psi(N_G[v]) = \bigcup_{u \in N_G[v]} \psi(u)$. As $\psi(N_G[v]) \subseteq N_{G/F}(\psi(v))$ and $|N[v]| > d+1$, vertex $\psi(v)$ is adjacent with $d+1$ or more vertices in G/F. This contradicts the fact that the maximum degree of vertices in G/F is at most d. Hence, our assumption was wrong and there are at least two vertices in $N[v]$ that are in some big-witness set in G/F-witness structure of G. ◀

3 A Lower Bound for the Algorithm

We present a reduction from $(k \times k)$-Permutation Independent Set (PIS) problem to Maximum Degree Contraction problem. In the $(k \times k)$-PIS problem we are given a graph H on a vertex set $[k] \times [k]$. In other words, the vertex set is formed by a $k \times k$ table. We denote vertices in the table by $v[i, j]$ for $1 \leq i, j \leq k$. The question is whether there exists an independent set X in H that contains exactly one vertex from each row and each column of the table. In other words, for every $i, j \in [k]$ there is exactly one element of X that has i on the first coordinate and j on the second coordinate. Note that without loss of generality we may assume that each row and each column of the table forms an independent set.

Reduction. The reduction accepts an instance, say (H, k), of $(k \times k)$-Permutation Independent Set as an input. Here, H is a graph with vertex set formed by a $k \times k$ table. The reduction modifies a copy of the graph H in the following way.

- It adds a vertex corresponding to each row in the table and makes it adjacent with all vertices in that row. Let $R = \{r_1, r_2, \ldots, r_k\}$ be the set of vertices corresponding to rows.
- It adds a vertex corresponding to each column in the table and makes it adjacent with all vertices in that column. Let $C = \{c_1, c_2, \ldots, c_k\}$ be the set of vertices corresponding to columns.
- It adds set $S = \{s_1, s_2, \ldots, s_k\}$ of k vertices. For every i in $[k]$, it makes s_i adjacent with every vertex in $V(H) \cup C$ and with r_i.
- For every vertex r_i in R, it adds k^2 pendant vertices and makes them adjacent with r_i.
- For every vertex c_j in C, it adds $(k^2 - k + 1)$ pendant vertices and makes them adjacent with c_j.

See Figure 1 for an illustration. Let G be the graph obtained from a copy of graph H with the above modifications. The algorithm returns $(G, k, k^2 + k)$ as instance of MDC.

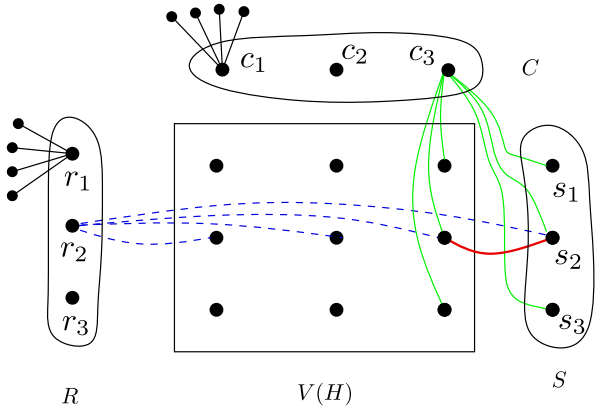

Figure 1 Dotted (blue) lines and thin (green) lines show the adjacency of vertices in R and C, respectively. Contracting the thick (red) edge $(v[2,3], s_2)$ represents selecting vertex $v[2,3]$ into the independent set. For the sake of clarity, we do not depict all edges present in the graph.

We present intuition of the proof of correctness. We describe how a solution, if it exists, to (G, k, d) leads to a solution to (H, k). We hope that this will also provide some intuition as to how a solution to (H, k) leads to a solution to (G, k, d). Note that S, C, R are independent sets in G. Every vertex in $R \cup C \cup S$ has degree $d+1$ and every vertex in $V(G) \setminus (R \cup C \cup S)$ has degree strictly less than d. We first argue that any solution for (G, k, d) can only contain edges in $E(G)$ that have one endpoint in $V(H)$ and another endpoint in S. Then, we prove that for every $i \in [k]$ a solution must pick an edge incident to some vertex in the i^{th} row and on s_i to reduce the degree of vertex r_i. We prove a similar statement for every column. Hence, for every $i \in [k]$, a solution contains an edge of the form $(v[i, j], s_i)$ for some $j \in [k]$. As there are at most k edges in a solution, every edge is of this form. For $i_1, i_2, j_1, j_2 \in [k]$, let $(v[i_1, j_1], s_{i_1})$ and $(v[i_2, j_2], s_{i_2})$ be two edges in a solution. We argue that if $(v[i_1, j_1], v[i_2, j_2])$ is an edge in G (and hence in H) then degrees of vertices obtained by contracting $(v[i_1, j_1], s_{i_1})$ and $(v[i_2, j_2], s_{i_2})$ are more than d. As this is true for any two arbitrary edges in the solution, their endpoints in $V(H)$ form an independent set in H.

We present a formal proof of correctness of this reduction in the full version of the paper. This reduction combined with the fact that unless ETH fails, $(k \times k)$-Permutation Independent Set can not be solved in time $k^{o(k)}$ [28] proves Theorem 1.

4 A Different FPT Algorithm

In this section, we present a different FPT algorithm for Maximum Degree Contraction. We introduce a variation of the problem called Labeled-Maximum Degree Contraction (Labeled-MDC). We present an FPT algorithm for Labeled-MDC and use it as a subroutine to present an FPT algorithm for MDC.

Informally, an instance of Labeled-MDC is an instance of MDC along with a labeling of vertices in the graph. Every vertex has a red or blue label. We are only interest in a solution that satisfies the following properties: (1) every edge has red labelled endpoints, and (2) for any red-labelled maximal connected component, a solution either spans none or all the vertices in that component. We remark that because of the second condition, this problem is *not* a restricted version of MDC. We formally define Labeled-MDC as follows.

Labeled-MDC	**Parameter:** $k+d$
Input: Graph G, a partition V_r, V_b of $V(G)$, and integers k, d	
Question: Does there exist a subset F of $E(G)$ of size at most k such that (a) every vertex in G/F has degree at most d; (b) $V(F) \subseteq V_r$; and (c) for a connected component C of $G[V_r]$, if $C \cap V(F) \neq \emptyset$ then $C \subseteq V(F)$.	

We say a set of edges F is a *solution* to instance $(G, (V_r, V_b), k, d)$ if the number of edges in F is at most k, the maximum degree of graph G/F is at most d, $V(F) \subseteq V_r$ and for a connected component C of $G[V_r]$, if $C \cap V(F) \neq \emptyset$ then $C \subseteq V(F)$.

It is easy to see that if $(G, (V_r, V_b), k, d)$ is a Yes instance of Labeled-MDC then (G, k, d) is a Yes instance of MDC. Let $\mathcal{U}$ be the family of all subsets of $V(G)$. If (G, k, d) is a Yes instance of MDC then $(G, (V_r, V(G) \setminus V_r), k, d)$ is a Yes instance of Labeled-MDC for some set V_r in $\mathcal{U}$. We use *universal sets* to construct a 'small' family of subsets of $V(G)$ that suffices for our purpose. We assume that there is a unique integer in $[n]$ for every vertex in $V(G)$. We use a subset of $[n]$ and a corresponding subset of $V(G)$ interchangeably.

▶ **Definition 6** (Universal Sets). *An (n, l)-universal set is a family $\mathcal{U}$ of subsets of $[n]$ such that for any $S \subseteq [n]$ of size l, the family $\{A \cap S \mid A \in \mathcal{U}\}$ contains all subsets of S.*

▶ **Proposition 7** ([5]). *For any $n, l \geq 1$ one can construct an (n, l)-universal set of size $2^{\mathcal{O}(l)} \cdot \log(n)$ in time $2^{\mathcal{O}(l)} \cdot n \log(n)$.*

In the following lemma, we argue that an FPT algorithm for Labeled-MDC leads to an FPT algorithm for MDC.

▶ **Lemma 8.** *Suppose there is an algorithm that given an instance $(G, (V_r, V_b), k, d)$ of Labeled-MDC runs in time $f(k, d) \cdot n^{\mathcal{O}(1)}$ and correctly determines whether it is a Yes instance. Then, there is an algorithm that given an instance (G, k, d) of MDC runs in time $2^{\mathcal{O}(dk)} \cdot f(k, d) \cdot n^{\mathcal{O}(1)}$ and correctly determines whether it is a Yes instance.*

Proof. Let $\mathcal{A}$ be an algorithm that given an instance $(G, (V_r, V_b), k, d)$ of Labeled-MDC runs in time $f(k, d) \cdot n^{\mathcal{O}(1)}$ and correctly determines whether it is a Yes instance. We first describe an algorithm for MDC that uses $\mathcal{A}$ as a subroutine. For the input (G, k, d), the algorithm constructs a $(U, 2k + kd)$-universal family $\mathcal{U}$ using Proposition 7. For every set V_r in $\mathcal{U}$, the algorithm runs Algorithm $\mathcal{A}$ with input $(G, (V_r, V(G) \setminus V_r), k, d)$. The algorithm returns Yes if Algorithm $\mathcal{A}$ returns Yes for one of these inputs otherwise it returns No. This completes the description of the algorithm. The running time of the algorithm follows from the description and Proposition 7. In the remaining proof, we argue the correctness of the algorithm. More precisely, we prove that (G, k, d) is a Yes instance of MDC if and only if there is a subset V_r in $\mathcal{U}$ such that $(G, (V_r, V(G) \setminus V_r), k, d)$ is a Yes instance of Labeled-MDC.

Suppose that (G, k, d) is a Yes instance of MDC and let F be a solution to it. Note that $|V(F)| \leq 2k$. We first argue that the number of vertices in $N(V(F))$ is at most kd. Let $\mathcal{W}$ be the G/F-witness structure of G and $\psi : V(G) \to V(G/F)$ be the corresponding function. Consider an arbitrary vertex v in $N(V(F))$. As v is not in $V(F)$, $\psi(v)$ corresponds to a small witness set in $\mathcal{W}$. As v is in $N(V(F))$, $\psi(v)$ is adjacent to a vertex in G/F that corresponds to a big witness set in $\mathcal{W}$. As $|F| \leq k$, there are at most k big witness sets in $\mathcal{W}$. Since the maximum degree of G/F is at most d, there are at most kd small witness sets in $\mathcal{W}$ that are adjacent with some big witness set. Hence, there are at most kd vertices in $N(V(F))$. As $\mathcal{U}$ is a $(n, 2k + dk)$-universal set and $|N[V(F)]| \leq 2k + dk$, there exists a set A in $\mathcal{U}$ such that the family $\{A \cap N[V(F)] \mid A \in \mathcal{U}\}$ contains all subsets of $N[V(F)]$. This implies, there exists a set, say V_r, such that $V_r \cap N[V(F)] = V(F)$. We argued that $(G, (V_r, V(G) \setminus V_r), k, d)$ is a Yes instance of Labeled-MDC.

Note that G/F has maximum degree at most d and $V(F) \subseteq V_r$. We need to prove that for a connected component C of $G[V_r]$ if $C \cap V(F) \neq \emptyset$ then $C \subseteq V(F)$. Assume that there exits a connected component C of $G[V_r]$ such that $C \cap V(F) \neq \emptyset$ and $C \setminus V(F) \neq \emptyset$. As C is a connected component and $C \cap V(F) \neq \emptyset$, there exists a vertex v in $C \setminus V(F)$ that is adjacent with some vertex in $V(F)$. Hence, there is a vertex in $N(V(F)) \cap V_r$. This contradicts the fact that $V_r \cap N[V(F)] = V(F)$. Hence, our assumption is wrong and $C \setminus V(F)$ is an empty set. This implies $(G, (V_r, V(G) \setminus V_r), k, d)$ is a YES instance of LABELED-MDC. As mentioned before, it is easy to see that if $(G, (V_r, V_b), k, d)$ is a YES instance of LABELED-MDC then (G, k, d) is a YES instance of MDC. This concludes the proof of the lemma. ◀

In the remaining section, we present a recursive algorithm for LABELED-MDC. We start with the following simple reduction rules.

▶ **Reduction Rule 9.** *For an instance $(G, (V_r, V_b), k, d)$, if the maximum degree of vertices in G is at most d and $k \geq 0$ then return a YES instance.*

It is easy to see that the first reduction rule is safe. Recall that a set of edges F is called solution to $(G, (V_r, V_b), k, d)$ if the number of edges in F is at most k, the maximum degree of graph G/F is at most d, $V(F) \subseteq V_r$, and for a connected component C of $G[V_r]$, if $C \cap V(F) \neq \emptyset$ then $C \subseteq V(F)$. Consider a connected component C of $G[V_r]$. If $|C| = 1$ then no solution edge can be incident to it. Also, if $|C| \geq 2k + 1$ then because of the last property and the fact that $|V(F)| \leq 2k$, no solution edge can be incident to vertices in C. These simple observations prove that the following reduction rule is safe.

▶ **Reduction Rule 10.** *For an instance $(G, (V_r, V_b), k, d)$, if there is a connected component, say C, of $G[V_r]$ such that $|C| = 1$ or $|C| \geq 2k + 1$ then move C from V_r to V_b i.e. return instance $(G, (V_r \setminus C, V_b \cup C), k, d)$.*

By Observation 5, vertex v in V_b can be adjacent to at most $d + k$ vertices in V_r. The following reduction rule ensures that the neighbors of v in V_r are not spread across many connected components.

▶ **Reduction Rule 11.** *For an instance $(G, (V_r, V_b), k, d)$, if there exists a vertex, say v, in V_b for which $N_G(v)$ intersects with $d+1$ different connected components of $G[V_r]$ then return a NO instance.*

▶ **Lemma 12.** *Reduction Rule 11 is safe.*

Proof. Assume that $(G, (V_r, V_b), k, d)$ is a YES instance. Let F be its solution and it contracts G to G/F via mapping ψ. Suppose $C_1, C_2, \ldots, C_{d+1}$ are connected components of $G[V_r]$ such that $C_i \cap N(v) \neq \emptyset$ for $i \in [d + 1]$. For every i, consider a vertex, say u_i, in $C_i \cap N(v)$. Let $U = \{u_1, u_2, \ldots, u_{d+1}\}$. Define $\psi(U) = \bigcup_{u \in U} \psi(u)$. For $i, j \in [d + 1]$, $i \neq j$ implies $\psi(u_i) \neq \psi(u_j)$ as C_i and C_j are two different connected components of $G[V_r]$ and $V(F) \subseteq V_r$. This implies $|\psi(U)| = |U| = d + 1$. As $V(F) \subseteq V_r$ and $v \in V_b$, F does not contain an edge incident on v. Hence, $\psi(v) \neq \psi(u_i)$ for any $i \in [d + 1]$. As $\psi(U) \subseteq N_{G/F}(\psi(v))$ and $|\psi(U)| \geq d + 1$, vertex $\psi(v)$ is adjacent with $d + 1$ or more vertices in G/F. This contradicts the fact that the maximum degree of vertices in G/F is at most d. Hence, our assumption was wrong and $(G, (V_r, V_b), k, d)$ is a NO instance. ◀

The algorithm exhaustively applies the reduction rules mentioned above. On a reduced instance, the algorithm creates multiple instances using the following subroutine. For an instance $(G, (V_r, V_b), k, d)$, a subset R of V_r, and a $(d + 1)$-coloring of R, the subroutine

creates a new instance by contracting each colored component of R into a single vertex, and (re-)label it blue. We need the notion of 'valid coloring' to filter out colorings that will not produce a 'smaller' instance. For graph H, a vertex coloring $\phi : V(H) \to [d+1]$ is said to be a *valid coloring* if every monochromatic connected component is of size at least two. We now describe the subroutine.

Subroutine `Colorwise-Contraction`**.** This subroutine takes as an input an instance $(G, (V_r, V_b), k, d)$ of Labeled-MDC, a non-empty subset R of V_r, and a valid coloring ϕ of $G[R]$. It returns another instance of Labeled-MDC. It initializes $G' = G$, $V'_r = V_r$, $V'_b = V_b$, and $k' = k$. For a monochromatic connected component C of $G[R]$, the subroutine finds a spanning tree of $G[C]$ and contracts all edges in it. Let v_C be the vertex obtained at the end of this series of edge contractions. It updates $V'_r = V_r \setminus C$, $V'_b = V_b \cup \{v_C\}$ and reduces k by $|C| - 1$. The subroutine repeats this procedure for every monochromatic connected component of $G[R]$. It returns $(G', (V'_r, V'_b), k', d)$ as instance of Labeled-MDC. This completes the description of the subroutine.

It is easy to verify that (V'_r, V'_b) is a partition of $V(G')$. As ϕ is a valid coloring of $G[R]$, a union of spanning trees of all monochromatic connected components of $G[R]$ contains at least $|R|/2$ edges. Hence, the subroutine contracts at least $|R|/2$ edges. This small observation will be helpful to get a bound on the running time of the algorithm.

▶ Remark 13. $k' \leq k - |R|/2$.

Let $\texttt{CC}[(G, (V_r, V_b), k, d); R; \phi]$ denote the instance returned by the subroutine when the input is $(G, (V_r, V_b), k, d)$, R, and ϕ. In the following lemma, we prove if the original instance is a Yes instance than at least one of the reduced instances is a Yes instance.

▶ **Lemma 14.** *Consider a* Yes *instance $(G, (V_r, V_b), k, d)$ of* Labeled-MDC*. Let R be a union of some connected components of $G[V_r]$. Suppose there is solution F to $(G, (V_r, V_b), k, d)$ such that $R \subseteq V(F)$. Then, there is a valid coloring $\phi : R \to [d+1]$ of $G[R]$ for which $\texttt{CC}[(G, (V_r, V_b), k, d); R; \phi]$ is a* Yes *instance.*

Proof. Let $H = G/F$. Consider the H-witness structure $\mathcal{W}$ of G and let G be contracted to H via ψ. Define a subset $\mathcal{W}_R$ of $\mathcal{W}$ as the collection of witness sets that intersects R. Formally, $\mathcal{W}_R = \{W \in \mathcal{W} \mid W \cap R \neq \emptyset\}$. Let $\mathcal{W}_R = \{W_1, W_2, \ldots, W_q\}$. For every $i \in [q]$, let h_i be the vertex corresponding to W_i. In other words, $W_i = \{v \in V(G) \mid \psi(v) = h_i\}$. Let $R_H = \{h_1, h_2, \ldots, h_q\}$.

Let F_1 be the collection of edges in F that are incident to some vertex in R. Hence, $R \subseteq V(F_1)$. As R is a union of connected components in $G[V_r]$ and $V(F_1) \subseteq V(F) \subseteq V_r$, we can conclude that $R = V(F_1) = \bigcup_{i \in [q]} W_i$. Hence, $\{W_1, W_2, \ldots, W_q\}$ is a partition of R. As there is a solution edge incident to every vertex in R, every witness set in $\mathcal{W}_R$ is a big witness set. This implies for every $i \in [q]$, there is a subset F_i of F such that $W_i = V(F_i)$. As the maximum degree of vertices in graph H is at most d, there is a proper $(d+1)$-coloring, say γ, of H. For $i, j \in [q]$, if (h_i, h_j) is an edge in H then $\gamma(h_i) \neq \gamma(h_j)$. Define a coloring $\phi : R \to [d+1]$ as follows. For $v \in R$, $\phi(v) = \gamma(h_i)$ where $v \in W_i$. As $\{W_1, W_2, \ldots, W_q\}$ is a partition of R, function ϕ is well defined. Since W_i is a big witness set, ϕ is a valid coloring.

By the construction of ϕ, any witness set in $\mathcal{W}$ is monochromatic. Since γ is a proper coloring of H, any two witness sets adjacent to each other have distinct colors. Hence, every witness set in $\mathcal{W}_R$ is a monochromatic connected component of coloring ϕ. As algorithm constructs every valid coloring of R, it also consider this coloring and create instance $(G', (V'_r, V'_b), k', d) = \texttt{CC}[(G, (V_r, V_b), k, d), R; \phi]$. For every $i \in [q]$, let F°_i be edges in a spanning tree of $G[W_i]$. Define $F^\circ = \bigcup_{i \in [q]} F^\circ_i$. As $W_i = V(F_i)$, graphs G/F_1 and G/F° are identical. Also, $|F^\circ_i| \leq |F_i|$ which implies $|F^\circ| \leq |F_1|$. Define $F^\star = (F \setminus F_1) \cup F^\circ$. It is easy to verify that $F^\star$ is also a solution to $(G, (V_r, V_b), k, d)$.

We now argue that $F^\star \setminus F^\circ$ is a solution to $(G', (V_r', V_b'), k', d)$. By the description of the algorithm, $k' = k - |F^\circ|$. As $|F^\star| \leq |F| \leq k$ and $F^\circ \subseteq F^\star$, we have $|F^\star \setminus F^\circ| \leq |F^\star| - |F^\circ| \leq k'$. Note that $G/F^\star = (G/F^\circ)/(F^\star \setminus F^\circ) = G'/(F^\star \setminus F^\circ)$ as $G' = G/F^\circ$. This implies the maximum degree of $G'/(F^\star \setminus F^\circ)$ is at most d. The only thing that remains to argue is that $V(F^\star \setminus F^\circ)$ is contained in V_r'. By construction, $F \setminus F_1 = F^\star \setminus F^\circ$. As F_1 is the set of edges in F that were incident to R, we can conclude that no edge in $F^\star \setminus F^\circ$ is incident to R. Recall that $V_r' = V_r \setminus R$. Hence, $V(F^\star \setminus F^\circ) \subseteq V_r'$. This implies that $F^\star \setminus F^\circ$ is a solution to $(G', (V_r', V_b), k', d)$ and concludes the proof of the lemma. ◀

In the above lemma, instead of considering any arbitrary subset V_r we only consider a subset that is the union of one or more connected components of $G[V_r]$. This suffices for our purpose as the algorithm calls the subroutine only on such subsets of V_r. Also, note that we do not need to know the solution F explicitly to apply the above lemma. It suffices to know that such a solution exists. We are now able to present an algorithm for Labeled-MDC

Algorithm for Labeled-MDC. The algorithm takes as input an instance $(G, (V_r, V_b), k, d)$ of Labeled-MDC and returns Yes or No. If $k < 0$ then the algorithm returns No. If $k = 0$ then it finds the maximum degree of G. If it is at most d then the algorithm returns Yes otherwise it returns No. The algorithm exhaustively applies Reduction Rules 9, 10, and 11. If the reduced instance is a trivial Yes (resp. No) instance then the algorithm returns Yes (resp. No). Otherwise, it creates multiple instances and makes recursive calls on these instances. The algorithm returns Yes if one of the recursive calls returns Yes, otherwise; it returns No.

We now describe the procedure used by the algorithm to create new instances. Let $(G, (V_r, V_b), k, d)$ be the instance on which reduction rules are not applicable. The algorithm finds a vertex, say v, in G such that $\deg_G(v) \geq d + 1$. It considers the following two cases.

1. (Vertex v is in V_r) Let R be the connected component of $G[V_r]$ that contains v. The algorithm constructs all valid colorings $\phi : R \to [d + 1]$ of $G[R]$. For each coloring, the algorithm calls subroutine `Colorwise-Contraction` with input $(G, (V_r, V_b), k, d)$, R, and ϕ. The algorithm calls itself with the instances returned by this subroutine as the input.
2. (Vertex v is in V_b) Let $C_1, C_2, \ldots, C_q$ be the connected components of $G[V_r]$ such that $N(v) \cap C_i \neq \emptyset$ for every $i \in [q]$. For a non-empty subset $I \subseteq [q]$, define $R_I := \bigcup_{i \in I} C_i$. For every non-empty subset $I \subseteq [q]$, the algorithm proceeds as follows. If $|R_I| \geq 2k + 1$, the algorithm discards this choice of I and moves to the next one. Otherwise, the algorithm constructs all valid coloring $\phi : R_I \to [d + 1]$ of $G[R_I]$. For each coloring, the algorithm calls subroutine `Colorwise-Contraction` with input $(G, (V_r, V_b), k, d)$, R_I, and ϕ. The algorithm calls itself with the instance returned by this subroutine as input.

This completes the description of the algorithm.

In the following lemma, we prove that the algorithm described above is correct and runs in the desired time.

▶ **Lemma 15.** *There is an algorithm that given an instance $(G, (V_r, V_b), k, d)$ of Labeled-MDC runs in time $2^{(d+2)k} \cdot (d + 1)^{2k} \cdot n^{\mathcal{O}(1)}$ and correctly determines whether it is a Yes instance.*

Proof. We argue that the algorithm described above solves Labeled-MDC in the desired time. We prove this lemma by the induction over the solution size k.

Consider the base case when the solution size is zero. Here, the algorithm finds a maximum degree of the graph and depending on its value returns Yes or No. It is easy to see that the lemma holds in this case. Assume that the lemma is true when the solution size is at most $k - 1$.

We first prove that given a YES instance the algorithm returns YES. Suppose $(G, (V_r, V_b), k, d)$ is a YES instance of LABELED-MDC and let F be its solution. Note that this implies that F is a solution to (G, k, d). If the algorithm returned YES because Reduction Rule 9 returned a YES instance then the lemma is vacuously true. By Lemma 12, Reduction Rule 11 is not applicable on the input. Consider the instance obtained by the exhaustive application Reduction Rules 9 and 10 on the input instance. For notational convenience, we denote this reduced instance by $(G, (V_r, V_b), k, d)$. As Reduction Rule 9 is not applicable, there is a vertex in G that has degree at least $d+1$. Let v be the vertex of degree at least $d+1$ found by the algorithm. By Observation 5, $V(F)$ intersects with $N[v]$.

Consider the case when v is in V_r and let R be the connected component of $G[V_r]$ that contains v. Since $V(F) \subseteq V_r$, we have $R \cap V(F) \neq \emptyset$. As F is a solution to $(G, (V_r, V_b), k, d)$, $R \cap V(F) \neq \emptyset$ implies $R \subseteq V(F)$. Instance $(G, (V_r, V_b), k, d)$, subset R of V_r, and solution F satisfies the premise of Lemma 14. Hence, there is a valid coloring $\phi : R \to [d+1]$ of $G[R]$ such that $\texttt{CC}[(G, (V_r, V_b), k, d), R; \phi]$ is a YES instance. As $R \neq \emptyset$, Remark 13 implies that $k' < k$. By the induction hypothesis, the algorithm correctly returns YES when the input is $(G', (V_r', V_b'), k', d)$. As one of the recursive calls returns YES, the algorithm returns YES when the input is $(G, (V_r, V_b), k, d)$ and v is in V_r.

Consider the case when v is in V_b. Let $C_1, C_2, \ldots, C_q$ be the connected components of $G[V_r]$ such that $N(v) \cap C_i \neq \emptyset$ for every $i \in [q]$. Recall that for a non-empty subset $I \subseteq [q]$, $R_I = \bigcup_{i \in I} C_i$. As $V(F)$ intersects $N[v]$ and $V(F) \subseteq V_r$, there exists a non-empty subset $I' \subseteq [q]$ such that for $i \in [q]$, $C_i \cap N(v) \neq \emptyset$ if and only if $i \in I'$. As F is a solution to $(G, (V_r, V_b), k, d)$, $C_i \cap V(F) \neq \emptyset$ implies $C_i \subseteq V(F)$. Hence, $R_{I'} \subseteq V(F)$. As $|V(F)| \leq 2k$, $|R_{I'}| \leq 2k$. For every non-empty subset $I \subseteq [q]$ for which $|R_I| \leq 2k$, the algorithm constructs all valid coloring $\phi : R_I \to [d+1]$ of $G[R_I]$ and calls `Colorwise-Contraction`. Instance $(G, (V_r, V_b), k, d)$, subset $R_{I'}$ of V_r, and solution F satisfies the premise of Lemma 14. Hence, there is a valid coloring $\phi : R_{I'} \to [d+1]$ of $G[R_{I'}]$ such that $(G', (V_r', V_b'), k', d) = \texttt{CC}[(G, (V_r, V_b), k, d), R_{I'}, \phi]$ is a YES instance. As $R \neq \emptyset$, Remark 13 implies that $k' < k$. By the induction hypothesis, the algorithm correctly returns YES when the input is $(G', (V_r', V_b'), k', d)$. As one of the recursive calls returns YES, the algorithm returns YES when the input is $(G, (V_r, V_b), k, d)$ and v is in V_b. This implies that if $(G, (V_r, V_b), k, d)$ is a YES instance then the algorithm returns YES.

We now prove that if the algorithm returns YES on instance $(G, (V_r, V_b), k, d)$ then it is a YES instance of LABELED-MDC. If the algorithm returned YES because Reduction Rule 9 returned a YES instance then the lemma is vacuously true. Otherwise, there is a newly created instance, say $(G', (V_r', V_b'), k', d)$, on which the recursive call of the algorithm returned YES. Let R be the subset of V_r and ϕ be its valid coloring such that `Colorwise-Contraction` returned this instance when input was $(G, (V_r, V_b), k, d)$, R, and ϕ. Let F° be the edges in G contracted by the subroutine to contract G'. In other words, F° is a collection of spanning trees of connected monochromatic components of $G[R]$. Note that $|F^\circ| = k - k'$. The algorithm calls `Colorwise-Contraction` only on non-empty subsets R. Hence, by Remark 13, $k' < k$. By the induction hypothesis, $(G', (V_r', V_b'), k', d)$ is a YES instance of LABELED-MDC. It is easy to see that if F' is a solution to $(G', (V_r', V_b'), k', d)$ then $F' \cup F^\circ$ is a solution to $(G, (V_r, V_b), k, d)$. This concludes the proof of the correctness of the algorithm.

We now bound the running time of the algorithm. The algorithm can apply all the reduction rules in polynomial time. It creates new instances only when none of the reduction rules are applicable. As Reduction Rules 10 is not applicable, any connected component of $G[V_r]$ has at least two and at most $2k$ vertices. In Case (1), the algorithm creates at most $(d+1)^{|R|}$ many instances. By Remark 13 and the induction hypothesis, the time taken by the algorithm in this case is

$$(d+1)^{|R|} \cdot 2^{(d+2)(k-|R|/2)} \cdot (d+1)^{2(k-|R|/2)} \cdot n^{\mathcal{O}(1)} \leq 2^{(d+2)k} \cdot (d+1)^{2k} \cdot n^{\mathcal{O}(1)}.$$

As Reduction Rule 11 is not applicable, for any vertex v in V_b, there are at most d connected components of $G[V_r]$ that intersects $N(v)$. In Case (2), the algorithm constructs all valid partitions of R_I only when $|R_I| \leq 2k$. Hence, in this case, the algorithm creates $2^d \cdot (d+1)^{|R|}$ many instances. By Remark 13 and the induction hypothesis, the time taken by the algorithm in this case is

$$2^d \cdot (d+1)^{|R|} \cdot 2^{(d+2)(k-|R|/2)} \cdot (d+1)^{2(k-|R|/2)} \cdot n^{\mathcal{O}(1)} \leq 2^{(d+2)k} \cdot (d+1)^{2k} \cdot n^{\mathcal{O}(1)}.$$

As $|R| \geq 2$, we have $2^d \cdot 2^{(d+2)(-|R|/2)} \leq 1$. This completes the proof of the lemma. ◀

The correctness of Theorem 2 immediately follows from Lemma 8 and Lemma 15.

5 No Polynomial Kernel

In this section, we present a sketch of a reduction from Red Blue Dominating Set (RBDS). In this problem, an input is comprised of a bipartite graph H with a bipartition (R, B) of $V(H)$, and a positive integer l. The question is, does there exist a subset R' of R of size at most l such that $N(R') = B$?

In this problem, an input comprises a bipartite graph H with a bipartition (R, B) of $V(H)$, and a positive integer l. The question is, does there exist a subset R' of R of size at most l such that $N(R') = B$? Without loss of generality, we can assume that $l + 3 < |B|$ and no vertex in R is adjacent to all but one vertices in B. We know the following result about the compression of the problem. See, for example, Theorem 15.18 in [14].

▶ **Proposition 16.** *Unless* NP ⊆ coNP/*poly, RBDS, parameterized by* $|B|$*, does not admit a polynomial compression.*

If $|R| > 2^{|B|}$ then there are at least two different vertices, say r_1, r_2 such that $N(r_1) = N(r_2)$. It is easy to see that it is safe to delete one of these two vertices. In this case, we can ensure, in polynomial time, that $|R| \leq 2^{|B|}$ by repeating the above process. This implies $\log_2 |R| \leq |B|$. Hence, we get the following corollary of Proposition 16.

▶ **Corollary 17.** *Unless* NP ⊆ coNP/*poly, RBDS, parameterized by* $|B| + \log_2 |R|$*, does not admit a polynomial compression.*

For the sake of clarity, we use both $|B|$ and $\log_2 |R|$ as parameters instead of replacing $\log_2 |R|$ by the larger parameter $|B|$. For notational convenience, we assume that $\log_2 |R|$ is an integer. If this is not the case, one can add some isolated vertices in R to ensure that $\log_2 |R|$ is an integer. This results in at most doubling of the number of vertices in it.

We first present an overview of the reduction. Consider an instance (H, R, B, l) of RBDS. See Figure 2 for an illustration. The reduction makes a copy of R and two copies of B, say B^1, B^2. For every vertex b in B, we denote its two copies in B^1, B^2 by b^1, b^2, respectively. For every edge (r, b), the reduction adds edges (r, b^1) and (r, b^2). It adds two independent sets U^1, U^2. For every vertex $u \in U^1 \cup U^2$, it adds some pendent vertices adjacent to it. The reduction adds all edges to make a complete bipartite graph with (B^1, U^1) as its bipartition. Similarly, it adds all edges to make a complete bipartite graph with (B^2, U^2) as its bipartition. For every vertex b in B, it adds a set of independent vertices X_b. For every x in X_b, it adds some pendent vertices adjacent to it and adds edges (b^1, x), (b^2, x). We briefly present an intuition behind the construction before presenting the last step. Let G be the graph constructed so far and k, d be two integers whose values depend only on $|B|, \log_2 |R|$. Suppose the reduction returns (G, k, d) as an instance of MDC.

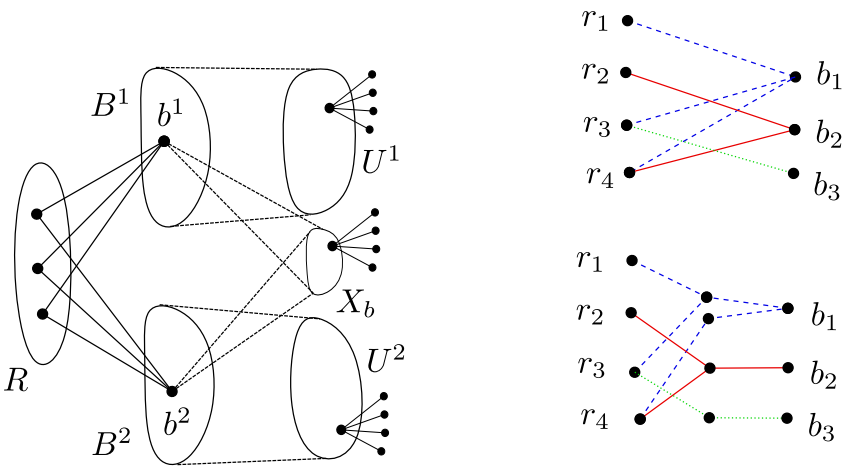

Figure 2 (Left) Overview of the reduction. The doted lines indicate that there is a complete bipartite graph cross two sets. (Right) The operation of replacing edges incident to vertex in B by a tree rooted at that vertex.

We set the value of d and the number of pendant vertices such that it is ensured that the only vertices in $U^1 \cup U^2 \cup X_b$ have degree more than d in G. We fix k and the sizes of sets U^1, U^2, X_b to ensure that any solution for the reduced instance of MDC satisfy the following properties.

1. It does not include an edge with one of its endpoints in $B^1 \cup B^2$ and another in $U^1 \cup U^2$.
2. For any b in B, it does not include an edge with one of its endpoints in $\{b^1, b^2\}$ and another in X_b.
3. It spans all vertices in $B^1 \cup B^2$. In other words, $B^1 \cup B^2 \subseteq V(F)$.
4. There are at most l witness sets in the G/F-witness structure of G that contain vertices in B^1 (similarly in B^2).
5. For every b in B, F includes b^1, b^2 in the same witness set.

Property (4) ensures that the degree constraints for the vertices in U^1 (similarly in U^2) are satisfied. Property (5) ensures that for every b in B, the degree constraints for the vertices in X_b are satisfied. Because of Property (1) and (2), only the vertices in R can make a witness set connected. Hence, each witness set should contain at least one vertex from R. We set the budget k such that each witness set contains exactly one vertex from R. To prove connectivity to witness set, this vertex needs to be adjacent to all vertices in that witness set. Hence, the set of endpoints of edges in a solution to (G, k, d) contains at most l vertices in R that dominates B. This naturally leads to a solution to $(H.R, B, l)$.

We now present the last step in the construction. The degree of the vertices formed by contracting a witness set can be larger than d. To avoid this, we replace star centered at b and whose leaves are in R by a binary tree rooted at that vertex. We ensure that for every edge incident b, there is a unique root-to-leaf path in the binary tree rooted at b and vice versa.

We present a formal reduction and its proof of correctness in the full version of the paper. This reduction combined with the fact that unless NP $\subseteq$ coNP/*poly*, RBDS, parameterized by $|B|$, does not admit a polynomial compression leads to a proof of Theorem 3.

6 Conclusion

In this article, we studied Maximum Degree Contraction problem. We prove that a simple brute force algorithm for this problem is optimal under ETH. This lower bound also implies that the known FPT algorithm with running time $(d+k)^k \cdot n^{\mathcal{O}(1)}$ is also optimal under the same hypothesis. We compliment this result by presenting another FPT algorithm with running time $2^{\mathcal{O}(dk)} \cdot n^{\mathcal{O}(1)}$. While these two FPT algorithms are incomparable, our algorithm runs faster for smaller values of d, for which the problem still remains NP-Hard. We also prove that unless NP $\subseteq$ coNP/*poly*, the problem does not admit a polynomial compression when parameterized by $k + d$.

Most of the $\mathcal{H}$-Contraction problems do not admit a polynomial kernel under the same complexity conjecture. For some graph classes like trees, cactus, cliques, splits graphs, such negative results have been complimented by establishing a *lossy kernel* of polynomial size for these problems. There are also examples like Chordal Contraction, s-Club Contraction (for $s \geq 2$) for which we know that lossy kernel of polynomial size do not exist. We conclude this article with following open question: Does Maximum Degree Contraction admit a lossy kernel of polynomial size?

References

1 Akanksha Agarwal, Saket Saurabh, and Prafullkumar Tale. On the parameterized complexity of contraction to generalization of trees. *Theory of Computing Systems*, 63(3):587–614, 2019.

2 Akanksha Agrawal, Fedor V Fomin, Daniel Lokshtanov, Saket Saurabh, and Prafullkumar Tale. Path contraction faster than 2^n. *SIAM Journal on Discrete Mathematics*, 34(2):1302–1325, 2020.

3 Akanksha Agrawal, Lawqueen Kanesh, Saket Saurabh, and Prafullkumar Tale. Paths to trees and cacti. In *International Conference on Algorithms and Complexity*, pages 31–42. Springer, 2017.

4 Akanksha Agrawal, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. Split contraction: The untold story. *ACM Transactions on Computation Theory (TOCT)*, 11(3):1–22, 2019.

5 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *Journal of the ACM (JACM)*, 42(4):844–856, 1995.

6 Takao Asano and Tomio Hirata. Edge-contraction problems. *Journal of Computer and System Sciences*, 26(2):197–208, 1983.

7 Balabhaskar Balasundaram, Shyam Sundar Chandramouli, and Svyatoslav Trukhanov. Approximation algorithms for finding and partitioning unit-disk graphs into co-k-plexes. *Optimization Letters*, 4(3):311–320, 2010.

8 Rémy Belmonte, Petr A. Golovach, Pim Hof, and Daniël Paulusma. Parameterized complexity of three edge contraction problems with degree constraints. *Acta Informatica*, 51(7):473–497, 2014.

9 Nadja Betzler, Robert Bredereck, Rolf Niedermeier, and Johannes Uhlmann. On bounded-degree vertex deletion parameterized by treewidth. *Discrete Applied Mathematics*, 160(1-2):53–60, 2012.

10 Hans L Bodlaender and Babette van Antwerpen-de Fluiter. Reduction algorithms for graphs of small treewidth. *Information and Computation*, 167(2):86–119, 2001.

11 Andries Evert Brouwer and Henk Jan Veldman. Contractibility and NP-completeness. *Journal of Graph Theory*, 11(1):71–79, 1987.

12 Leizhen Cai and Chengwei Guo. Contracting few edges to remove forbidden induced subgraphs. In *International Symposium on Parameterized and Exact Computation*, pages 97–109. Springer, 2013.

13 Zhi-Zhong Chen, Michael Fellows, Bin Fu, Haitao Jiang, Yang Liu, Lusheng Wang, and Binhai Zhu. A linear kernel for co-path/cycle packing. In *International Conference on Algorithmic Applications in Management*, pages 90–102. Springer, 2010.

14 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

15 Anders Dessmark, Klaus Jansen, and Andrzej Lingas. The maximum k-dependent and f-dependent set problem. In *International Symposium on Algorithms and Computation*, pages 88–97. Springer, 1993.

16 Michael R Fellows, Jiong Guo, Hannes Moser, and Rolf Niedermeier. A generalization of nemhauser and trotter's local optimization theorem. *Journal of Computer and System Sciences*, 77(6):1141–1158, 2011.

17 Fedor V. Fomin, Daniel Lokshtanov, Ivan Mihajlin, Saket Saurabh, and Meirav Zehavi. Computation of Hadwiger Number and Related Contraction Problems: Tight Lower Bounds. In *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, pages 49:1–49:18. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020.

18 Robert Ganian, Fabian Klute, and Sebastian Ordyniak. On structural parameterizations of the bounded-degree vertex deletion problem. In *35th Symposium on Theoretical Aspects of Computer Science (STACS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

19 Petr A Golovach, Marcin Kaminski, Daniël Paulusma, and Dimitrios M Thilikos. Increasing the minimum degree of a graph by contractions. *Theoretical computer science.*, 481:74–84, 2013.

20 Petr A Golovach, Pim van't Hof, and Daniël Paulusma. Obtaining planarity by contracting few edges. *Theoretical Computer Science*, 476:38–46, 2013.

21 Sylvain Guillemot and Dániel Marx. A faster fpt algorithm for bipartite contraction. *Information Processing Letters*, 113(22-24):906–912, 2013.

22 Spoorthy Gunda, Pallavi Jain, Daniel Lokshtanov, Saket Saurabh, and Prafullkumar Tale. On the parameterized approximability of contraction to classes of chordal graphs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020)*, 2020 (To Appear).

23 Pinar Heggernes, Pim Van'T Hof, Daniel Lokshtanov, and Christophe Paul. Obtaining a bipartite graph by contracting few edges. *SIAM Journal on Discrete Mathematics*, 27(4):2143–2156, 2013.

24 Pinar Heggernes, Pim Van't Hof, Benjamin Lévêque, Daniel Lokshtanov, and Christophe Paul. Contracting graphs to paths and trees. *Algorithmica*, 68(1):109–132, 2014.

25 Christian Komusiewicz, Falk Hüffner, Hannes Moser, and Rolf Niedermeier. Isolation concepts for efficiently enumerating dense subgraphs. *Theoretical Computer Science*, 410(38-40):3640–3654, 2009.

26 R Krithika, Pranabendu Misra, and Prafullkumar Tale. An fpt algorithm for contraction to cactus. In *International Computing and Combinatorics Conference*, pages 341–352. Springer, 2018.

27 Ramaswamy Krithika, Pranabendu Misra, Ashutosh Rai, and Prafullkumar Tale. Lossy kernels for graph contraction problems. In *36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

28 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Slightly superexponential parameterized problems. *SIAM Journal on Computing*, 47(3):675–702, 2018.

29 Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. On the hardness of eliminating small induced subgraphs by contracting edges. In *International Symposium on Parameterized and Exact Computation*, pages 243–254. Springer, 2013.

30 Barnaby Martin and Daniël Paulusma. The computational complexity of disconnected cut and 2k2-partition. *Journal of combinatorial theory, series B*, 111:17–37, 2015.

31 Naomi Nishimura, Prabhakar Ragde, and Dimitrios M Thilikos. Fast fixed-parameter tractable algorithms for nontrivial generalizations of vertex cover. *Discrete Applied Mathematics*, 152(1-3):229–245, 2005.

32 Saket Saurabh, Uéverton dos Santos Souza, and Prafullkumar Tale. On the parameterized complexity of grid contraction. In *17th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

33 Toshimasa Watanabe, Tadashi Ae, and Akira Nakamura. On the removal of forbidden graphs by edge-deletion or by edge-contraction. *Discrete Applied Mathematics*, 3(2):151–153, 1981.

34 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.