# Algorithms and Intractability of Some NP-hard Domination Problems with Private Structure

Louis Dublois

PSL
UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à Université Paris-Dauphine

# Algorithmes et Intractabilité de Certains Problèmes de Domination NP-difficiles avec Structure Privée


Soutenue par
**Louis DUBLOIS**
Le 01 juillet 2021


École doctorale n°543
**Ecole Doctorale SDOSE**

Spécialité
**Informatique**

Composition du jury :

| | |
|---|---|
| Christophe PICOULEAU<br>Professeur, CNAM | *Président* |
| Bruno ESCOFFIER<br>Professeur, Sorbonne Université | *Rapporteur* |
| Cédric BENTZ<br>Maître de conférences, CNAM | *Rapporteur* |
| Valia MITSOU<br>Maîtresse de conférences, Université de Paris | *Examinatrice* |
| Michael LAMPIS<br>Maître de conférences, Université Paris-Dauphine | *Co-encadrant de thèse* |
| Vangelis Th. PASCHOS<br>Professeur, Université Paris-Dauphine | *Directeur de thèse* |

Dauphine | PSL
UNIVERSITÉ PARIS

Université Paris-Dauphine, PSL Research University, CNRS, LAMSADE

# Algorithms and Intractability of Some NP-hard Domination Problems with Private Structure

Louis Dublois

Manuscrit pour l'obtention d'un doctorat ès sciences

Spécialité Informatique

Avril 2021

---

| | | |
|---|---|---|
| Bruno Escoffier | ..... | Rapporteur |
| Cédric Bentz | ..... | Rapporteur |
| Valia Mitsou | ..... | Examinateur |
| Christophe Picouleau | ..... | Examinateur |
| Michael Lampis | ..... | Co-directeur de thèse |
| Vangelis Th. Paschos | ..... | Directeur de thèse |

## Résumé

Pour résoudre des problèmes **NP**-difficiles, plusieurs paradigmes ont été développés durant les dernières décennies : l'*approximation polynomiale*, qui a pour objet de produire des solutions proches de l'optimum en temps polynomial ; la *résolution exacte*, qui a pour objectif de renvoyer une solution optimal en temps exponentiel, où la partie exponentielle dépend de la taille de l'instance dans le cas d'un algorithme dit exact, ou du paramètre choisi dans le cas d'un algorithme paramétré ; ou encore l'*approximation super-polynomiale*, qui permet d'obtenir un compromis continu entre la qualité de la solution et le temps d'exécution (allant de l'approximation polynomiale à la résolution exacte). Seulement, à partir de certaines hypothèses de complexité, il a été prouvé que, sous ces hypothèses, il est impossible d'obtenir certains algorithmes, et donc certaines performances, selon le problème **NP**-difficile étudié.

Dans ce manuscrit, nous présentons certaines méthodes permettant d'obtenir des algorithmes dans ces différents paradigmes, ainsi que des méthodes pour obtenir des résultats d'impossibilité. Nous illustrons ces méthodes en les mettant en œuvre et en les adaptant à trois problèmes **NP**-difficiles que nous avons étudiés : Min Mixed Dominating Set, où l'on cherche un ensemble minimum d'arêtes et de sommets qui dominent toutes les arêtes et sommets du graphe ; Max Min Feedback Vertex Set, où l'on cherche un feedback vertex set minimal de taille maximum ; et Upper Dominating Set, où l'on cherche un dominating set minimal de taille maximum. Ces trois problèmes **NP**-difficiles que nous avons étudiés sont des problèmes de domination, et de plus ils offrent tous une *structure privée*. En effet, pour Min Mixed Dominating Set, nous prouvons qu'il existe toujours une solution optimale pour laquelle tout sommet pris dans la solution a deux voisins privés dominés seulement par lui. Pour Max Min Feedback Vertex Set, nous cherchons un feedback vertex set minimal, c'est-à-dire une solution où chaque sommet pris a au moins un cycle privé dominé seulement par lui. Et pour Upper Dominating Set, nous cherchons un ensemble dominant minimal, c'est-à-dire une solution où chaque sommet pris a au moins un sommet privé dominé seulement par lui, que ce soit un voisin privé dominé seulement par lui ou le sommet lui-même dominé seulement par lui. Nous étudions ces trois problèmes de domination sous ce spectre de structure privée.

Plus précisément, pour ces problèmes étudiés, nous obtenons les résultats suivants. Pour Min Mixed Dominating Set : nous obtenons une $(2-\varepsilon)$-inapproximabilité en temps polynomial sous l'Unique Games Conjecture ; nous présentons un algorithme exact de complexité $O^*(1.912^n)$ ; nous

décrivons un algorithme paramétré par la taille de la solution $k$ de complexité $O^*(3.510^k)$ ; nous obtenons un algorithme paramétré par la treewidth $tw$ de complexité $O^*(5^{tw})$ et nous prouvons que ce résultat est le meilleur que l'on puisse espérer obtenir sous la Strong Exponential Time Hypothesis. Pour Max Min Feedback Vertex Set : nous présentons un algorithme polynomial de rapport d'approximation $O(n^{2/3})$ et nous prouvons que sous l'hypothèse $\mathbf{P} \neq \mathbf{NP}$ c'est le meilleur algorithme que l'on puisse expérer obtenir ; nous décrivons un algorithme super-polynomial qui renvoie une $r$-approximation en temps $n^{O(n/r^{3/2})}$ et nous montrons que nous ne pouvons pas obtenir de meilleur algorithme sous l'Exponential Time Hypothesis. Pour Upper Dominating Set : nous montrons que le problème ne peut pas être résolu en temps $O(n^{o(k)})$ et qu'il n'existe pas de $r$-approximation en temps $O(n^{k^{1-\varepsilon}})$ sous l'Exponential Time Hypothesis ; nous présentons un algorithme paramétré par la pathwidth $pw$ de complexité $O^*(6^{pw})$ et nous prouvons que ce résultat est le meilleur que l'on puisse espérer obtenir sous la Strong Exponential Time Hypothesis ; nous décrivons un algorithme super-polynomial qui renvoie une $r$-approximation en temps $n^{O(n/r)}$ et nous montrons que nous ne pouvons pas obtenir de meilleur algorithme sous l'Exponential Time Hypothesis.

# Contents

# Chapter 1

# Introduction

In this manuscript, we study three **NP**-hard problems: Min Mixed Dominating Set[1], Max Min Feedback Vertex Set and Upper Dominating Set. These three problems belong to the class of **NP**-hard domination problems. Indeed, in the Min Mixed Dominating Set problem we seek to dominate all edges and all vertices of the given graph by taking edges and vertices. In the Max Min Feedback Vertex Set problem we seek to dominate all cycles of the graph by taking vertices. And in the Upper Dominating Set problem we seek to dominate all vertices of the graph by taking vertices.

Furthermore, we study these three problems under the scope of *private structure*. What we mean by private structure is the fact that for a set of elements to be a feasible solution every element of a specific structure $S$ of the graph must have another structure which is private for this element. For the Min Mixed Dominating Set problem, we prove (Lemma 2.3) that there always exists an optimal solution for which every vertex taken in the solution has two private neighbors, that is two neighbors dominated only by this vertex. For the Max Min Feedback Vertex Set problem, which is the Max-Min version of the more studied Min Feedback Vertex Set problem, we seek a set of vertices which dominates all cycles of the graph and which is minimal, that is, such that every vertex taken in the solution has a private cycle, i.e. a cycle dominated only by this vertex. For the Upper Dominating Set problem, which is the Max-Min version of the well-known Min Dominating Set problem, we seek a set of vertices which dominates all vertices of the graph and which is minimal, that is, such that every vertex taken in the solution has a private vertex, i.e. every vertex taken either has a neighbor dominated only by itself or it is its own vertex dominated only

[1]The problems studied, used, or at least explicitly described in this manuscript, are all defined in the Appendix.

by itself.

We study these three **NP**-hard problems under the following frameworks: the polynomial-time approximation, the exact algorithms, and the super-polynomial approximation. For each of these three problems, we improve the state of the art by either giving faster algorithms or by designing better algorithms for which we give a matching lower bound under some complexity assumptions showing that our algorithms are the best algorithms we can hope under these assumptions.

The notion of private structure is crucial in our results, depending on the problem. For the Min Mixed Dominating Set problem, the fact that there always exists an optimal solution for which each vertex taken has two private neighbors allows us to improve the known exact algorithms by using extensively this more restricted definition, and we also give a tight algorithm parameterized by the treewidth. For the Max Min Feedback Vertex Set problem, since it is the Max-Min version of Min Feedback Vertex Set, the corresponding private structure is mandatory to obtain a feasible solution and we use it rightfully in order to completely settle the polynomial-time and the super-polynomial approximations of this problem. For the Upper Dominating Set problem, again since it is the Max-Min version of Min Dominating Set, the corresponding private structure is inherent to the definition of any feasible solution, and we use this private structure to settle the super-polynomial approximation of this problem, to improve the best algorithm parameterized by the pathwidth of this problem, and along the way we give improved hardness results.

We first begin in the next section to present how this manuscript is organized.

## 1.1 Organization

This manuscript is based on the following four papers:

- [DLP20] Louis Dublois, Michael Lampis, and Vangelis Th. Paschos, New Algorithms for Mixed Dominating Set, *15th International Symposium on Parameterized and Exact Computation, IPEC 2020*, volume 180, pages 9:1-9:17, 2020.
- [DLP21a] Louis Dublois, Michael Lampis and Vangelis Th. Paschos, New Algorithms for Mixed Dominating Set, *Discrete Mathematics & Theoretical Computer Science, DMTCS, 2021.*
- [DHG+20] Louis Dublois, Tesshu Hanaka, Mehdi Khosravian Ghadikolaei, Michael Lampis, and Nikolaos Melissinos, (In)approximability of

Maximum Minimal FVS, *31st International Symposium on Algorithms and Computation, ISAAC 2020*, volume 181, pages 3:1-3:14, 2020.

- [DLP21b], Louis Dublois, Michael Lampis, and Vangelis Th. Paschos, Upper Dominating Set: Tight Algorithms for Pathwidth and Sub-Exponential Approximation, *12th International Conference on Algorithms and Complexity, CIAC 2021*, 2021.

In Chapter 2, we focus on the Min Mixed Dominating Set problem. In this problem, we are given a graph[2] $G = (V, E)$, and we seek to find a subset of vertices $D \subseteq V$ and a subset of edges $M \subseteq E$ such that $|D \cup M|$ is minimized and the set $D \cup M$ dominates $V \cup E$, where a vertex dominates itself, its neighbors, and its incident edges, and an edge dominates itself, its two endpoints, and all edges with which it shares an endpoint. For this problem, we present the following results we have obtained:

- There is no polynomial-time approximation algorithm that can output a $(2-\varepsilon)$-approximate solution for any $\varepsilon > 0$, under the Unique Games Conjecture (UGC). Thus we show that the 2-approximation algorithm of Hatami [Hat07] is the best we can hope under this complexity assumption.

- An exact algorithm of complexity $O^*(1.912^n)$ using polynomial space, improving on the $O^*(2^n)$ algorithm of Madathil et al. [MPSS19].

- An FPT algorithm parameterized by the size of the solution $k$ of complexity $O^*(3.510^k)$, improving on the $O^*(7.465^k)$ and $O^*(4.172^k)$ algorithms of Jain et al. [JJPS17] and Xiao and Sheng [XS19], respectively.

- An FPT algorithm parameterized by the treewidth $tw$ of the given graph with complexity $O^*(5^{tw})$, improving on the $O^*(6^{tw})$ algorithm of Jain et al. [JJPS17], and a tight lower bound stating that there is no algorithm that solves this problem in time $O^*((5-\varepsilon)^{pw})$ for any $\varepsilon > 0$ under the Strong Exponential Time Hypothesis (SETH). Thus we prove that our previous algorithm for treewidth and the one of Jain et al. [JJPS17] for pathwidth of complexity $O^*(5^{pw})$ are optimal under the SETH.

In Chapter 3, we consider the Max Min Feedback Vertex Set problem. In this problem, we are given a graph $G = (V, E)$, and we seek to find a subset of vertices $S \subseteq V$ which forms a minimal feedback vertex set

[2] A definition of a graph is given in Section 1.6.

of maximum size, where a set $S \subseteq V$ is a feedback vertex set if the graph induced by $V \setminus S$ is a forest, and is minimal if no proper subset of it is a feedback vertex set of $G$. For this problem, we present the following results we have obtained:

- An improved **NP**-hardness from $\Delta \geq 9$ [MS00] to $\Delta \geq 6$.
- A non-trivial polynomial-time approximation algorithm of ratio $O(n^{2/3})$, and a matching lower bound stating that unless $\mathbf{P} = \mathbf{NP}$ no polynomial-time algorithm can output a $n^{2/3-\varepsilon}$-approximation for any $\varepsilon > 0$. Thus we completely settle the polynomial-time approximation of this problem.
- Along the way, we obtain an optimal $O(\Delta)$-approximation algorithm (optimal unless $\mathbf{P} = \mathbf{NP}$), some extremal results on the size of any minimal feedback vertex set in a graph, and a cubic kernel parameterized by the size of the solution $k$
- A super-polynomial algorithm that outputs an $r$-approximation in time $n^{O(n/r^{3/2})}$ for any ratio $r \leq n^{2/3}$, thus matching our polynomial-time approximation, and a matching lower bound stating that under the randomized Exponential Time Hypothesis (ETH) no algorithm can output an $r$-approximation in time $n^{(n/r^{3/2})^{1-\varepsilon}}$ for any $r$ and any $\varepsilon > 0$. Thus we also completely settle the super-polynomial approximation of this problem.

In Chapter 4, we focus on the Upper Dominating Set problem. In this problem, we are given a graph $G = (V, E)$, and we seek to find a subset of vertices $D \subseteq V$ which forms a minimal dominating set of maximum size, where a set $D \subseteq V$ is a dominating set if every vertex of $V$ is either in $D$ or adjacent to a vertex in $D$, and is minimal if no proper subset of it is a dominating set of $G$. For this problem, we present the following results we have obtained:

- An improved **W[1]**-hardness compared to the one of Bazgan et al. [BBC+18a], stating that under the ETH there is no algorithm that solves the problem in time $O(n^{o(k)})$.
- An FPT-approximation intractability: under the ETH, there is no algorithm that outputs an $r$-approximation in time $O(n^{k^{1-\varepsilon}})$ for any constant $r > 0$ and any $\varepsilon > 0$. Thus we answer negatively to the existence of such an FPT-approximation algorithm.

- An FPT algorithm parameterized by the pathwidth $pw$ of the given graph of complexity $O^*(6^{pw})$, improving on the $O^*(7^{pw})$ algorithm of Bazgan et al. [BBC+18a], and a tight lower bound stating that there is no algorithm that solves this problem in time $O^*((6-\varepsilon)^{pw})$ for any $\varepsilon > 0$ under the SETH. Thus we prove that our previous algorithm for pathwidth is optimal under the SETH.

- A super-polynomial algorithm that outputs an $r$-approximation in time $n^{O(n/r)}$ for any ratio $r > 1$, and a matching lower bound stating that under the randomized ETH there is no algorithm that can output an $r$-approximation in time $n^{(n/r)^{1-\varepsilon}}$ for any $r$ and any $\varepsilon > 0$. Thus we completely settle the super-polynomial approximation of this problem.

Before presenting these results we have obtained for the three problems we have studied - Min Mixed Dominating Set, Max Min Feedback Vertex Set and Upper Dominating Set, we will make in the rest of this chapter a brief introduction on the concepts we have used to obtain our results. More precisely, in the next section, we present what we call a *private structure*. In Section 1.3, we describe the framework of *polynomial-time approximation*, where we present a polynomial-time approximation algorithm for the Max-Min problem Max Min Vertex Cover, the notion of *gap-introducing* reduction, an inapproximability result under the conjecture $\mathbf{P} \neq \mathbf{NP}$ obtained via a *gap-preserving* reduction, and the Unique Games Conjecture. In Section 1.4, we focus on *exact algorithms*, whether they are FPT algorithms or not, and to do so we give a general exact algorithms solving two well-known problems, we present the two conjectures Exponential Time Hypothesis and Strong Exponential Time Hypothesis, we give a branching FPT algorithm for Max Min Vertex Cover, we present the W hierarchy, and finally we describe the notions of *treewidth* and *pathwidth*. In Section 1.5, we describe the framework of *super-polynomial approximation*, where we first present a super-polynomial algorithm for Max Min Vertex Cover, a matching lower bound on the super-polynomial inapproximability of this problem, and the notion of *parameterized approximation*. Finally, we give in Section 1.6 some notation.

## 1.2 Private Structure

The problems we are interested in in this manuscript are *optimization* **NP**-hard problems. The problem Min Vertex Cover is an example of an optimization problem, and is the optimization version of the *decision* problem Vertex Cover. In the Min Vertex Cover problem, we are given a graph

$G = (V, E)$, and we seek to find a vertex cover of minimum size, where a vertex cover is a set of vertices $C \subseteq V$ such that for every edge $(u, v) \in E$ at least one of $u$ and $v$ is in $C$. The Min Vertex Cover is a *minimization* problem since we are looking for a vertex cover of minimum size. Among the optimization problems, there are also the subclass of *maximization* problems. An example of such a maximization problem is the Max Independent Set problem. In this problem, we are given a graph $G = (V, E)$, and we seek to find an independent set of maximum size, where an independent set is a set of vertices $I \subseteq V$ such that for every pair of vertices $u, v$ in $I$ there is no edge $(u, v) \in E$.

These two problems, Min Vertex Cover and Max Independent Set, are studied with a minimization and a maximization objective, respectively. However, these two problems are also studied with, in a sense, an inverse objective. For Vertex Cover, we would be interested in finding a vertex cover of maximum size, while being minimal, where a vertex cover $C$ is minimal if no proper subset $C' \subset C$ is a vertex cover. This problem is called Max Min Vertex Cover. For Independent Set, we would be interested in finding an independent set of minimum size, while being maximal, where an independent set $I$ is maximal if no proper superset $I' \supset I$ is an independent set. This problem is called Min Max Independent Set (also called Min Independent Dominating Set).

From these two definitions of Max Min Vertex Cover and Min Independent Dominating Set, which are both **NP**-hard problems, we can introduce the notion of *private structure*. Indeed, for the former problem, the vertex cover $C$ is minimal if no proper subset of it is a vertex cover, that means if for every vertex $u \in C$, at least one of its neighbors $v$ is not in $C$, because otherwise removing $u$ from $C$ would give a vertex cover that is a proper subset of $C$. Thus, from this, we can say that the edge $(u, v)$ is a *private edge* of $u$, that is an edge covered only by $u$. For Min Independent Dominating Set, we cannot make such an observation. Indeed, for this problem, an independent set $I$ is maximal if no proper superset of it is an independent set, that means, if for every vertex $u \in V$, at least one vertex of $N[u]$ is in $I$, because otherwise adding $u$ to $I$ would give an independent set that is a proper superset of $C$. Nonetheless, note that this vertex $v \in N[u]$ that belongs to $I$ is not necessarily *private* to $u$, since it can also be shared to another vertex $w$ for which $v \in N[w]$. Thus, for Min Independent Dominating Set, we cannot infer any *private structure* associated to this problem.

We can observe first that any Max-Min optimization problem has such a mandatory private structure. Indeed, consider such a Max-Min problem. For such a problem, we seek to find a solution that is minimal, so we seek a

solution $S$ such that no proper subset of it is a solution. Thus, this minimality implies that every element in $S$ must have a private structure that depends on the problem, because otherwise $S$ is not minimal. Consider now a Min-Max problem. As we have mentioned for Min Independent Dominating Set, Min-Max problems do not have such a private structure. Indeed, for such a problem, we seek to find a solution that is maximal, so we seek a solution $S$ such that no proper superset of it is a solution. Thus, this maximality implies that every element of the instance must intersect with $S$, but this intersection is not necessarily private for the considered element of the instance, as we have showed for Min Independent Dominating Set.

Although Max-Min problems have this mandatory private structure in their definitions, the natural Min problems also have such a private structure, but which is implicit in their definitions. Take for example the Min Vertex Cover problem. An optimal solution for this problem is necessarily minimal, because otherwise it is not minimum, but it is possible to solve this problem regardless of the minimality, since the fact that a solution is minimum implies that is is minimal. Note that we cannot make the same observation for maximization problems, since, as we have described earlier, the maximality of such problems does not imply a private structure associated to any solution. Thus, even though a maximization problem can be solved using its maximality property, this property has no private structure associated to it.

Thus, the notion of private structure has to be taken into account, can be taken into account, or cannot be taken into account, depending on the nature of the considered problem. Indeed, when studying a Max-Min problem, the private structure is inherent in the definition of the problem, and thus algorithms solving this problem must depend on this private structure in order to obtain a feasible solution, as we will show throughout the Introduction and when studying the two problems Max Min Feedback Vertex Set and Upper Dominating Set. On the other hand, when studying a minimization problem, the private structure is implicit in the definition of the problem, and thus might not be used in an algorithm solving the problem, but can be used, and can sometimes help to design faster algorithms solving the considered problem, as we will show in Section 1.4 and later in Chapter 2 when studying Min Mixed Dominating Set. Finally, when studying a maximization problem or its Min-Max version, there is *a priori* no private structure associated to its maximality, and in this case the notion of private structure cannot be used to solve the considered problem.

The three problems we have studied have such a private structure, implicit or inherent in the definition of the problem. Naturally, both the Max Min Feedback Vertex Set and Upper Dominating Set problems

have such a mandatory private structure since they are Max-Min versions of the Min Feedback Vertex Set and Min Dominating Set problems, respectively. More precisely, in the Max Min Feedback Vertex Set problem, the private structure is the fact that a vertex $u$ in a feedback vertex set $S$ must have a *private cycle*, that is a cycle dominated only by $u$, because otherwise $u$ can be removed from $S$. In the Upper Dominating Set problem, the private structure is the fact that a vertex $u$ in a dominating set $D$ must have a *private vertex*, that is either that $u$ is dominated only by itself, or that $u$ has a neighbor dominated only by $u$, because otherwise $u$ can be removed from $D$.

For our last problem studied, Min Mixed Dominating Set, we prove in Lemma 2.3 that although this problem has an implicit private structure, this structure is strong. Indeed, we prove that there always exist an optimal mixed dominating set $D \cup M$ such that every vertex $u \in D$ has two private neighbors, that is two neighbors in $V \setminus (D \cup V(M))$ dominated only by $u$. Interestingly, this implicit private structure is quite strong, even compared to the two Max-Min problems we have studied, Max Min Feedback Vertex Set and Upper Dominating Set. Indeed, in these two problems, the cardinality associated to their corresponding private structure is one, one private cycle and one private vertex, respectively. For Min Mixed Dominating Set, this associated cardinality is two (every vertex in $D$ has two private neighbors), which allows us to obtain faster algorithms solving this problem by extensively using this implicit private structure.

Now that we have presented the notion of private structure and showed how it is related to the three problems we have studied, we present in the next sections the frameworks of polynomial-time approximation, exact algorithms and super-polynomial approximation, through the scope of private structure, beginning with the polynomial-time approximation.

## 1.3 Polynomial-Time Approximation

Since only polynomial-time algorithms seem to be easy to implement practically, since a large variety of problems are **NP**-hard, and since **NP**-hard problems cannot be solved exactly in polynomial time under the complexity assumption $\mathbf{P} \neq \mathbf{NP}$, it seems natural to develop a framework where it is possible to design polynomial-time algorithms for such problems, with the downside that such algorithms do not output an optimal solution. Such algorithms are called *polynomial-time approximation* algorithms. In this section, our objective is to remind the reader of the basic notions of this framework, and to present some background and cornerstone notions that are relevant to

the results we have obtained for the two problems Min Mixed Dominating Set and Max Min Feedback Vertex Set.

More precisely, we present: (i) the $\sqrt{n}$-approximation algorithm for Max Min Vertex Cover due to Boria et al. [BCP13] which is a good example of a polynomial-time approximation algorithm for a Max-Min problem, and which we simplify to mirror with our algorithm for Max Min Feedback Vertex Set of Section 3.3 (ii) the notion of *gap-introducing* reduction and the PCP Theorem (PCP stands for Probabilistically Checkable Proofs) from which the inapproximability of Max Independent Set is obtained (iii) the $n^{1/2-\varepsilon}$-inapproximability result for Max Min Vertex Cover also due to Boria et al. [BCP13], designed via a *gap-preserving* reduction from the Max Independent Set and the inapproximability of this latter problem, and which is related to our inapproximability for Max Min Feedback Vertex Set of Section 3.4 (iv) the Unique Games Conjecture (UGC) from which a $(2-\varepsilon)$-inapproximability was obtained by Dudycz et al. [DLM19] for the Min Edge Dominating Set problem and from which we obtain the same inapproximability for our considered problem Min Mixed Dominating Set, which we present in Section 2.1.

Let us begin by presenting the $\sqrt{n}$-approximation algorithm of Boria et al. [BCP13] for the Max Min Vertex Cover problem. Recall that for this problem the private structure is that every vertex in the solution has to have at least one private edge covered, or dominated, only by this vertex. This algorithm, which we simplify, works as follows: (1) preprocess the given graph by deleting the isolated vertices (2) if there exists a vertex $u$ with degree strictly more than $\sqrt{n}$, take the vertex cover $C = V \setminus \{u\}$, delete vertices from $C$ until $C$ is minimal, and output the solution obtained (3) else, begin with the vertex cover $C = V$, delete vertices from it until $C$ is minimal, and output the solution obtained.

Clearly, this algorithm works in polynomial time, and outputs a minimal vertex cover since in the two cases (2) and (3) we output $C$ when it is a minimal vertex cover. So let us now prove that this algorithm is a $\sqrt{n}$-approximation algorithm. Suppose first that the given graph $G = (V, E)$ after being preprocessed is connected, because otherwise the approximation ratio can be applied to every connected component. Recall that a maximum minimal vertex cover in the given graph $G$ has at most $n$ vertices, so it is sufficient to prove that the vertex cover output by our algorithm has size at least $\sqrt{n}$. In the first case (2), necessarily $C$ contains at the end all neighbors of $u$ because otherwise $C$ is not a vertex cover, so $|C| > \sqrt{n}$, and it is indeed a $\sqrt{n}$-approximation. In the second case (3), since the case (2) does not occur, we have $\Delta \leq \sqrt{n}$, and we also have that the number of edges in $G$ is at least $n-1$ since $G$ is connected. So every (minimal) vertex cover has size

at least $c\sqrt{n}$ for a constant $c$ smaller than 1. Indeed, suppose that a minimal vertex cover has at most $c\sqrt{n}$ vertices. Since the maximum degree in $G$ is $\sqrt{n}$, the number of edges dominated by this minimal vertex cover is at most $c\sqrt{n} \times \sqrt{n} = cn$, which is smaller than $n - 1$ for $n$ sufficiently large. So any minimal vertex cover has size at least $c\sqrt{n}$, and we output such a minimal vertex cover, so our algorithm asymptotically outputs an $\sqrt{n}$-approximation.

We observe that this algorithm also states that any graph without isolated vertices has a minimal vertex cover of size at least $\sqrt{n}$. Interestingly, our polynomial-time approximation for Max Min Feedback Vertex Set we describe in Section 3.3 is similar to this approximation algorithm for Max Min Vertex Cover: our algorithm for Max Min Feedback Vertex Set begins by preprocessing the given graph by deleting vertices of degree 1 and contracting edges whose endpoints have degree 2 (whereas the preprocessing step of the algorithm for Max Min Vertex Cover simply removes isolated vertices), and then constructs a minimal feedback vertex set of size at least $cn^{1/3}$ for a constant $c$. It gives us our $O(n^{2/3})$-approximation algorithm for Max Min feedback Vertex Set, and similarly to Max Min Vertex Cover it is a constructive algorithm which implies that any properly preprocessed graph has a minimal feedback vertex set of size $cn^{1/3}$ (whereas it is $cn^{1/2}$ for Max Min Vertex Cover).

Coming back to Max Min Vertex Cover, Boria et al. [BCP13] in the same paper designed a matching lower bound stating that, unless $\mathbf{P} = \mathbf{NP}$, no polynomial-time algorithm can output a $n^{1/2-\varepsilon}$-approximation for Max Min Vertex Cover for any $\varepsilon > 0$. To obtain this inapproximability result, they produced a *gap-preserving* reduction from the Max Independent Set problem, for which there exists a $n^{1-\varepsilon}$-inapproximability under $\mathbf{P} \neq \mathbf{NP}$ [Zuc05]. Before presenting what a gap-preserving reduction is, and giving this reduction from Boria et al. as an example, let us first present the $n^{1-\varepsilon}$-inapproximability for Max Independent Set.

This inapproximability result for Max Independent Set is due to a new characterization of the class **NP** through the PCP Theorem, by Arora and Safra [AS98]. The PCP Theorem is defined through the notion of *probabilistically checkable proof system*, as follows:

**Definition 1.1** (Probabilistically Checkable Proof System)**.** *Given a decision problem* $\Pi$*, a probabilistically checkable proof system for* $\Pi$ *is a quadruplet* $(c(n), s(n), r(n), q(n))$ *along with a* prover *and a* verifier*. The prover, given an instance* $I$ *of* $\Pi$ *of size* $n$ *for which the claimed solution is Yes, but which might be No, produces a proof* $P$ *which states that* $I$ *is a Yes-instance of* $\Pi$*. The verifier is a randomized oracle Turing machine* $T$ *which checks the proof* $P$ *by randomly checking some part of* $P$ *and decide to accept the*

*statement that $I$ is a Yes-instance or not. The system has the following properties:*

- *(Completeness): For any Yes-instance $I$ of $\Pi$, given the proof $P$ produced by the prover for the instance $I$, the verifier accepts the statement with probability at least $c(n)$.*
- *(Soundness): For any No-instance $I$ of $\Pi$, and given any proof $P$ for $I$, the verifier rejects the statement with probability at least $s(n)$.*

*The two functions $r(n)$ and $q(n)$ concern the complexity of the verifier: $q(n)$ is the maximum number of random queries of size at most $r(n)$ the verifier makes on proof $P$ for all instances $I$.*

*The complexity class $\mathbf{PCP}_{c(n),s(n)}[r(n), q(n)]$ is the class of all decision problems which have a probabilistically checkable proof system of completeness $c(n)$ and soundness $s(n)$, and where the verifier works in polynomial time making all its queries before receiving the answers and depends on $q(n)$ and $r(n)$.*

Arora and Safra [AS98] proved that $\mathbf{NP} = \mathbf{PCP}_{1,1/2}[O(\log(n)), O(1)]$.

From this new characterization of the class of problems **NP**, *gap-introducing* reductions were produced for two well-studied problems: Max-3-Sat and Max Independent Set. Informally, a gap-introducing reduction between a decision problem $\Pi_1$ and an optimization $\Pi_2$ is a reduction from $\Pi_1$ to $\Pi_2$ such that the existence of an algorithm for $\Pi_2$ with a specific ratio $r$ would decide problem $\Pi_1$, that is would decide for a given instance if the answer is Yes or No for $\Pi_1$. From these gap-introducing reductions, inapproximability results have been obtained for these two problems. For the former problem, there exists a constant $\varepsilon > 0$ such that there is no $(1 - \varepsilon)$-approximation algorithm for Max-3-Sat unless $\mathbf{P} = \mathbf{NP}$ [AB09]. For the latter problem, and any $\varepsilon > 0$, there is no $n^{1-\varepsilon}$-approximation algorithm for Max Independent Set unless $\mathbf{P} = \mathbf{NP}$ [Hås96]. This last inapproximability result for Max Independent Set was later derandomized by Zuckerman [Zuc05]. Interestingly, this inapproximability result for Max Independent Set states that the best polynomial-time approximation algorithm for this problem has ratio $n$, which was a long-awaited open question.

Moreover, this $n^{1-\varepsilon}$-inapproximability for Max Independent Set has been widely used as starting point to produce other inapproximability results for other **NP**-hard problems, through *gap-preserving* reductions. Informally, a gap-preserving reduction between two optimization problems $\Pi_1$ and $\Pi_2$ is a reduction from $\Pi_1$ to $\Pi_2$ such that $\Pi_1$ admits an $r$-approximation algorithm for a specific $r$ if and only if $\Pi_2$ admits an $r$-approximation algorithm.

We now present such a gap-preserving reduction, the one from Max Independent Set to Max Min Vertex Cover designed by Boria et al. [BCP13] we have mentioned earlier, and use the inapproximability result of the former problem to obtain a $n^{1/2-\varepsilon}$-inapproximability for the latter problem. Given an instance $G = (V, E)$ of Max Independent Set, we construct an instance $G' = (V', E')$ of Max Min Vertex Cover in the following way: we keep the original graph $G$; and for each vertex $u \in V$, we add $n$ vertices only connected to $u$.

Clearly, this reduction can be made in polynomial time, and the order of the graph $G'$ is $n^2$. So let us now prove that the gap is transferred from $n^{1-\varepsilon}$ to $n^{1/2-\varepsilon}$. For the first direction, let $C$ be any minimal vertex cover of $G'$. We construct an independent set $I$ in $G$ by taking all vertices of $V \setminus C$. Now, observe the following: for a vertex $u \in I$, the set $C$ necessarily takes the $n$ newly vertices of $V'$ connected to $u$. Thus, $|I| \geq \frac{|C \setminus V|}{n} \geq \frac{|C|-n}{n}$. For the other direction, let $I$ be any independent set of $G$. We construct a minimal vertex cover $C$ in $G'$ by taking the $n$ newly vertices connected to every vertex $u \in I$ and all the vertices of $V \setminus I$. Thus, $|C| \geq n|I| + (n - |I|)$. And for a maximum independent set $I^*$ of $G$, we obtain $|C^*| \geq |C| \geq n|I^*| + (n - |I^*|)$, for a maximum minimal vertex cover $C^*$ of $G'$. Now, suppose there exists an $r$-approximation algorithm for Max Min Vertex Cover; we have $|C| \geq r|C^*|$. Putting all together, we obtain:

$$|I| \geq \frac{|C| - n}{n} \geq \frac{r|C^*| - n}{n} \geq \frac{r}{n}(n|I^*| + (n - |I^*|)) - 1 \geq r|I^*|$$

Where the last inequality holds for $n$ sufficiently large. Thus, since the gap $r$ is preserved from Max Independent Set to Max Min Vertex Cover, and since this last problem is $n^{1-\varepsilon}$-inapproximable unless $\mathbf{P} = \mathbf{NP}$, we have $r \geq n^{1-\varepsilon}$. Now, since the order of the graph $G'$ is $n^2$, we obtain a $n^{1/2-\varepsilon}$-inapproximability under $\mathbf{P} \neq \mathbf{NP}$ for Max Min Vertex Cover by choosing $\varepsilon$ carefully.

This inapproximability result for Max Min Vertex Cover matches the $\sqrt{n}$-approximation algorithm we have described earlier in this section. We also give in Section 3.4 a matching lower bound under $\mathbf{P} \neq \mathbf{NP}$ to our $O(n^{2/3})$-approximation algorithm for Max Min Feedback Vertex Set, again with the inapproximability result of Max Independent Set as starting point.

These inapproximability results for Max Independent Set and Max Min Vertex Cover show that the conjecture $\mathbf{P} \neq \mathbf{NP}$ can provide tight inapproximability results for some **NP**-hard problems, but is sometimes not strong enough for other **NP**-hard problems. For example, for the more nat-

ural Min Vertex Cover problem, Dinur and Safra [DS02] proved that under this conjecture we cannot 1.36-approximate the problem, which was recently improved by Khot et al. [KMS18] to a $(\sqrt{2}-\varepsilon)$-inapproximability under the same complexity assumption, whereas the best polynomial-time approximation for this problem has ratio 2.

Nonetheless, for Min Vertex Cover, a $(2-\varepsilon)$-inapproximability for any $\varepsilon > 0$ was obtained by Khot and Regev [KR08] under the Unique Games Conjecture (UGC). This complexity assumption was introduced by Khot in 2002 [Kho02] and is commonly used to derive inapproximability results. This conjecture can be defined in an handful number of ways, but we will describe it through the Unique Label Cover viewpoint, and more precisely through the definition made by Khot and Regev [KR08].

In the Unique Label Cover problem, we are given an instance $\Phi = (V, E, C, \Psi)$, where $(V, E)$ defines a graph, $C \subseteq \mathbb{N}$ is a set of colors, and $\Psi$ is a set of constraints, i.e., for every edge $(u, v) \in E$, $\Psi_{u,v} : C_u \to C_v$ for $C_u, C_v \subseteq C$ is a bijective function that defines the pairs of colors that are accepted for $u$ and $v$. The constraints of $\Psi$ mean the following: for an edge $(u, v)$ and a color $c \in C$, if the vertex $u$ is affected to color $c$, then $v$ must be affected the color $c' = \Psi_{u,v}(c)$, and conversely if the vertex $v$ is affected the color $c'$, then $u$ must be affected the color $c = \Psi^{-1}_{u,v}(c')$. A label $L$ is an assignment of colors given to the vertices of $V$, and we say that an edge $(u, v)$ is satisfied by the label if $c$ the color given to $u$ belongs to $C_u$ and $c'$ the color given to $v$ satisfies $\Psi_{u,v}(c) = c'$. The Unique Games Conjecture is the following:

**Definition 1.2** (Unique Games Conjecture)**.** *For any $\varepsilon, \gamma > 0$ and $t \in \mathbb{N}$, there exists an integer $k$ such that it is **NP**-hard to decide, for an instance $\Phi = (V, E, C, \Psi)$ of Unique Label Cover with $|C| = k$, between the following:*

- *(Yes-instance): There exists a 1-label $L$ and a subset $W \subseteq V$ with $|W| \geq (1-\varepsilon)|V|$ such that $L$ satisfies all edges in the induced subgraph $G[W]$.*
- *(No-instance): For any $t$-label $L$ and any subset $W \subseteq V$ with $|W| \geq \gamma|V|$, there exists at least one edge in $G[W]$ that is not satisfied by $L$.*

As we mentioned before, Khot and Regev [KR08] obtained a $(2-\varepsilon)$-inapproximability result for any $\varepsilon > 0$ for Min Vertex Cover, via a gap-preserving reduction from the Unique Label Cover problem to the Min Vertex Cover problem. More recently, Dudycz et al. [DLM19] obtained,

by a simple modification of the reduction of Khot and Regev, a $(2-\varepsilon)$-inapproximability result for any $\varepsilon > 0$ for the Min Max Matching problem (also called Min Edge Dominating Set). We will in Section 2.1 present a gap-preserving reduction from Min Edge Dominating Set to our considered problem Min Mixed Dominating Set which allows us to obtain the same inapproximability result for the latter problem.

We now finish this section, where we have presented the basic notions of the polynomial-time approximation framework: a $\sqrt{n}$-approximation algorithm for Max Min Vertex Cover; the notion of gap-introducing reductions through the PCP Theorem and the UGC; and a matching $n^{1/2-\varepsilon}$-inapproximability result for Max Min Vertex Cover using a gap-preserving reduction. In the next section, we focus on exact algorithms.

## 1.4 Exact Algorithms

Although solving an **NP**-hard problem exactly in polynomial time is not possible under the assumption $\mathbf{P} \neq \mathbf{NP}$, it is still possible to solve them exponentially. This way of solving **NP**-hard problems in exponential time is called *exact algorithms*. The first approach to solve exactly an **NP**-hard problem in exponential time was to design such algorithms with the inherent exponential blow-up in the size of the instance. Nonetheless, the instances encountered are in practice specific, and some structural parameters of the instance might be smaller and known, for example the maximum degree of the graph, the maximum number of appearance of a variable in a CNF formula, or directly the size of an optimum solution. If we manage to move the inherent exponential blow-up to such a parameter, then our algorithm becomes polynomial if we consider the parameter as a constant. Such an algorithm is called a *parameterized algorithm*. In this section, we focus on exact algorithms with the exponential blow-up in the size of the instance, and on parameterized algorithms, where we remind the basic notions of this framework of exact algorithms which are relevant to the results we have obtained for the two problems Min Mixed Dominating Set and Upper Dominating Set.

More precisely, we present: (i) a general exact algorithm solving the two problems Min Vertex Cover and Max Min Vertex Cover, which is a good example of an exact algorithm with the exponential blow-up in the size of the instance, and which is a subroutine algorithm for our algorithm for Min Mixed Dominating Set of Section 2.2 (ii) the two conjectures Exponential Time Hypothesis (ETH) and Strong Exponential Time Hypothesis (SETH) for which we give some background and show that they fit the state

of the art (iii) a parameterized algorithm for Max Min Vertex Cover parameterized by the size of the solution $k$ inspired by the algorithm of Boria et al. [BCP13], which is a good example of a parameterized algorithm using the private structure of the considered problem, similar to our algorithm for Min Mixed Dominating Set of Section 2.3 (iv) the different complexity classes of the parameterized complexity framework (v) the two well-studied parameters *treewidth* and *pathwidth* for which we present how to design dynamic programming algorithms for these parameters as well as how to obtain tight lower bounds of such algorithms, notions we use to obtain such results for Min Mixed Dominating Set in Section 2.4 and for Upper Dominating Set in Sections 4.2 and 4.3.

Let us begin by presenting an exact algorithm where the exponential blow-up lies in the size of the instance, almost the same for the two problems Min Vertex Cover and Max Min Vertex Cover. By a result of Moser and Moon [MM65], it is possible to compute all maximal independent sets of a given graph in time $O^*(3^{n/3})$[3], and this is an upper-bound on the number of such maximal independent sets. Thus, using this algorithm and the well-known observation that a maximal independent set is the complement of a minimal vertex cover, we can solve both Min Vertex Cover and Max Min Vertex Cover in time $O^*(3^{n/3}) = O^*(1.4423^n)$.

Indeed, consider the Min Vertex Cover problem: begin by enumerating all maximal independent sets of the given graph; and then output the complement of the maximal independent set of maximum size, which is a minimum vertex cover. For this problem, as we have said in Section 1.2, its private structure is implicit, and it can be used to design faster algorithms. And indeed, this algorithm, which uses that a minimum vertex cover is also minimal, runs in time $O^*(1.4423^n)$, faster than the brute-force algorithm of complexity $O^*(2^n)$ which does not use the implicit private structure of Min Vertex Cover. Observe nonetheless that the current best algorithm for this problem by Xiao and Nagamochi [XN17] works in time $O^*(1.1996^n)$. For Max Min Vertex Cover: again begin by enumerating all maximal independent sets of the given graph; and then in contrary to the previous algorithm output the complement of the maximal independent set of minimum size. For this problem, with a mandatory private structure, this algorithm directly gives a $O^*(1.4423^n)$ exact algorithm. Note that the current best algorithm by Bourgeois et al. [BEP09] for Max Min Vertex Cover works in time $O^*(1.3351^n)$.

There is no intractability result of the type "problem Π cannot be solved in time $O^*((x-\varepsilon)^n)$ for a constant $x$ and any $\varepsilon$" in this framework of exact

[3] $O^*$ notation suppresses polynomial factors in the input size.

algorithms with the exponential blow-up in the size of the instance, but a major conjecture, the Exponential Time Hypothesis (ETH), has been made by Impagliazzo and Paturi in 1999 [IP01] on the exact resolution of the $k$-SAT problem. To define this conjecture, consider the problem $k$-SAT for a given integer $k$, which is a restriction of the SAT problem where each clause contains at most $k$ literals. For a given $k$, let $x_k$ be the smaller real number $x$ such that the problem $k$-SAT can be solved in time $O^*(2^{xn})$. Then, we have $x_k \leq x_{k+1}$ since $k$-SAT is less hard than $(k+1)$-SAT. The Exponential Time Hypothesis is the following:

**Definition 1.3** (Exponential Time Hypothesis)**.** *For every $k > 2$, $x_k > 0$.*

Informally, the ETH conjectures that the $k$-SAT problem cannot be solved in sub-exponential time, that is in $O^*(2^{o(n)})$ time. This conjecture is widely used to derive intractability results in the parameterized complexity and super-polynomial approximation frameworks. We will present later such hardness results under the ETH. Note that the ETH fits the state of the art concerning the best exact algorithms for solving $k$-SAT. Indeed, the best exact algorithm for 3-SAT is a randomized algorithm due to Hansen et al. [HKZZ19] and works in time $O^*(1.307^n)$. For $k$-SAT, the best exact algorithm is a randomized algorithm also due to Hansen et al. [HKZZ19] and works in time $O^*(2^{cn})$ for a constant $c < 1$ that depends on the probability that a variable is guessed in this randomized algorithm

We have mentioned that the ETH conjectures that the $k$-SAT problem cannot be solved in time $O^*(2^{o(n)})$. In fact, the ETH states an even stronger result: if the ETH holds, the problem $k$-SAT cannot be solved in time $O^*(2^{o(n+m)})$, where $m$ is the number of clauses in the corresponding $k$-CNF formula. Impagliazzo and Paturi [IP01] proved this harder result using the *Sparsification Lemma*. This Lemma states that, given a $k$-CNF formula $\phi$ with $n$ variables and $m$ clauses, and any $\varepsilon > 0$, it is possible to construct in $O^*(2^{\varepsilon n})$ time $O(2^{\varepsilon n})$ 3-CNF formulas $\phi_i$, each containing $O(n)$ clauses, such that $\phi$ is satisfiable if and only if there exists at least an $i$ such that $\phi_i$ is satisfiable. This Sparsification Lemma implies the $O^*(2^{o(n+m)})$ hardness result. Indeed, suppose there exists an algorithm which, given a $k$-CNF formula $\phi$, solves it in time $O^*(2^{o(n+m)})$. From this instance $\phi$, apply the Sparsification Lemma in order to obtain $O(2^{\varepsilon n})$ 3-CNF instances $\phi_i$, each with $O(n)$ clauses. Use the previous algorithm on every instance $\phi_i$ obtained: by the Sparsification Lemma, if one of them is satisfiable, then the original instance $\phi$ is satisfiable. Thus, this algorithm solves the original instance of $k$-SAT. Now, let us determine its running-time. The Sparsification Lemmas works in time $O^*(2^{\varepsilon n})$ and the algorithm applied to each $\phi_i$ works in time $O^*(2^{o(n+O(n))}) = O^*(2^{o(n)})$ since the instances $\phi_i$ have $O(n)$ clauses. So the

total running time of this procedure is $O^*(2^{o(n)})$, contradicting the ETH. So indeed, if the ETH holds, the problem $k$-Sat cannot be solved in time $O^*(2^{o(n+m)})$.

Furthermore, an even stronger conjecture based on the ETH, the Strong Exponential Time Hypothesis (SETH), has also been made by Impagliazzo and Paturi [IP01]. To define this conjecture, observe that, since $x_k \leq x_{k+1}$, the real numbers $x_3, x_4, \ldots$ form a monotonic sequence, which is upper-bounded by one (since Sat can be solved in time $O^*(2^n)$ by a brute-force algorithm), so the sequence must converge to a limit $x_\infty$. The Strong Exponential Time Hypothesis is the following:

**Definition 1.4** (Strong Exponential Time Hypothesis)**.** $x_\infty = 1$.

Informally, the SETH is stronger than the ETH and conjectures that Sat cannot be solved in less than $O^*(2^n)$, that is it conjectures that the best exact algorithm for Sat is the brute-force algorithm, which currently fits the state of the art, since no better algorithm than the brute-force exists for Sat. If the SETH holds, then Sat cannot be solved in time $O^*((2-\varepsilon)^n)$. The SETH is also widely used to derive intractability results, generally in the parameterized complexity framework. We will present later such hardness results under the SETH. But note that the SETH is also used to derive intractability results for some problems in **P**, for example for the Diameter problem [Bon21], the 2-Sat problem [PW10], and in general in the fine-grained complexity framework [Wil15].

Now that we have presented some exact algorithms with the exponential blow-up in the size of the instance and the two main conjectures in this framework, we will focus in the rest of this section on parameterized algorithms.

We first consider parameterized algorithms parameterized by the solution size $k$, also called the *natural* parameter, and we present an algorithm with this parameter for the Max Min Vertex Cover problem, inspired by the algorithm of Boria et al. [BCP13] running in time $O^*(1.5874^k)$. This algorithm is a branching algorithm composed of two branching rules and one reduction rule.

In this algorithm, we keep track of the minimal vertex cover $C$ we are constructing. The algorithm applies the rules in order as long as the budget $k$ is not exceeded. For the first branching rule, consider a vertex $u \in V$ of degree $d(u) \geq 2$ such that all its neighbors have degree at least $d(u)$, and branch on the following sub-instances: either discard $u$ from $C$ and add all its neighbors in $C$; or, for every neighbor $v$ of $u$, discard $v$ from $C$ and add all its neighbors in $C$. This first rule is correct because the mandatory private structure of Max Min Vertex Cover is that every vertex in the solution

has at least one private edge, and in this branching rule we consider all cases: either $u$ is not in $C$ and all its neighbors are in $C$ with their common edge with $u$ as private edge; or $u$ is in $C$ and one of its neighbors $v$ is not in $C$ so that the edge $(u, v)$ is the private edge of $u$, and we consider all neighbors of $u$. Observe also that when a vertex is discarded, we add all its neighbors in $C$. For the second branching rule, consider a vertex $u$ of degree $d(u) \geq 2$ and with at least one of its neighbors, $v$, of degree 1, and branch on the following two sub-instances: either discard $v$ from $C$ and add $u$ to $C$; or discard $u$ from $C$ and add its neighbors in $C$. This rule is correct for the same argument as previously: either $v$ is in $C$ and the edge $(u, v)$ must be its private edge (since when we discard a vertex we add all its neighbors in $C$); or $v$ is discarded and $u$ must be in $C$.

Observe that if the two previous branching rules do not apply, the vertices left form a set of connected components, each of size at most 2. At this point, we apply the following reduction rule: if there is a connected component of size 2, add one of these two vertices in $C$; if there is a connected component of size 1, discard the corresponding vertex. Since in the two branching rules we add all the neighbors of a vertex we discard, the neighbors of the vertices left in the connected components are all in $C$. Thus, if there is a connected component with two vertices, only the edge between these two vertices is not dominated, and putting any one of these two vertices is sufficient to have a correct solution and to have the considered edge as private edge of the vertex taken, and if there is a connected component with one vertex, all its incident edges are already dominated, so the vertex must be discarded. So the reduction rule is correct.

Let us now determine the running time of this algorithm. For the first branching rule, the reduction is given by the expression $T(k) \leq (d(u) + 1)T(k-d(u))$, which has the worst case when $d(u) = 2$ and gives a complexity of $O^*(1.7321^k)$. For the second branching rule, the reduction is given by $T(k) \leq T(k-1)+T(k-2)$, which gives a complexity of $O^*(1.6181^k)$. Finally, the reduction rule can be executed in polynomial time. So the total running time of this algorithm is $O^*(1.7321^k)$.

This algorithm, similar to the algorithm of Boria et al. [BCP13], is simpler, and this is why we obtain a $O^*(1.7321^k)$ algorithm compared to the $O^*(1.5874^k)$ algorithm of Boria et al., which is the current best parameterized algorithm for Max Min Vertex Cover with $k$ as parameter. Interestingly, our parameterized algorithm with $k$ as parameter for Min Mixed Dominating Set we present in Section 2.3 is similar to this algorithm for Max Min Vertex Cover. Indeed, both use the private structure of the considered problem: for Max Min Vertex Cover, every vertex put in the solution has to have at least one neighbor not in the solution, in order to

have a private edge, and to satisfy the mandatory private structure of this problem; for Min Mixed Dominating Set, we put two neighbors of a vertex $u$ in the solution outside the solution in order for $u$ to have two private neighbors, to satisfy the implicit private structure of this problem we have mentioned in Section 1.2. We note also that, similarly to the algorithm for Max Min Vertex Cover, the problem Min Mixed Dominating Set can be solved in polynomial time when the vertices left at the end of the branching procedure form a set of connected components of size at most 2.

Since Max Min Vertex Cover admits a $O^*(1.7321^k)$ algorithm parameterized by the natural parameter, we say that Max Min Vertex Cover is in **FPT** with respect to this parameter. Indeed, for a decision problem $\Pi$ that admits an exact algorithm running in time $O(f(p)n^c)$ for a function $f$, a constant $c$ and a parameter $p$, we say that $\Pi$ is in **FPT** parameterized by $p$, and we say that such an algorithm is an FPT (for Fixed Parameter Tractable) algorithm for $\Pi$ with respect to the parameter $p$.

Two other well-studied classes in parameterized complexity are the **W[1]** and **W[2]** classes, which contain most of the problems for which it is widely believed that there is no FPT algorithm with the considered parameter, while the *W hierarchy* is $\mathbf{FPT} \subseteq \mathbf{W[1]} \subseteq \mathbf{W[2]} \subseteq \ldots \subseteq \mathbf{XP}$, where **XP** regroup the problems which can be solved in time $O(n^{f(p)})$ for a function $f$. Despite the original definitions of the two classes **W[1]** and **W[2]**, a characterization of these two classes using Turing machines was developed by Cesati in 2001 [Ces01]. By this result, two problems were proved to be **W[1]**-hard and **W[2]**-hard, respectively. Concerning the **W[1]**-hardness, the associated problem is to determine if a non-deterministic Turing machine accepts its input in $p$ steps for an integer $p$ while using a single tape. Concerning the **W[2]**-hardness, the associated problem is similar to the one for the **W[1]**-hardness, but with the key difference that this time the Turing machine has a multi-tape, which makes this problem harder that the one with a single tape, and indeed the **W[2]**-hard problems with respect to a parameter $p$ are harder than the **W[1]**-hard problems with the same parameter.

These two classes **W[1]** and **W[2]** are of particular interest since they characterize many well-known **NP**-hard problems for which there is currently no FPT algorithm for the considered parameter, which is widely believed and implied by the ETH, as we explain below. For example, with $k$ the natural parameter, the Independent Set and Clique problems are **W[1]**-hard, while the Dominating Set problem is **W[2]**-hard [DF99].

The two problems Clique and Dominating Set are generally used to prove the **W[1]**-hardness and **W[2]**-hardness, respectively, of the considered problem, using a *linear fpt reduction*. Informally, a linear fpt reduction with parameter $p$ between two decisions problems $\Pi_1$ and $\Pi_2$ is a reduction from

$\Pi_1$ to $\Pi_2$ such that there exists a solution for $\Pi_1$ with parameter $p$ if and only if there exists a solution for $\Pi_2$ with parameter $f(p)$ for a function $f$, and if this reduction can be made in linear FPT time $O(g(p)|I|)$ for a function $g$ and $|I|$ the input size of the problem $\Pi_1$. We explain below why such an fpt reduction has to be linear in order to derive intractability results in this framework.

Indeed, when proving that the considered problem is **W[1]**-hard or **W[2]**-hard parameterized by the natural parameter $k$, it is also interesting to prove that no algorithm can solve the considered problem in time $O(n^{o(k)})$ if it is **W[1]**-hard, and in time $O(n^{O(1)}m^{o(k)})$ if it is **W[2]**-hard, where $m$ is the size of the search space. The first problems for which such results were obtained are the Clique and Independent Set problems for the former hardness result, and which hold under the ETH, and the Set Cover, Hitting Set and Dominating Set problems for the latter hardness result, and which hold under **FPT** $\neq$ **W[1]**, in the extensive paper of Chen et al. [CHKX06]. To obtain these hardness results, Chen et al. introduced the notion of linear fpt reductions.

As we have mentioned before, it is crucial that such reductions are linear in order to derive the considered intractability results. Indeed, suppose we have an fpt reduction with parameter $p$ from $\Pi_1$ to $\Pi_2$ such that the size of the instance $I_2$ of $\Pi_2$ is $f(p)|I_1|^c$ for an instance $I_1$ of $\Pi_1$ and a constant $c > 1$, and suppose that there exists an FPT algorithm for $\Pi_2$ running in time $O(g(f(p))|I_2|^{c'})$. With this reduction and this algorithm, we can solve $\Pi_1$ in time $O(g(f(p))f(p)^{c'}|I_1|^{cc'})$. Thus, with such a running time, an $O(n^{o(k)})$ or $O(n^{O(1)}m^{o(k)})$ intractability for $\Pi_1$ cannot be transferred to $\Pi_2$, implying that the fpt reductions have to be linear to derive such hardness results. In Section 4.1, we present such a linear fpt reduction from Independent Set to Upper Dominating Set which allows us to derive the considered intractability result to the latter problem.

Another type of hardness result for **W[1]**-hard and **W[2]**-hard problem is under the SETH, initiated by Patrascu and Williams [PW10], and which states that Dominating Set cannot be solved in time $O(n^{k-\varepsilon})$ for a $\varepsilon > 0$. This result of Patrascu and Williams was obtained by a reduction from an instance of Sat to an instance of Dominating Set with sub-exponentially many vertices.

Until now, we have first focused on exact algorithms with the exponential blow-up in the size of the instance, after which we have presented the two widely used conjectures ETH and SETH, and after that we have focused on parameterized algorithms parameterized by the natural parameter while presenting the W hierarchy which holds for any parameter. We now present two other well-studied parameters, the *treewidth* and the *pathwidth*. These

parameters are defined through the notion of *tree decomposition*:

**Definition 1.5** (Tree decomposition)**.** *A* tree decomposition *of a graph* $G = (V, E)$ *is a tree* $T$ *with nodes* $X_1, \ldots, X_t$ *with* $t \leq |V|$*, each of which is a subset of* $V$*, satisfying the following properties:*

- $\bigcup_{i=1}^{t} X_t = V$*;*
- *If two nodes* $X_i$ *and* $X_j$ *both contain the vertex* $u$*, then all nodes* $X_k$ *of* $T$ *in the path from* $X_i$ *to* $X_j$ *contain* $u$ *as well;*
- *For every edge* $(u, v) \in E$*, there is a node* $X_i$ *that contains both* $u$ *and* $v$*.*

What we call *width* of a tree decomposition is the size of the largest set $X_i$ in the decomposition, minus one. The treewidth $tw(G)$ of the graph $G$ is the minimum width among all possible tree decompositions of $G$. The treewidth of a graph is a measure of how distant the considered graph is from being a tree. Similarly, the pathwidth of a graph is defined in the same way as the treewidth, via decompositions, with the difference that $T$ is in this case a path. Thus, the pathwidth of a graph is bigger or equal to the treewidth of the graph. And again, the pathwidth of a graph is a measure of how distant the considered graph is from being a path.

Interestingly, the problem of deciding if the treewidth (resp. the pathwidth) of a graph is at most $tw$ (resp. $pw$) is **NP**-hard [ACP87]. However, if the width $tw$ is a constant, there is an FPT algorithm which determines if the considered graph has a treewidth of width $tw$, where the dependence on the size of the instance is linear (although exponential in $tw$) [Bod96].

Nonetheless, these two parameters are mostly known for their tractability in FPT time. There are many **W[1]**-hard and **W[2]**-hard problems parameterized by the natural parameter for which there exists an FPT algorithm parameterized by the treewidth or the pathwidth. Furthermore, a meta-theorem designed by Courcelle [Cou90] states that all graph problems which are MSO-expressible can be solved in FPT time parameterized by the treewidth, in time linear on the size of the instance. This celebrated theorem is known as *Courcelle's Theorem*. Unfortunately, the exponential blow-up of Courcelle's Theorem in the treewidth parameter is usually super-exponential. Thus, since most of the **NP**-hard problems we are interested in are known to be FPT parameterized by the treewidth, obtaining the best algorithm for such problems, and in particular obtaining such an algorithm with the base of the exponential blow-up as small as possible, has become an interesting topic.

To obtain such algorithms, the common technique is to design a dynamic programming algorithm working on the tree decomposition associated with the given treewidth. To ease the production of such dynamic programming algorithms, specific tree decompositions were defined. The most noteworthy is the *nice* tree decomposition, which contains the following type of nodes: Leaf nodes which are leaf of the tree decomposition and which are empty sets; Introduce nodes which are nodes $X_t$ with one child $t'$ such that $X_t = X_{t'} \cup \{v\}$ for a vertex $v \notin X_{t'}$; Forget nodes which are nodes $X_t$ with one child $t'$ such that $X_t = X_{t'} \setminus \{w\}$ for a vertex $w \in X_{t'}$; and Join nodes which are nodes $t'$ with two children $t_1, t_2$ such that $X_t = X_{t_1} = X_{t_2}$. Note that path decompositions do not contain Join nodes since such nodes imply a tree-like structure.

On such nice tree decompositions, it suffices to prove for each type of nodes how the dynamic programming algorithm works to design an FPT algorithm for the considered problem parameterized by the treewidth. For example, the two problems Min Independent Dominating Set and Max Min Vertex Cover can be solved in time $O^*(3^{tw})$ by a result of van Rooij et al. [vRBR09], while their more natural Max and Min versions can be solved in time $O^*(2^{tw})$. In Section 2.4, we present a $O^*(5^{tw})$ algorithm for Min Mixed Dominating Set, obtained via a reduction, and in Section 4.2, we present a $O^*(6^{pw})$ algorithm for Upper Dominating Set, by modifying the dynamic programming algorithm of complexity $O^*(7^{pw})$ of Bazgan et al. [BBC+18a].

As we have mentioned earlier, obtaining the smaller base of the exponential blow-up is an interesting topic. It follows that designing hardness results on the existence of such FPT algorithms parameterized by the treewidth and the pathwidth have been studied. The first results obtained in this direction come from the extensive paper of Lokshtanov et al. [LMS18]. They proved that the known algorithms parameterized by the treewidth for Max Independent Set, Min Dominating Set, Max Cut and other problems are asymptotically optimal under the SETH. To obtain such hardness results, they provided reductions from Sat, which cannot be solved in time $O^*((2-\varepsilon)^n)$ under the SETH, to the considered problems, where the instances of the latter problems have the pathwidth upper-bounded by $n + O(1)$, thus implying that if the latter problem could be solved in time $O^*((c-\varepsilon)^{pw})$ for a constant $c$ depending on the problem, then this algorithm could solve Sat in time $O^*((2-\varepsilon)^n)$, contradicting the SETH. Since for a given graph $G$ we have $tw(G) \leq pw(G)$, such an hardness result holds also for the treewidth.

Such SETH-based intractability results have been obtained for a variety of **NP**-hard problems. Nonetheless, this method introduced by Lokshtanov et al. [LMS18] has the downside to design the reductions by beginning with

an instance of Sat for which we directly use a consequence of the SETH, that is the $O^*((2-\varepsilon)^n)$ intractability. Thus, such reductions designed this way had to go from a base of value 2 to a base of value $c$ for a constant $c$ depending on the problem. In particular, such reductions implied the grouping of several variables of the given instance of Sat, which make the reductions more involved. Fortunately, Lampis [Lam18] obtained a direct implication of the SETH to the problem $q$-CSP-$B$. In this problem, we are given a CSP instance with $n$ variables which can take values over a set of size $B$, and $m$ constraints such that each constraint contains at most $q$ variables and is given as a list of acceptable assignments of these variables, where an acceptable assignment is a $q$-tuple of values from the set $B$ given to the $q$ variables. Lampis proved that under SETH, for any constant $B \geq 2$ and any $\varepsilon > 0$, there exists a $q$ such that $q$-CSP-$B$ cannot be solved in time $O^*((B-\varepsilon)^n)$. With such an intractability result, it is now easier to design reductions to obtain hardness result for problems parameterized by the pathwidth, beginning with an instance of $q$-CSP-$B$ to the considered problem where the base of the target lower bound is equal to $B$. In Section 2.4 we give such an intractability result for Min Mixed Dominating Set matching our $O^*(5^{tw})$ algorithm we present in the same Section, and in Section 4.3 we give such an intractability result for Upper Dominating Set matching our $O^*(6^{pw})$ algorithm we present in Section 4.2.

We now finish this section, where we have presented the basic notions of the exact algorithms framework: an exact algorithm for the two problems Min Vertex Cover and Max Min Vertex Cover; the two complexity assumptions ETH and SETH; an FPT algorithm for Max Min Vertex Cover; the W hierarchy; and the treewidth and pathwidth and how to obtain positive and negative results for these parameters. In the next section, we focus on super-polynomial approximation.

## 1.5 Super-Polynomial Approximation

For any **NP**-hard problem, when we look simultaneously at an exact algorithm and an approximation algorithm, we observe that there exists a double gap between these two types of algorithms: one gap between the approximation ratio (1 for an exact algorithm and $r > 1$ for an approximation algorithm); and one gap between the time complexity (exponential for an exact algorithm and polynomial for an approximation algorithm). By this observation, the framework of *super-polynomial approximation* (also called sub-exponential approximation) was developed at the end of the 2000s, and aims to fill these two gaps by designing schemata of algorithms that can

achieve any approximation ratio lying on the first gap, in a time complexity bigger than polynomial, but smaller than the one of exact computation. In this section, we present the basic notions of this framework, from positive to negative results, which are relevant to the results we have obtained for the two problems Max Min Feedback Vertex Set and Upper Dominating Set.

More precisely, we present: (i) the $r$-approximation algorithm running in time $2^{O(n/r^2)}$ for Max Min Vertex Cover due to Bonnet et al. [BLP16] which is a good example of a super-polynomial approximation algorithm for a Max-Min problem, and which we mirror with our algorithm for Max Min Feedback Vertex Set of Section 3.5 and our algorithm for Upper Dominating Set of Section 4.4 (ii) the inapproximability of Max Independent Set obtained from the PCP Theorem, and an argument which explains why we consider that this inapproximability result "matches" the best known super-polynomial approximation algorithm for this problem (iii) the lower bound for Max Min Vertex Cover that matches the algorithm presented previously also due to Bonnet et al. [BLP16], and which is similar to the two inapproximability results in this framework we obtain for Max Min Feedback Vertex Set in Section 3.6 and for Upper Dominating Set in Section 4.5 (iv) the notion of *parameterized approximation*, some lower bounds obtained in this framework and how to derive such inapproximability results.

Let us begin by presenting the $r$-approximation algorithm, for any $r \leq \sqrt{n}$, of Bonnet et al. [BLP16], for the Max Min Vertex Cover, running in time $O^*(2^{O(n/r^2)})$. Recall that for this problem the private structure is that every vertex in the solution has to have at least one private edge dominated only by this vertex. This algorithm is a *divide-and-conquer* algorithm, which works as any divide-and-conquer algorithm in the following way: divide the initial instance into $x$ sub-instances; then execute a sub-exponential procedure on each of these sub-instances; and finally extend or combine in polynomial time the solutions found to obtain a solution of the whole graph.

The algorithm of Bonnet et al. [BLP16] works as follows, for any ratio $r \leq \sqrt{n}$: (1) compute a maximal matching $M$ (2) if $|M| \geq n/r$, output any minimal vertex cover (3) else, partition the set $V(M)$ into $r$ almost equal-sized sets $V_1, \ldots, V_r$, and let $I$ be the set of unmatched vertices (4) for each subset $V_i$, iterate through all subsets $Z \subseteq V_i$ such that $Z$ is an independent set (5) for each such independent set $Z$, consider the set $Z' = Z \cup (I \setminus N(Z'))$, let $C_Z = V \setminus Z'$, and remove vertices from $C_Z$ until it is minimal (6) at the end, output the best solution encountered.

Let us first determine the running-time of this algorithm. The first two steps can be executed in polynomial time. At Step (3), which occurs if

$|M| < n/r$, every subset $V_i$ has size at most $2n/r^2$. Thus, at Step (4), the number of independent sets in the considered set $V_i$ is at most $2^{O(n/r^2)}$. The rest of the algorithm works in polynomial time. Let us now prove that the solution output is a minimal vertex cover. At Step (2), if $|M| \geq n/r$, we output any minimal vertex cover. At Step (5), the set $Z'$ is an independent set since $Z$ is an independent set. So $C_Z = V \setminus Z'$ is a vertex cover of the given graph, and we remove vertices from $C_Z$ until it is minimal. So the solution output is a minimal vertex cover.

Let us now prove the approximation ratio. At Step (2), if $|M| \geq n/r$, then any minimal vertex cover will have to contain at least $|M|$ vertices, which directly gives an $r$-approximation. So suppose that $|M| < n/r$. Fix an optimal solution $C^*$, and let $R_i$ be the set of vertices of $V_i$, for $1 \leq i \leq r$, that do not belong to $C^*$, and similarly let $R_I$ be the set of vertices of $I$ not in $C^*$. Observe first that $R = R_I \cup \bigcup_{i=1}^{r} R_i$ is an independent set since $R = V \setminus C^*$, and furthermore we have $|C^*| \geq |N(R)|$. Now, observe that there exists an $1 \leq i \leq r$ such that $|N(R_I \cup R_i)| \geq |N(R)|/r$ by the facts that for two independent set $I_1, I_2$ such that $I_1 \cup I_2$ we have $N(I_1 \cup I_2) = N(I_1) \cup N(I_2)$, that $\sum_{i=1}^{r} |N(R_I) \cup N(R_i)| \geq |N(R)|$, and the pigeonhole principle. For this $i$, since we went through all independent sets $Z$ of $V_i$, we have tried the set $Z = R_i$. From this, we have build at Step (5) the set $Z' = R_i \cup (I \setminus N(R_i))$, and we have $R_I \cup R_i \subseteq Z'$ since $R_I \subseteq I \setminus N(R_i)$ since it contains no neighbor of $R_i$. Thus, we obtain $|N(Z')| \geq |N(R_I \cup R_i)| \geq |N(R)|/r$. Finally, observe that the solution $C_Z$ we have constructed from $Z'$ has size at least $|N(Z')|$ since all neighbors of $Z'$ must have been kept in $C_Z$ to remain a vertex cover. So we obtain that the solution output is bigger or equal to $|C_Z| \geq |N(Z')| \geq |N(R)|/r$, and thus the algorithm outputs an $r$-approximation.

Observe that this algorithm matches the $\sqrt{n}$-approximation algorithm for Max Min Vertex Cover we have described in Section 1.3, and allows us to obtain any ratio $r \leq \sqrt{n}$ in sub-exponential time. We note also that for the super-polynomial approximation for our considered problem Max Min Feedback Vertex Set we present in Section 3.5, we also use the divide-and-conquer technique, but this time by dividing the original graph into $\sqrt{r}$ equal-sized subsets $V_i$, and in the proof of the algorithm for Max Min Feedback Vertex Set we need to consider pairs of subsets $V_i$ instead of single subsets $V_i$, while using a parameterized approximation algorithm. Moreover, for the super-polynomial approximation for our other considered problem Upper Dominating set we present in Section 4.4, we also use the divide-and-conquer technique, where similarly to the algorithm for Max Min Vertex Cover we enumerate all independent sets for each subset $V_i$, as well as other subroutine algorithms.

As in the polynomial-time approximation and the parameterized frame-

works, it is of interest to prove some intractability results matching the known super-polynomial approximation algorithms. Interestingly, one of the first such inapproximability results was obtained for Max Independent Set by Chalermsook et al. [CLN13]. Similarly to the $n^{1-\varepsilon}$-inapproximability in polynomial time for this problem we have presented in Section 1.3, this super-polynomial inapproximability result was obtained using the PCP Theorem, via a gap-introducing reduction from an instance of 3-Sat, and states the following: under the ETH, for any $\varepsilon > 0$ and any sufficiently large $r > 1$, there cannot exist an $r$-approximation algorithm for Max Independent Set running in time $2^{(n/r)^{1-\varepsilon}}$.

Note that for the Max Independent Set problem, there exists the following super-polynomial approximation, obtained by Bourgeois et al. [BEP11]: for any $r < 1$, there exists an $r$-approximation algorithm for Max Independent Set running in time $O^*(\alpha^{rn})$, where the problem can be solved exactly in time $O^*(\alpha^n)$. In this manuscript, we consider that these two results "match", that is we consider that the super-polynomial approximation for Max Independent Set is the best we can hope under the ETH by the inapproximability of Chalermsook et al. [CLN13]. In fact, it is not totally exact since, in particular for Max Independent Set, we can improve the super-polynomial approximation with a $\log(r)$ factor. Indeed, Bansal et al. [BCL$^+$19] proved the following result: there is a randomized algorithm that outputs with constant probability an $r$-approximation for Max Independent Set running in time $O^*(e^{(\tilde{O}(n/(r\log^2 r)+r\log^2 r)})$[4]. This result of Bansal et al. [BCL$^+$19] prove that the inapproximability result of Chalermsook et al. [CLN13] is not "tight" compared to the super-polynomial approximation of Bourgeois et al. [BEP11].

Nonetheless, we consider that the super-polynomial approximation of Bourgeois et al. [BEP11] and the corresponding inapproximability of Chalermsook et al. [CLN13] "match" for the following argument: with the current PCP constructions, we are unable to distinguish between running times $2^{n/r}$ and $2^{n/(r\log r)}$. Indeed, the current most efficient PCP constructions are quasi-linear: they reduce an instance of 3-Sat with $n$ variables into an instance of Max-3-Sat of size $n\log(n)^{O(1)}$ in which there is a gap on the fraction of clauses satisfied. Thus, under the ETH, these PCP constructions give an intractability result with complexity $2^{o(n/(\log(n)^{O(1)}))}$, which we prefer to write as $2^{n^{1-\varepsilon}}$ since only a linear PCP construction would allow us to distinguish between these two running times. We make such a consideration for any problem we consider. Moreover, when we encounter such an intractability result, we specifically say that it "matches" the associated algorithm, whereas

[4] $\tilde{O}$ notation suppresses $\log(\log(r))^c$ factors for a constant $c$.

for intractability results which are not based on the PCP we say they are "tight" (for example for the intractability results for pathwidth and treewidth under the SETH).

We now present a gap-preserving reduction from Max Independent Set to Max Min Vertex Cover, designed by Bonnet et al. [BLP16], which allowed them to derive an inapproximability result for the latter problem that matches the super-polynomial approximation we have described earlier for this problem. For this sake, we recall some details of the reduction of Chalermsook et al. [CLN13] to obtain the hardness result for Max Independent Set. They obtained an instance $G = (V, E)$ of Max Independent Set with $|V| = n^{1+\varepsilon}r^{1+\varepsilon}$ and such that: if the original instance $\phi$ of 3-Sat with $n$ variables is a Yes-instance, then $\alpha(G) \geq n^{1+\varepsilon}r$, where $\alpha(G)$ denotes the maximum size of an independent set in $G$; and if $\phi$ is a No-instance, then $\alpha(G) \leq n^{1+\varepsilon}r^{2\varepsilon}$. Thus, if there would be an $r^{1-2\varepsilon}$-approximation algorithm for Max Independent Set running in time $2^{(n/r)^{1-\varepsilon}}$, then we would be able to determine if the original instance $\phi$ of 3-Sat is satisfiable or not in sub-exponential time, contradicting the ETH.

From this instance $G = (V, E)$ of Max Independent Set and with any ratio $r \leq \sqrt{n}$, we construct an instance $G' = (V', E')$ of Max Min Vertex Cover, similarly as the reduction we made between these two problems in Section 1.3 to obtain the polynomial-time inapproximability for Max Min Vertex Cover, in the following way: for every vertex $u \in V$, we add $r$ vertices connected only to $u$.

For the first direction of the proof, suppose $G$ admits a maximum independent set $I$ of size at least $n^{1+\varepsilon}r$ (Yes-instance). We construct a minimal vertex cover by taking all vertices of $V \setminus I$ and the $r$ newly vertices of every vertex in $I$. The solution is clearly a vertex cover of $G'$. Let us now show that the solution satisfies its mandatory private structure: since $I$ is maximum, it is maximal, so every vertex in $V \setminus I$ has at least one neighbor in $I$ and thus a private edge; and for a vertex $u \in I$, the $r$ newly vertices connected to $u$ in the vertex cover have their common edge with $u$ as private edge. Now, the size of this minimal vertex cover is at least $r\alpha(G) \geq n^{1+\varepsilon}r^2$. For the other direction, consider any minimal vertex cover $C$ of $G'$. For every vertex $u \in V \setminus C$, at most the $r$ newly vertices connected to $u$ are in $C$. And at most $|V|$ vertices are in $V \cap C$. Thus, since in this case (No-instance) every independent set has size at most $n^{1+\varepsilon}r^{2\varepsilon}$, we have that $|C| \leq |V| + r\alpha(G) \leq n^{1+\varepsilon}r^{1+\varepsilon} + n^{1+\varepsilon}r^{1+2\varepsilon} \leq 2n^{1+\varepsilon}r^{1+2\varepsilon}$.

Thus, an approximation algorithm with ratio $r' = r^{1-2\varepsilon}/2$ would determine if the original instance of 3-Sat is satisfiable or not. Now, since $|V'| = r|V| = n^{1+\varepsilon}r^{2+\varepsilon}$, and by adjusting correctly $r$ and $\varepsilon$, if this algorithm works in time $2^{(n/r^2)^{1-\varepsilon}}$, then we would be able to solve 3-Sat in

sub-exponential time, contradicting the ETH.

So we obtain that, for Max Min Vertex Cover, there is no $r$-approximation for any $r \leq \sqrt{n}$ running in time $2^{(n/r^2)^{1-\varepsilon}}$ under the ETH. Observe that this inapproximability of Max Min Vertex Cover matches the super-polynomial approximation algorithm we have presented earlier in this section and due also to Bonnet et al. [BLP16].

In the reduction from Max Independent Set to Max Min Vertex Cover, the gap is preserved from the former problem to the latter, and the order of $G'$ is quadratic in $r$ compared to $G$, thus giving the right inapproximability result. Nonetheless, sometimes the blow-up in the size of the instance is too big, quadratic or worse, compared to the blow-up in the gap. To answer this difficulty, a solution is to *sparsify* the instance obtained. One way to do this is to make a deterministic sparsification, that is to sparsify the instance by removing vertices in a deterministic way such that the blow-up on the size of the instance is chosen accordingly to how the gap is transferred from the first problem to the second. Another way to do this is to make a *probabilistic* sparsification, that is to sparsify the instance by removing vertices depending on a probability. Then, it is possible to use a Chernoff bound which ensures that the blow-up in the size of the instance is contained, while preserving the gap. Generally, the following bound is used for this sake: suppose that $X = \sum_{i=1}^{p} X_i$ is the sum of $p$ independent random 0/1 variables $X_i$, and that $E[X] = \sum_{i=1}^{p} E[X_i] = \mu$, then for all $0 \leq \delta \leq 1$, we have $Pr[|X - \mu| \geq \delta\mu] \leq 2e^{-\mu\delta^2/3}$. In Section 3.6, we use this Chernoff bound along with the inapproximability of Chalermsook et al. [CLN13] for Max Independent Set to derive a super-polynomial inapproximability for our considered problem Max Min Feedback Vertex Set, and in Section 4.5 we do the same for Upper Dominating Set.

Until now, we have presented in this section the framework of super-polynomial approximation through the eye of exact algorithms with the exponential blow-up in the size of the instance, to obtain trade-offs between such algorithms and polynomial-time approximation. Interestingly, we can apply the same kind of reasoning to obtain trade-offs between exact parameterized algorithms and polynomial-time approximation. This framework is called *parameterized approximation*.

Similarly as with exact algorithms with the blow-up in the size of the instance, the main method to design such schemata of parameterized approximation algorithms is the divide-and-conquer method by splitting the instance into $x$ sub-instances, run an FPT subroutine on each sub-instance, and finally combine the solutions found to obtain a solution of the whole instance. This method is used in the same way they are used to develop super-polynomial algorithms with the sub-exponential blow-up in the size

of the instance, with the difference that in this case an FPT algorithm is executed instead, and which depends on the parameter considered.

We now focus on the parameterized approximation with the natural parameter $k$. On the intractability side of parameterized approximation algorithms, a commonly used negative result is again for the Independent Set problem. This result, found by Bonnet et al. [BEKP13], is again obtained via gap-introducing reduction from an instance of 3-Sat to an instance of Independent Set, using the PCP Theorem. This result states that under the ETH, for any $\varepsilon > 0$ and any constant $r > 0$, there is no $r$-approximation algorithm for Independent Set running in time $O(n^{k^{1-\varepsilon}})$. In this reduction from 3-Sat to Independent Set of Bonnet et al. [BEKP13], the instance $G = (V, E)$ of Independent Set obtained has a specific form: the set of vertices $V$ is partitioned into $k$ subsets $V_1, \ldots, V_k$, each being a clique, and with $k$ the natural parameter. In Section 4.1, we use this parameterized inapproximability result for Independent Set and the structure of the instance $G$ from the reduction of Bonnet et al. [BEKP13] to derive the same inapproximability result for our considered problem Upper Dominating Set, via a linear fpt reduction.

Note that the result of Bonnet et al. [BEKP13] holds for any ratio $r$ being a constant, and under the ETH. Very recently, this hardness result for Independent Set was improved by Lin [Lin21] for the Clique problem: under $\mathbf{FPT} \neq \mathbf{W[1]}$, and any constant $r > 1$, there is no algorithm that outputs an $r$-approximation for Clique running in FPT time $f(k)n^{O(1)}$. To obtain better intractability results with the ratio not only constant, there is the hardness result of Chen and Lin [CL15] for Dominating Set stating that for any ratio $r$ constant or super-constant, there is no $r$-approximation for this problem running in FPT time, again under the complexity assumption $\mathbf{FPT} \neq \mathbf{W[1]}$. To obtain even stronger intractability results, that is with a ratio being a function of the natural parameter $k$, there is the hardness result of Chalermsook et al. [CCK+17] for Clique and Dominating Set, stating that there is no $o(k)$-approximation for Clique or $f(k)$-approximation for any function $f$ for Dominating Set that runs in FPT time, under the Gap Exponential Time Hypothesis (Gap-ETH). The conjecture Gap-ETH is stronger that the more natural ETH and can be formulated as follows: there exists a constant $\varepsilon > 0$ such that no algorithm running in time $O^*(2^{o(m)})$ can distinguish for an instance $\phi$ of 3-Sat if $\phi$ is satisfiable or if at most $(1 - \varepsilon)$ fraction of the $m$ clauses of $\phi$ are satisfied. We note that the result of Lin [Lin21] for Clique is stronger, in some sense, than the one of Chalermsook et al. [CCK+17] since it holds under $\mathbf{FPT} \neq \mathbf{W[1]}$, whereas the one of Chalermsook et al. holds under the Gap-ETH.

We now finish this section, where we have presented the basic notions

of the super-polynomial approximation framework: an $r$-approximation algorithm for Max Min Vertex Cover for any $r \leq \sqrt{n}$ running in time $2^{O(n/r^2)}$; the inapproximability of Max Independent Set along with a discussion of what is a "matching" inapproximability result in this framework; a matching lower bound for Max Min Vertex Cover under the ETH; and finally the notion of parameterized complexity, and in particular inapproximability results in this framework. In the next Section, we give some notations.

## 1.6 Notations

In this section, we recall some basic definitions about graph theory, boolean formulas, CSPs and Chernoff bounds.

### Graphs

The definitions here can be found in the book of Berge [Ber73]. An *undirected graph* $G$ is a pair $(V, E)$, where $V$ is a finite set of elements called *vertices*, and $E \subseteq \binom{V}{2}$ is a set of pairs of vertices called *edges*. We denote by $V(G)$ the vertex set of the graph $G$ if there is ambiguity, and $E(G)$ the edge set, for the same reason. An edge $e$ between two vertices $u$ and $v$ is denoted $(u, v)$. A directed graph is a pair $(V, A)$, where $V$ is a finite set of vertices, and $E \subseteq V \times V$ is a set of *arcs*. An arc is noted $(u, v)$. A (*directed*) path is a sequence of vertices $u_1, \ldots, u_x$ such that, for each $1 \leq i \leq x-1$, $(u_i, u_{i+1}) \in E$ ($(u_i, u_{i+1}) \in A$). A *cycle* is a path with $u_1 = u_x$. A *directed acyclic graph* (DAG) is a directed graph with no cycle. Except when we will mention it explicitly, a *graph* will be an undirected graph.

In an undirected graph $G = (V, E)$ without parallel edges, the two vertices $u$ and $v$ are *neighbors* if $(u, v) \in E$. We also say that $v$ is a neighbor of $u$, and vice versa. In a directed graph $G = (V, A)$, $v$ is a neighbor of $u$ if $(u, v) \in A$. The set of neighbors of a vertex $u$ in an undirected or directed graph is denoted $N(u)$, or $N_G(u)$ in case of ambiguity, and is called the *open neighborhood* of $u$. $N[u] = N(u) \cup \{u\}$ is called the *closed neighborhood* of $u$. For any subset of vertices $U \subseteq V$, $N[U]$ is defined as $\bigcup_{u \in U} N[u]$, and $N(U) = N[U] \setminus U$. The degree $d(u)$ of a vertex $u$ is the size of $N(u)$. We use $\Delta(G)$ (or simply $\Delta$) to denote the maximum degree of $G$.

A *subgraph* of a graph $G = (V, E)$ is a graph $G' = (V', E')$ such that $V' \subseteq V$ and $E' \subseteq E$. An *induced subgraph* of $G = (V, E)$ is a subgraph $G' = (V', E')$ of $G$ such that $V' \subseteq V$ and each edge of $E$ having both endpoints in $V'$ is in $E'$. The *subgraph induced by* $U \subseteq V$, denoted by $G[U]$, is the induced

subgraph of $G$ of the form $(V', E')$ with $U = V'$. The *complementary* $\overline{G}$ of $G$ is defined by $\overline{G} = (V, \overline{E})$, where $\overline{E} = \{(u, v) | (u, v) \notin E\}$.

For $u \in V$, $G - u$ is the graph $G[V \setminus \{u\}]$. For $(u, v) \in E$ the graph $G/uv$ is the graph obtained by contracting the edge $(u, v)$, that is, replacing $u, v$ by a new vertex connected to $N(u) \cup N(v)$.

An *independent set* $I \subseteq V$ is a subset of vertices of $V$ such that for all $u, v \in I$, $(u, v) \notin E$. A *clique* $K \subseteq V$ is a subset of vertices of $V$ such that for all $u, v \in K$, $(u, v) \in E$. A *dominating set* $D \subseteq V$ is a subset of vertices of $V$ such that for all $u \in V \setminus D$, there is a vertex $v \in N(u) \cap D$. A *vertex cover* $C \subseteq V$ is a subset of vertices of $V$ such that for all $e = (u, v) \in E$, at least one of $u$ and $v$ is in $C$. A *forest* is an acyclic graph, that is, a graph without any cycle, and a *tree* is a connected forest.

An hypergraph $G$ is a pair $(V, F)$, where $V$ is a finite set of elements called *vertices*, and $F \subseteq 2^V$ is a set of tuples of vertices called *hyperedges*. The definitions given for an undirected graph hold for an hypergraph.

## Boolean Formulas

A *boolean variable* is a variable that can be either True or False. A *boolean formula* is either a boolean variable, either the negation ($\neg$) of a boolean formula, either the conjunction ($\wedge$) of two boolean formulas, or the disjunction ($\vee$) of two boolean formulas. As $\vee$ and $\wedge$ are associative, we can extend these two operations to any arity. A *literal* is a variable or the negation of a variable. A *clause* is a disjunction of literals. A *conjunctive normal form* (CNF for short) formula is a boolean formula which is the conjunction of clauses. A *truth assignment* is a function mapping each variable to True or False. A *satisfying* truth assignment is a truth assignment which makes the formula to be True. A formula is *satisfiable* if there is a satisfying truth assignment.

## CSPs

A *constraint satisfaction problem* is defined as a triple $(X, D, C)$, where: $X = \{x_1, \ldots, x_n\}$ is a set of variables; $D$ is an *alphabet* and is generally a finite subset of $\mathbb{N}$; $C = \{c_0, \ldots, c_{m-1}\}$ is a set of constraints. Each variable $x_i$, $1 \leq i \leq n$, can take value over the alphabet $D$. An *evaluation* of the variables is a function $f : X \to D$ from the variables to the alphabet that sets values of $D$ to the variables of $X$. Each constraint $c_j$ is a list of acceptable assignments of the variables appearing in $c_j$, that is it is a list of pairs $(Y, b)$, where $Y \subseteq X$ is a subset of $q$ variables, for $1 \leq q \leq n$, and $b$ is a $q$-ary relation setting the values from $D$ to the $q$ variables of $Y$ appearing in $c_j$. We say

that a constraint $c_j$ is *satisfied* if the evaluation of the variables satisfies at least one pair $(Y, b)$ in the list of the constraint $c_j$. We say that an evaluation *satisfies* the problem if all constraints are satisfied by this evaluation, and we say that the problem is *satisfiable* if there exists an evaluation that satisfies all constraints.

## Chernoff bounds

Chernoff bounds, named after Herman Chernoff, give, in probability theory, exponentially decreasing bounds on distributions of the sums of independent 0/1 random variables. In contrary to Markov's inequality and Chebyshev's inequality, Chernoff bounds require the random variables to be independent.

In this manuscript, we consider the multiplicative form of the Chernoff bounds, which is given as follows: we are given $p$ independent 0/1 random variables $X_1, \ldots, X_p$, with $X = \sum_{i=1}^{p} X_i$ and $E[X] = \mu$ the expected value of the sum of these $p$ variables. The multiplicative Chernoff bound is given as follows, for any $0 \leq \delta \leq 1$:

$$Pr[|X - \mu| \geq \delta\mu] \leq \left( \frac{e^{\delta}}{(1+\delta)^{1+\delta}} \right)^{\mu}$$

We use a less strong form of the multiplicative Chenoff bound, given as follows, for any $0 \leq \delta \leq 1$:

$$Pr[|X - \mu| \geq \delta\mu] \leq 2e^{-\mu\delta^2/3}$$

# Chapter 2

# Min Mixed Dominating Set

Domination problems in graphs are one of the most well-studied topics in theoretical computer science. In this chapter we study a variant called Min Mixed Dominating Set: we are given a graph $G = (V, E)$ and are asked to select $D \subseteq V$ and $M \subseteq E$ such that $|D \cup M|$ is minimized and the set $D \cup M$ dominates $V \cup E$, where a vertex dominates itself, its neighbors, and its incident edges and an edge dominates itself, its endpoints, and all edges with which it shares an endpoint.

The notion of Min Mixed Dominating Set was first introduced in 1977 by Alavi et al. [ABLN77], and has been studied extensively in graph theory [ALWZ92, EM77, Mei78, PS94]. See the chapter in [HHS98] for a survey on the Min Mixed Dominating Set problem. The computational complexity of Min Mixed Dominating Set was first studied in 1993 by Majumbar [Maj93], where he showed that the problem is **NP**-complete. The problem remains **NP**-complete on split graphs [ZKS11] and on planar bipartite graphs of maximum degree 4 [Man99]. Majumbar [Maj93], Lan and Chang [LC13], Rajaati et al. [RSS+17] and Madathil et al. [MPSS19] showed that the problem is polynomial-time solvable on trees, cacti, generalized series-parallel graphs and proper interval graphs, respectively.

Min Mixed Dominating Set is a natural variation of domination in graphs as it can be seen as a *mix* between four standard problems: Min Dominating Set, where vertices dominate vertices; Min Edge Dominating Set, where edges dominate edges; Min Vertex Cover, where vertices dominate edges; and Min Edge Cover, where edges dominate vertices. In Min Mixed Dominating Set we are asked to select vertices *and* edges in a way that dominates all vertices *and* edges. As only the last of these four problems is in **P**, it is not surprising that Min Mixed Dominating Set is **NP**-hard.

On the polynomial-time approximation algorithms side, Hatami gave a

2-approximation algorithm [Hat10]. Interestingly, this algorithm is based on a maximum matching computation, as the 2-approximation of Min Vertex Cover and the 2-approximation of Min Edge Dominating Set. As we have mentioned in Section 1.3, Dudycz et al. [DLM19] showed that, under the Unique Games Conjecture, no algorithm can achieve a ratio better than 2 for the Min Edge Dominating Set problem, result they have obtained by slightly modifying the $(2-\varepsilon)$-inapproximability of Min Vertex Cover obtained by Khot and Regev [KR08]. As we explain in the following section (Proposition 2.1), this hardness result easily carries over to Min Mixed Dominating Set, thus essentially settling the polynomial approximability of the problem.

Thus we are interested in exact algorithms for Min Mixed Dominating Set. In this context, this problem has recently been the focus of several works. Concerning exact algorithms with the blow-up in the size of the instance, an $O^*(3^n)$ algorithm is easily obtained by the following observation: it is safe to assume that the optimal solution has a specific structure where the selected edges form a matching whose endpoints are disjoint from the set of selected vertices. This algorithm was recently improved by Madathil et al. [MPSS19] who obtained an $O^*(2^n)$ algorithm using $O^*(2^n)$ space, by a dynamic programming algorithm. In Section 2.2, we design an improved exact algorithm of complexity $O^*(1.912^n)$ using only polynomial space. To obtain this algorithm, we observe (Lemma 2.3) that there always exists an optimal solution which has a specific form: the endpoints of the selected edges are disjoint from the set of selected vertices; and every vertex $u$ selected has two private neighbors, that is two vertices not involved in the solution and dominated only by $u$. We call such a solution a *nice* solution. In contrary to the observation which implies the $O^*(3^n)$ exact algorithm, letting the selected edges be an edge cover and not a matching allows us to obtain the fact that every vertex $u$ taken has two private neighbors. This definition of nice mixed dominating sets is the private structure of the problem Min Mixed Dominating Set we have briefly presented in the Introduction, and our $O^*(1.912^n)$ exact algorithm is a branching algorithm that uses extensively this implicit private structure in the different branching rules, enabling us to obtain this improved algorithm.

In the context of FPT algorithms, Min Mixed Dominating Set has also recently been the focus of several works. With respect to the natural parameter (the size of the solution $k$), an $O^*(7.465^k)$ algorithm was given by Jain et al. [JJPS17], more recently improved to $O^*(4.172^k)$ by Xiao and Sheng [XS19]. In Section 2.3, we give an improved FPT algorithm parameterized by $k$ running in time $O^*(3.510^k)$. As for our exact algorithm, this FPT algorithm is a branching algorithm which uses the implicit private structure

of the solution in a way that speeds up the branching on low-degree vertices.

With respect to the treewidth $tw$ and the pathwidth $pw$, Jain et al. [JJPS17] gave algorithms running in $O^*(6^{tw})$ time and $O^*(5^{pw})$ time, improving upon the $O^*(3^{tw^2})$ algorithm of Rajaati et al. [RHDS18]. Furthermore, Jain et al. [JJPS17] showed that no algorithm can solve Min Mixed Dominating Set in time $O^*((2-\varepsilon)^{pw})$ under the Set Cover Conjecture (see [CDL+12] for more details about the Set Cover Conjecture). In Section 2.4, we first improve the FPT algorithm parameterized by the treewidth by giving a $O^*(5^{tw})$ algorithm. Interestingly, this result is directly obtained from the observation that Min Mixed Dominating Set is equivalent to Min Distance-2-Dominating Set in the incidence graph. Finally, we prove that our algorithm for treewidth and the one of Jain et al. [JJPS17] for pathwidth are optimal under the SETH. Indeed, we prove that for any $\varepsilon > 0$ Min Mixed Dominating Set cannot be solved in time $O^*((5-\varepsilon)^{pw})$ under the SETH. To obtain this result, we make a reduction from the $q$-CSP-5 problem to our problem, as we have briefly presented in Section 1.5.

In the next Section, we begin by giving the $(2-\varepsilon)$-inapproximability of Min Mixed Dominating Set under the UGC and defining the notion of *nice* mixed dominating set which gives us the implicit private structure of Min Mixed Dominating Set we will use in our subsequent algorithms.

## 2.1 Inapproximability and Nice Solution

We begin by noting that the problem Min Mixed Dominating Set is harder than the more well-studied Min Edge Dominating Set problem, by a reduction that preserves most parameters from an FPT viewpoint and the size of the optimal solution. Hence, essentially all hardness results for the latter problem, such as its $(2-\varepsilon)$-inapproximability under the UGC obtained by Dudycz et al. [DLM19] or its **W[1]**-hardness for clique-width from Fomin et al. [FGLS10], carry over to Min Mixed Dominating Set.

**Proposition 2.1.** *There is an approximation and parameter-preserving reduction from Min Edge Dominating Set to Min Mixed Dominating Set.*

*Proof.* Given an instance $G = (V, E)$ of Min Edge Dominating Set we seek a set $M$ of $k$ edges such that all edges have an endpoint in $V(M)$. We add a new vertex $u$ connected to all of $V$ and attach to $u$ $|V|+2$ leaves. The new graph has a mixed dominating set of size $k+1$ if and only if $G$ has an edge dominating set of size $k$. □

We now define a restricted notion of mixed dominating set.

**Definition 2.2.** *A* nice *mixed dominating set of a graph $G = (V, E)$ is a mixed dominating set $D \cup M$ which satisfies the following: (i) $D \cap V(M) = \emptyset$ (ii) for all $u \in D$ there exists at least two private neighbors of $u$, that is, two vertices $v_1, v_2 \in V \setminus (D \cup V(M))$ with $N(v_1) \cap D = N(v_2) \cap D = \{u\}$.*

We note that a similar notion of nice mixed dominating set was used in the algorithms of Jain et al. [JJPS17], with the key difference that these algorithms do not use the fact that every vertex of $D$ must have at least two *private* neighbors, that is, two neighbors which are dominated only by this vertex, though these algorithms use the fact that such vertices have at least one private neighbor.

Let us now prove that restricting ourselves to nice solutions does not change the value of an optimal solution, which gives us the implicit private structure of Min Mixed Dominating Set we use extensively in our following algorithms. The idea of the proof is to reuse the arguments of Madathil et al. [MPSS19] to obtain an optimal solution satisfying the first property; and then, while there exists $u \in D$ with at most one private neighbor, we replace it by an edge while maintaining a valid solution satisfying the first property.

**Lemma 2.3.** *For any graph $G = (V, E)$ without isolated vertices, $G$ has a mixed dominating set $D \cup M$ of size at most $k$ if and only if $G$ has a nice mixed dominating set $D' \cup M'$ of size at most $k$.*

*Proof.* One direction is trivial, since any nice mixed dominating set is also by definition a mixed dominating set. For the other direction, we first recall that it was shown by Madathil et al. [MPSS19] that if a graph has a mixed dominating of size $k$, then it also has such a set that satisfies the first condition of Definition 2.2. Suppose then that $D \cup M$ is such that $D \cap V(M) = \emptyset$.

We will now edit this solution so that we obtain the missing desired property, namely the fact that all vertices of $D$ have two private neighbors. Our transformations will be applicable as long as there exists a vertex $u \in D$ without two private neighbors, and will either decrease the size of the solution, or decrease the size of $D$, while maintaining a valid solution satisfying the first property of Definition 2.2. As a result, applying these transformations at most $n$ times yields a nice mixed dominating set.

Let $I = V \setminus (D \cup V(M))$. If there exists $u \in D$ with exactly one private neighbor, let $v \in I$ be this private neighbor. We set $D' = D \setminus \{u\}$ and $M' = M \cup \{(u, v)\}$ to obtain another solution. This solution is valid because $N(u) \setminus \{v\}$ is dominated by $(D \cup M) \setminus \{u\}$, otherwise $u$ would have more that one private neighbors.

Let us now consider $u \in D$ such that $u$ has no private neighbor. If $N(u) \subseteq D$, then we can simply remove $u$ from the solution and obtain a better solution (recall that $u$ is not an isolated vertex). Otherwise, let $v \in N(u) \setminus D$. We set $D' = D \setminus \{u\}$ and $M' = M \cup \{(u, v)\}$ to obtain another feasible solution with fewer vertices, while still satisfying the first property. We repeat these modifications until we obtain the claimed solution. □

In the remainder, when considering a nice mixed dominating set $D \cup M$ of a graph $G = (V, E)$, we will associate with it the partition $V = D \cup P \cup I$ where $P = V(M)$ and $I = V \setminus (D \cup P)$. We will call this a *nice mds partition*. We have the following properties: (i) $M$ is an edge cover of $G[P]$ since $P = V(M)$ and $M$ is a set of edges (ii) $I$ is an independent set because if there were two adjacent vertices in $I$ then the edge between them would not be dominated (iii) $D$ dominates $I$ because if there was a vertex in $I$ not dominated by $D$ it would not be dominated at all (iv) each $u \in D$ has two private neighbors $v_1, v_2 \in I$, that is $N(v_1) \cap D = N(v_2) \cap D = \{u\}$.

We also note the following useful relation we will use in our exact algorithm we present in Section 2.2.

**Lemma 2.4.** *For any graph $G = (V, E)$ and any nice mds partition $V = D \cup P \cup I$ of $G$, there exists a minimal vertex cover $C$ of $G$ such that $D \subseteq C \subseteq D \cup P$.*

*Proof.* Since $I$ is an independent set of $G$, $D \cup P$ is a vertex cover of $G$ and hence contains some minimal vertex cover. We claim that any such minimal vertex cover $C \subseteq D \cup P$ satisfies $D \subseteq C$. Indeed, for each $u \in D$ there exists two private neighbors $v_1, v_2 \notin D \cup P$. Hence, if $u \notin C$, the edge $(u, v_1)$ is not covered, contradiction. □

In the next section, we present our improved exact algorithm running in time $O^*(1.912^n)$ which extensively uses the private structure issued from Lemma 2.3.

## 2.2 Improved Exact Algorithm

Let us first give an overview of our algorithm. Consider an instance $G = (V, E)$ of the Min Mixed Dominating Set problem and fix, for the sake of the analysis, an optimal solution which is a nice mixed dominating set $D \cup M$. Such an optimal solution must exist by Lemma 2.3, so suppose it gives the nice mds partition $V = D \cup P \cup I$.

By Lemma 2.4, there exists a minimal vertex cover $C$ of $G$ for which $D \subseteq C \subseteq D \cup P$. Our first step is to "guess" $C$, by enumerating all minimal

vertex covers of $G$. This decreases our search space, since we can now assume that vertices of $C$ only belong to $D \cup P$, and vertices of $V \setminus C$ only belong to $P \cup I$.

For our second step, we branch on the vertices of $V$, placing them in $D$, $P$, or $I$. The goal of this branching is to arrive at a situation where our partial solution dominates $V \setminus C$. The key idea is that, using the implicit private structure of Min Mixed Dominating Set we have obtained through the definition of nice mixed dominating set (Definition 2.2 and Lemma 2.3), any vertex of $C$ that may belong to $D$ must have at least two private neighbors, hence this allows us to significantly speed up the branching for low-degree vertices of $D$. Finally, once we have a partial solution that dominates all of $V \setminus C$, we show how to complete this optimally in polynomial time using a minimum edge cover computation.

We now describe the three steps of our algorithm in order and give the properties we are using step by step. In the remainder we assume that $G$ has no isolated vertices (since these are handled by taking them in the solution). Therefore, by Lemma 2.3 there exists an optimal nice mixed dominating set. Denote the corresponding partition as $V = D \cup P \cup I$.

**Step 1:** Enumerate all minimal vertex covers of $G$, which takes time $O^*(3^{n/3})$ by a result of Moon and Moser [MM65], as the algorithm we have presented in Section 1.4 for Min Vertex Cover and Max Min Vertex Cover. For each such vertex cover $C$ we execute the rest of the algorithm. In the end we output the best solution found.

Thanks to Lemma 2.4, there exists a minimal vertex cover $C$ with $D \subseteq C \subseteq D \cup P$. Since we will consider all minimal vertex covers, in the remainder we focus on the case where the set $C$ considered satisfies this property. Let $Z = V \setminus C$. Then $Z$ is an independent set of $G$. We now get two properties we will use in the branching step of our algorithm:

1. For all $u \in C$, $u$ can be either in $D$ or in $P$, because $C \subseteq D \cup P$.

2. For all $v \in Z$, $v$ can be either in $P$ or in $I$, because $D \subseteq C$.

**Step 2:** Branch on the vertices of $V$ as described below.

The branching step of our algorithm will be a set of Reduction and Branching Rules over the vertices of $C$ or $Z$. In order to describe a recursive algorithm, it will be convenient to consider a slightly more general version of the problem: in addition to $G$, we are given three disjoint sets $D_f, P_f, P'_f \subseteq V$, and the question is to build a nice mds partition $V = D \cup P \cup I$ of minimum cost which satisfies the following properties: $D_f \subseteq D \subseteq C$, $P_f \subseteq P \cap C$, and $P'_f \subseteq P \cap Z$. Clearly, if $D_f = P_f = P'_f = \emptyset$ we have the original problem and

all properties are satisfied. We will say that a branch where all properties are satisfied is *good*, and our proof of correctness will rely on the fact that when we branch on a good instance, at least one of the produced branches is good. The intuitive meaning of these sets is that when we decide in a branch that a vertex belongs to $D$ or to $P$ in the optimal partition we place it respectively in $D_f$, $P_f$ or $P'_f$ (depending on whether the vertex belongs in $C$ or $Z$).

We now describe a series of Rules which, given an instance of Min Mixed Dominating Set and three sets $D_f, P_f, P'_f$, will recursively produce sub-instances where vertices are gradually placed into these sets. Our algorithm will consider the Reduction and Branching Rules in order and apply the first Rule that can be applied. Note that we say that a vertex $u$ is "decided" if it is in one of the sets $D_f \subseteq D$, $P_f \subseteq P$, or $P'_f \subseteq P$. All the other vertices are considered "undecided".

Throughout the description that follows, we will use $U$ to denote the set of undecided vertices which are not dominated by $D_f$, that is, $U = V \setminus (D_f \cup P_f \cup P'_f \cup (N(D_f) \cap Z))$. We will show that when no rule can be applied, $U$ is empty, that is, all vertices are decided or dominated by $D_f$. In the third step of our algorithm we will show how to complete the solution in polynomial time when $U$ is empty. Since our Rules do not modify the graph, we will describe the sub-instances we branch on by specifying the tuple $(D_f, P_f, P'_f)$.

To ease notation, let $U_C = U \cap C$ and $U_Z = U \cap Z$ (see Figure 2.1). For $u \in V$, we use $d_{U_C}(u)$ and $d_{U_Z}(u)$ to denote the size of the sets $N(u) \cap U_C = N_{U_C}(u)$ and $N(u) \cap U_Z = N_{U_Z}(u)$, respectively.

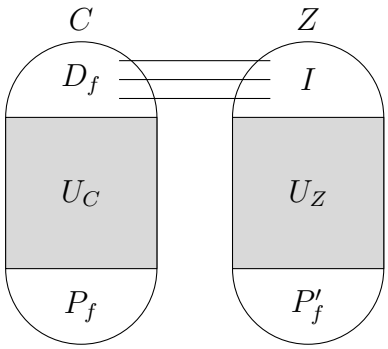


Figure 2.1: Partition of $V = C \cup Z$, of $C = D_f \cup P_f \cup U_C$, and of $Z = P'_f \cup I \cup U_Z$ during the process of the algorithm, where $I = N(D_f) \cap Z$. The only edges drawn show that $I$ is dominated by $D_f$.

We will present each Rule individually and directly after explain why it

is correct and its associated running-time, to ease presentation while having a consistent analysis.

**Reduction Rule (R1):** If there exists $u \in U_C$ such that $d_{U_Z}(u) \leq 1$, then put $u$ in $P_f$, that is, recurse on the instance $(D_f, P_f \cup \{u\}, P'_f)$.

- Observe that no neighbor of $u$ in $U_C$ can be a private neighbor of $u$ since $U_C \subseteq C \subseteq D \cup P$, and because $d_{U_Z}(u) \leq 1$ the vertex $u$ can have at most one private neighbor, so it must be the case that $u \in P$.

**Reduction Rule (R2):** If there exists $v \in U_Z$ such that $d_{U_C}(v) = 0$, then put $v$ in $P'_f$, that is, recurse on the instance $(D_f, P_f, P'_f \cup \{v\})$.

- The vertex $v$ must be dominated, but it has no neighbor in $U_C$, so it must be the case that $v \in P$.

Now that we have presented the two Reduction Rules which first apply in our algorithm, we will describe the Branching Rules. Thus, we need first to define our measure of progress. We define it to be the size of the set $\{u \in U_C \mid d_{U_Z}(u) \geq 2\} \cup \{u \in U_Z \mid d_{U_C}(u) \geq 1\}$. In other words, we count the undecided vertices of $U_C$ that have at least two undecided, non-dominated vertices in $U_Z$, and the undecided, non-dominated vertices of $U_Z$ that have at least one undecided neighbor in $C$. This is motivated by the fact that undecided vertices that do not respect these degree bounds are eliminated by the Reduction Rules and hence do not affect the running time. Let $l$ denote the number of the vertices that we counted according to this measure. Clearly, $l \leq n$. Let $T(l)$ be the maximum number of branches produced for an instance where the measure has value $l$. We now consider each Branching Rule individually:

**Branching Rule (B1):** If there exists $u \in U_C$ such that $d_{U_Z}(u) \geq 4$, then branch on the following two sub-instances: $(D_f \cup \{u\}, P_f, P'_f)$ and $(D_f, P_f \cup \{u\}, P'_f)$.

- Branching Rule B1 is correct from $U_C \subseteq C \subseteq D \cup P$.
- We have $T(l) \leq T(l-1) + T(l-5)$, since in the branch where $u \in D_f$ at least 4 vertices of $U_Z$ become dominated.
- $T(l) \leq T(l-1) + T(l-5)$ gives $x^5 = x^4 + 1$ with root $r < 1.3248$.

Note that we may now assume that all vertices of $U_C$ have $d_{U_Z} \in \{2, 3\}$. The following two Rules eliminate vertices $u \in U_C$ with $d_{U_Z}(u) = 2$.

**Branching Rule (B2.1):** If there exist $u_1, u_2 \in U_C$ such that $d_{U_Z}(u_1) = 3$, $d_{U_Z}(u_2) = 2$, and $N_{U_Z}(u_1) \cap N_{U_Z}(u_2) \neq \emptyset$ then branch on the following instances: $(D_f \cup \{u_1\}, P_f \cup \{u_2\}, P'_f)$ and $(D_f, P_f \cup \{u_1\}, P'_f)$.

- Branching Rule B2.1 is correct because if $u_1 \in D$, then $u_2$ cannot have two private neighbors and it is forced to be in $P$.
- We have $T(l) \leq T(l-1)+T(l-5)$, since in the branch where $u_1 \in D_f$ we also set $u_2 \in P_f$ and 3 vertices of $U_Z$ become dominated.
- $T(l) \leq T(l-1)+T(l-5)$ gives $x^5 = x^4 + 1$ with root $r < 1.3248$.

**Branching Rule (B2.2):** If there exists $u \in U_C$ with $d_{U_Z}(u) = 2$ we branch on the instances $(D_f \cup \{u\}, P_f, P'_f)$ and $(D_f, P_f \cup \{u\}, P'_f)$.

- Branching Rule B2.2 is correct again from $U_C \subseteq C \subseteq D \cup P$.
- Let $N(u) \cap U_Z = \{v_1, v_2\}$. Note that if $d_{U_C}(v_1) \geq 2$, then all vertices $u' \in U_C$ adjacent to $v_1$ must have $d_{U_Z}(u') = 2$. This is because Rules R1, B1 and B2.1 do not apply. Let $s$ be the number of vertices of $\{v_1, v_2\}$ which have at least two neighbors in $U_C$. We consider the following cases:
  - If $s = 0$, then $T(l) \leq T(l-3) + T(l-3)$, because when $u \in P_f$, $v_1, v_2$ no longer contribute to $l$ (they have no other neighbor in $U_C$).
  - If $s = 1$, then $T(l) \leq T(l-4)+T(l-2)$. To see this, let $u' \in U_C$ be a neighbor of $\{v_1, v_2\}$. As we said, $d_{U_Z}(u') = 2$, so setting $u \in D_f$ will activate Rule R1 on $u'$, decreasing $l$ by 4. On the other hand, if $u \in P_f$, then one of $\{v_1, v_2\}$ is deleted by Rule R2.
  - If $s = 2$ and $|(N(v_1) \cup N(v_2)) \cap U_C| = 2$, then we must have $N(v_1) \cap U_C = N(v_2) \cap U_C = \{u, u'\}$ for some $u' \in U_C$ with $d_{U_Z}(u') = 2$. In this case Rule B2.2 (and Rules R1, R2) will be applied successively to $u, u'$ giving $T(l) \leq 3T(l-4)$.
  - If none of the above applies, then $s = 2$ and we have $T(l) \leq T(l-5) + T(l-1)$, because when $u \in D_f$ we force at least two other vertices of $U_C$ into $P_f$.
- 
  - $T(l) \leq 2T(l-3)$ gives $x^3 = 2$ with root $r < 1.2600$.
  - $T(l) \leq T(l-4)+T(l-2)$ gives $x^4 = x^2 + 1$ with root $r < 1.2721$.
  - $T(l) \leq 3T(l-4)$ gives $x^4 = 3$ with root $r < 1.3161$.
  - $T(l) \leq T(l-5)+T(l-1)$ gives $x^5 = x^4 + 1$ with root $r < 1.3248$.

We now have that all vertices $u \in U_C$ have $d_{U_Z}(u) = 3$. Let us now branch on vertices of $U_Z$ to ensure that these also do not have too low degree.

**Branching Rule (B3.1):** If there exists $v \in U_Z$ with $d_{U_C}(v) = 1$ let $N_{U_C}(v) = \{u\}$. We branch on the instances $(D_f \cup \{u\}, P_f, P'_f)$ and $(D_f, P_f \cup \{u\}, P'_f)$.

- Branching Rule B3.1 is correct again from $U_C \subseteq C \subseteq D \cup P$.
- We have $T(l) \leq T(l-4) + T(l-2)$, since when $u \in P_f$ we apply Rule R2.
- $T(l) \leq T(l-4) + T(l-2)$ gives $x^4 = x^2 + 1$ with root $r < 1.2721$.

**Branching Rule (B3.2):** If there exists $v \in U_Z$ with $d_{U_C}(v) = 2$ let $N_{U_C}(v) = \{u_1, u_2\}$. We branch on the instances $(D_f \cup \{u_1\}, P_f, P'_f)$, $(D_f \cup \{u_2\}, P_f \cup \{u_1\}, P'_f)$, and $(D_f, P_f \cup \{u_1, u_2\}, P'_f)$.

- Branching rule B3.2 is correct since we have the three following cases: $u_1 \in D$; or $u_1 \in P$ and $u_2 \in D$; or $u_1$ and $u_2 \in P$.
- We have $T(l) \leq T(l-3) + T(l-4) + T(l-5)$ using the fact that $d_{U_Z}(u_1) = d_{U_Z}(u_2) = 3$ and the fact that Rule R2 is applied when $u_1, u_2 \in P_f$.
- $T(l) \leq T(l-3) + T(l-4) + T(l-5)$ gives $x^5 = x^2 + x + 1$ with root $r < 1.3248$.

If we cannot apply any of the above Rules, for all $u \in U_C$ we have $d_{U_Z}(u) = 3$ and for all $v \in U_Z$ we have $d_{U_C}(v) \geq 3$. We now consider three remaining cases: (i) two vertices of $U_C$ have two common neighbors in $U_Z$ (ii) there exists a vertex $v \in U_Z$ with $d_{U_C}(v) = 3$ (iii) everything else.

**Branching Rule (B4):** If there exist $u_1, u_2 \in U_C$ and $v_1, v_2 \in U_Z$ with $(u_i, v_j) \in E$ for all $i, j \in \{1, 2\}$, then we branch on the instances $(D_f \cup \{u_1\}, P_f \cup \{u_2\}, P'_f)$ and $(D_f, P_f \cup \{u_1\}, P'_f)$.

- Branching Rule B4 is correct because if $u_1 \in D$, then $u_2$ cannot have two private neighbors since $d_{U_Z}(u_2) = 3$.
- We have $T(l) \leq T(l-1) + T(l-5)$.
- $T(l) \leq T(l-5) + T(l-1)$ gives $x^5 = x^4 + 1$ with root $r < 1.3248$.

**Branching Rule (B5):** If there exists $v \in U_Z$ with $d_{U_C}(v) = 3$, let $N_{U_C}(v) = \{u_1, u_2, u_3\}$ and for $i \in \{1, 2, 3\}$ let $X_i = \{w \in U_C \setminus \{u_1, u_2, u_3\}, N(w) \cap N(u_i) \cap (U_Z \setminus \{v\}) \neq \emptyset\}$, that is, $X_i$ is the set of vertices of $U_C$ that share a neighbor with $u_i$ in $U_Z$ other than $v$. Then we branch on the following 8 instances: (i) the instance $(D_f, P_f \cup \{u_1, u_2, u_3\}, P'_f \cup \{v\})$ (ii) for $i \in \{1, 2, 3\}$, we produce the instances $(D_f \cup \{u_i\}, P_f \cup (\{u_1, u_2, u_3\} \setminus \{u_i\}), P'_f)$ (iii) for $i, j \in \{1, 2, 3\}$, with $i < j$ we produce the instances $(D_f \cup \{u_i, u_j\}, P_f \cup (\{u_1, u_2, u_3\} \setminus \{u_i, u_j\}) \cup X_i \cup X_j, P'_f)$ (iv) we produce the instance $(D_f \cup \{u_1, u_2, u_3\}, P_f \cup X_1 \cup X_2 \cup X_3, P'_f)$.

- Branching Rule B5 is correct since we have the following cases: (i) all vertices $u_1, u_2$ and $u_3$ are in $P$ (ii) or exactly one of them is in $D$ (iii) or exactly two of them are in $D$ (iv) or all of them are in $D$. Note first that $u_1$, $u_2$ and $u_3$ only share $v$ as neighbor in $U_Z$ since Branching Rule B4 is not triggered. Branching Rule B5 is correct by the following arguments:

  (i) $v$ must be dominated so it must be the case that $v \in P$;

  (ii) The two vertices not in $D$ necessarily are in $P$;

  (iii) Since $u_i$ and $u_j$ share $v$ as common neighbor and both have exactly three neighbors in $U_Z$, the vertices of $X_i$ and $X_j$ have to be in $P$ because otherwise $u_i$ and $u_j$ do not have two private neighbors;

  (iv) For the same reason, the vertices of $X_1$, $X_2$ and $X_3$ have to be in $P$.

- We have $T(l) \leq T(l-4) + 3T(l-6) + 3T(l-12) + T(l-14)$. Indeed we have: (i) the branch where $u_1, u_2, u_3 \in P_f$, which also effectively eliminates $v$ (ii) the branch where $u_1 \in D_f$ and $u_2, u_3 \in P_f$, which also dominates $N(u_1) \cap U_Z$ (plus two more symmetric branches) (iii) the branch where $u_1, u_2 \in D_f$ and $u_3 \in P_f$ (plus two more symmetric branches). Here we first observe that $\{v, u_1, u_2, u_3\} \cup ((N(u_1) \cup N(u_2)) \cap U_Z)$ contains exactly 8 distinct vertices, because $d_{U_Z}(u_1) = d_{U_Z}(u_2) = 3$, while $N(u_1) \cap U_Z$ and $N(u_2) \cap U_Z$ share exactly one common element $(v)$, since Rule B4 does not apply. In addition to eliminating these 8 vertices, this branch also eliminates $X_1 \cup X_2$. We argue that $X_1$ alone contains at least 4 additional vertices, distinct from the 8 eliminated vertices. Let $N(u_1) \cap U_Z = \{v, w_1, w_2\}$. We know that $d_{U_C}(w_1), d_{U_C}(w_2) \geq 3$, since Rule B3.2 did not apply. Furthermore, since $w_1, w_2$ share $u_1$ as a common neighbor in $U_C$, they cannot share another, as Rule B4 would apply. In addition, neither $w_1$ nor $w_2$ can be connected to $u_2$ or $u_3$,

since together with $v, u_1$ this would active Rule B4. Hence, we eliminate at least 12 vertices for each of these three branches. Finally, the case (iv) where $u_1, u_2, u_3 \in D_f$ is similar, except we also eliminate two additional neighbors of $u_3$ in $U_Z$ which now become dominated.

- $T(l) \leq T(l-4) + 3T(l-6) + 3T(l-12) + T(l-14)$ gives $x^{14} = x^{10} + 3x^8 + 3x^2 + 1$ with root $r < 1.3252$.

**Branching Rule (B6):** Consider $u \in U_C$ and let $N_{U_Z}(u) = \{v_1, v_2, v_3\}$. We branch on the following instances: $(D_f, P_f \cup \{u\}, P'_f)$, $(D_f \cup \{u\}, P_f \cup (N_{U_C}(v_1) \setminus \{u\}), P'_f)$, and $(D_f \cup \{u\}, P_f \cup ((N_{U_C}(v_2) \cup N_{U_C}(v_3)) \setminus \{u\}), P'_f)$.

- Branching Rule B6 is correct because if $u \in D$, then either $v_1$ is one of its private neighbors, or both $v_2$ and $v_3$ are its private neighbors.
- We have $T(l) \leq T(l-1)+T(l-7)+T(l-10)$. Here we use the fact that since Rule B5 does not apply, $d_{U_C}(v_i) \geq 4$ and also that since Rule B4 does not apply, $N(v_i) \cap N(v_j) \cap U_C = \{u\}$ for all $i, j \in \{1, 2, 3\}$. Hence, the branch where $u \in D_f$ and $v_1$ is a private neighbor of $u$ forces three more vertices of $U_C$ into $P_f$, and the branch where $v_2, v_3$ are private neighbors of $u$ forces six more vertices of $U_C$ into $P_f$.
- $T(l) \leq T(l-1) + T(l-7) + T(l-10)$ gives $x^{10} = x^9 + x^3 + 1$ with root $r < 1.3001$.

Our algorithm applies the above Rules in order as long as possible. Since we have proved the correctness of our Rules individually, we can explain what happens when no Rule is applicable. But first, let us establish a useful property.

**Lemma 2.5.** *If none of the Rules can be applied then $U = \emptyset$.*

*Proof.* Observe that, by applying rules R1, B1, B2.2, B6, we eventually eliminate all vertices of $U_C$, since these rules alone cover all the cases for $d_{U_Z}(u)$ for any $u \in U_C$. So, if none of these rules applies, $U_C$ is empty. But then applying R2 will also eliminate $U_Z$, which makes all of $U$ empty. □

**Step 3:** When $U$ is empty, reduce the problem to Min Edge Cover. We now show how to complete the solution in polynomial time.

**Lemma 2.6.** *Let $(D_f, P_f, P'_f)$ be a good tuple such that no Rule can be applied. Then it is possible to construct in polynomial time a mixed dominating set of size at most $|D| + |M|$.*

*Proof.* Because no Rule can be applied, by Lemma 2.5, we have that $U = V \setminus (D_f \cup P_f \cup P'_f \cup (N(D_f) \setminus C)) = \emptyset$.

Let $M'$ be a minimum edge cover of $G[P_f \cup P'_f]$. Then, we claim that $|D| + |M| \geq |D_f| + |M'|$. First, $|D| \geq |D_f|$ because $D_f \subseteq D$. We now claim that $|M| \geq |M'|$. Note that $P_f \subseteq P \cap C$ and $P'_f \subseteq P \cap Z$, so $P_f \cup P'_f \subseteq P$. $M$ is an edge cover of $G[P]$, and $M'$ is a minimum edge cover of $G[P_f \cup P'_f]$, with $P_f \cup P'_f \subseteq P$, so necessarily $|M| \geq |M'|$.

Since Min Edge Cover is in **P**, we have constructed in polynomial time a mixed dominating set of size at most $|D| + |M|$. □

We can now prove the main result of this section :

**Theorem 2.7.** *Min Mixed Dominating Set can be solved in time* $O^*(1.912^n)$ *and polynomial space.*

*Proof.* The algorithm first enumerates all minimal vertex covers $C$, then applies all Rules exhaustively, and then for each branch invokes Lemma 2.6. At the end we output the best solution found.

By Lemmas 2.3 and 2.4 we obtain (assuming we have already taken isolated vertices) that there exists an optimal nice mds partition $V = D \cup P \cup I$ and a minimal vertex cover with $D \subseteq C \subseteq D \cup P$, so consider the execution of the algorithm on $C$. We have proven that one of the branches will end up with a good tuple, and by Lemma 2.5 when we can no longer apply any Rules, $U$ is empty, so we correctly solve the resulting instance in polynomial time by Lemma 2.6. Hence, the algorithm produces a correct solution.

Let us now analyze the running time. First, enumerating all minimal vertex cover takes time at most $O^*(3^{n/3})$, which is also an upper bound on the number of such covers by a result of Moon and Moser [MM65]. Moreover, we observe that we can decide if a Rule applies in polynomial time, and the algorithm of Lemma 2.6 runs in polynomial time. We therefore only need to bound the number of sub-instances the branching step will produce, as a function of $n$.

Of all the branching vectors, the worst case is given by Branching Rule B5, which leads to a complexity of $1.3252^l$. Taking into account the cost of enumerating all minimal vertex covers and the fact that $l \leq n$, the running time of our algorithm is $O^*(3^{n/3} \cdot 1.3252^n) = O^*(1.912^n)$. □

Now that we have presented our improved exact algorithm, we describe in the next section our improved FPT algorithm parameterized by the natural parameter $k$, which is also a branching algorithm using the implicit private structure of Min Mixed Dominating Set.

## 2.3 Improved FPT Algorithm

In this section we present our improved FPT algorithm parameterized by the solution size $k$ running in time $O^*(3.510^k)$. Let us give an overview of our algorithm. Consider an instance $(G = (V, E), k)$ of the Mixed Dominating Set problem parameterized by $k$, and fix, for the sake of the analysis, a solution of size $k$ which is a nice mixed dominating set. If a solution of size $k$ exists, then a nice solution must exist by Lemma 2.3 (assuming without loss of generality that $G$ has no isolated vertices), so suppose it gives the nice mds partition $V = D \cup P \cup I$. Note that for such a solution $D \cup M$ of size $k$, we have $k \geq |D| + |P|/2$ since $|M| \geq |P|/2$.

Our algorithm begins by performing a branching step trying to guess a part of this partition. In particular, we gradually build up two disjoint sets $D_f, P_f$ which store the vertices that must belong to $D$ and $P$ respectively. Let $U = V \setminus (D_f \cup P_f)$ be the set of "undecided" vertices and, furthermore, let $U^* = U \setminus N(D_f)$ be the set of undecided vertices which are not currently dominated by the solution. Our algorithm proceeds in the following steps: (i) first, we branch with the goal of eliminating $U^*$, that is, with the goal of finding a partial solution that dominates all vertices (ii) then, because the considered solution is nice, we observe that we cannot place any more vertices in $D_f$; we therefore perform a simple "vertex cover"-type branching in $G[U]$, until we arrive at a situation where the maximum degree of $G[U]$ is 1 (iii) then, we invoke a result of Xiao and Sheng [XS19] to complete the solution in polynomial time.

**Step 1:** Branch to eliminate $U^*$.

Recall that we have fixed for the sake of the analysis an optimal nice mds partition $V = D \cup P \cup I$. It will be convenient to describe a recursive algorithm which is given with two disjoint sets of vertices $D_f, P_f$. We will say that the sets $(D_f, P_f)$ are *good* if $D_f \subseteq D$ and $P_f \subseteq P$. Clearly, these conditions are satisfied if $D_f = P_f = \emptyset$. We will describe a series of Rules, which must be applied exhaustively, always selecting the first Rule that can be applied. For correctness, we will show that for each Branching Rule, if the current instance is characterized by a good pair $(D_f, P_f)$, at least one of the produced instances is also good. When no Rule can be applied, we will proceed to the next step.

Recall that we denote $U = V \setminus (D_f \cup P_f)$ and $U^* = U \setminus N(D_f)$. Our strategy will be to branch in a way that eliminates $U^*$ as quickly as possible because, as we will see in the next step, once this is done the problem becomes much easier. We begin by branching on low-degree vertices, which will allow us to assume that all remaining vertices are high-degree as we consider later Rules, and which is sped up by the implicit private structure we have obtained

by Lemma 2.3. Here, for a vertex $u \in U^*$ we are mostly interested in its degrees in $G[U]$ and in $G[U^*]$. Note that $d_{U^*}(u) \leq d_U(u)$ since $U^* \subseteq U$.

We will present each Rule individually and directly after explain why it is correct and its associated running-time. Recall for the sake of the analysis that an instance is good if $D_f \subseteq D$ and $P_f \subseteq P$.

**Sanity Check Rule:** If $|D_f| + \frac{|P_f|}{2} > k$, reject. If there exists $u \in D_f$ such that $|N_U(u) \setminus N(D_f \setminus \{u\})| \leq 1$, reject.

- The Sanity Check Rule will reject if either the currently marked vertices in $(D_f, P_f)$ have total cost more than $k$ (which implies that this is not a good instance, as the correct partition has cost at most $k$ if one such solution exists); or if a vertex $u \in D_f$ has at most one private neighbor in $U$. Since the number of private neighbors of $u$ can only diminish if we add vertices to $D_f$, if $u \in D$ this would contradict the niceness of the partition $D \cup P \cup I$. Hence, in this case also the current instance is not good.

**Reduction Rule (R1):** If there exists $u \in U^*$ with $d_U(u) = 0$, then put $u$ in $P_f$, that is, recurse on the instance $(D_f, P_f \cup \{u\})$.

- If the current instance is good, then $u \notin D$ (because it would not have two private neighbors) and $u \notin I$ (because it would not be dominated). Hence, the new instance is also good.

Now that we have presented our Sanity Check Rule and our only Reduction Rule which first apply in our algorithm, we will describe the Branching Rules. Thus, we need first to define our measure of progress. We define it to be $l = 2k - 2|D_f| - |P_f|$. Initially, $l = 2k$, and we observe that because of the Sanity Check Rule in all produced instances we have $l \geq 0$. We will therefore upper bound the number of produced instances by measuring how much each Branching Rule decreases $l$. Let $T(l)$ be the maximum number of branches produced for an instance where measure has value $l$. We now consider each Branching Rule individually:

**Branching Rule (B1):** If there exists $u \in U^*$ with $d_U(u) = 1$, then let $N_U(u) = \{v\}$. Branch on the following two sub-instances: $(D_f, P_f \cup \{u\})$ and $(D_f \cup \{v\}, P_f)$.

- We note that $u \notin D$, because it would not have two private neighbors. If $u \in I$, then $v \in D$, because $u$ must be dominated. Hence, one of the branches is good.

- We have $T(l) \leq T(l-1) + T(l-2)$.
- $T(l) \leq T(l-1) + T(l-2)$ gives $x^2 = x + 1$ with root $r < 1.6181$.

We are now at a situation where all vertices $u \in U^*$ have $d_U(u) \geq 2$.

**Branching Rule (B2.1):** If there exists $u \in U^*$ with $d_U(u) = 2$ and $d_{U^*}(u) \in \{0, 1\}$, then let $N_U(u) = \{v_1, v_2\}$. Branch on the sub-instances: $(D_f, P_f \cup \{u\})$, $(D_f \cup \{v_1\}, P_f \cup \{v_2\})$, $(D_f \cup \{v_2\}, P_f \cup \{v_1\})$ and $(D_f \cup \{v_1, v_2\}, P_f)$.

- We again have $u \notin D$, because it would not have two private neighbors. If $u \in I$, then $\{v_1, v_2\} \subseteq D \cup P$ and we consider all such possibilities, except $v_1, v_2 \in P$, because $u$ must be dominated.
- We have $T(l) \leq T(l-1) + 2T(l-3) + T(l-4)$.
- $T(l) \leq T(l-1) + 2T(l-3) + T(l-4)$ gives $x^4 = x^3 + 2x + 1$ with root $r < 1.7944$.

Before presenting Branching Rule 2.2, we make a simple observation we will use several times in the rest of the algorithm: for a vertex $u \in U^*$ with two neighbors $v_1, v_2 \in U$, if we put $u$ in $D_f$ with $v_1$ and $v_2$ its private neighbors, then we must put $(N_U(v_1) \cup N_U(v_2)) \setminus \{u\}$ in $P_f$ in order to have $v_1$ and $v_2$ the private neighbors of $u$.

Let us also introduce another helpful definition. For $v_1, v_2 \in U$, we will say that $\{v_1, v_2\}$ is a *feasible pair* if $|(N(v_1) \cap U^*) \setminus N(v_2)| \geq 2$ and $|(N(v_2) \cap U^*) \setminus N(v_1)| \geq 2$. Informally, $\{v_1, v_2\}$ is feasible if it is possible that both $v_1, v_2 \in D$. In other words, $\{v_1, v_2\}$ is *not* feasible, if placing both vertices in $D_f$ would immediately activate the Sanity Check rule because one of the two vertices would not have enough private neighbors.

**Branching Rule (B2.2):** If there exists $u \in U^*$ with $d_U(u) = 2$, then let $N_U(u) = \{v_1, v_2\}$. Branch on the following sub-instances: $(D_f \cup \{u\}, P_f \cup ((N_U(v_1) \cup N_U(v_2)) \setminus \{u\}))$, $(D_f, P_f \cup \{u\})$, $(D_f \cup \{v_1\}, P_f \cup \{v_2\})$, $(D_f \cup \{v_2\}, P_f \cup \{v_1\})$ and $(D_f \cup \{v_1, v_2\}, P_f)$.

- We have the same cases as before, but now it is possible that $u \in D$. However, in this case $v_1, v_2$ must be private neighbors of $u$, hence $(N_U(v_1) \cup N_U(v_2)) \setminus \{u\}$ must be a subset of $P$.
- Since B2.1 does not apply, we have $v_1, v_2 \in U^*$, so $d_U(v_1), d_U(v_2) \geq 2$. We consider the following cases:

  - If $\{v_1, v_2\}$ is not a feasible pair, that is (without loss of generality) we have $|N_{U^*}(v_1) \setminus N(v_2)| \leq 1$. First, we note that $d_U(v_1) \geq 2$, since Rule B1 did not apply, so $|N_U(v_1) \setminus \{u\}| \geq 1$. Also, the Sanity Check Rule is activated for the instance where $v_1, v_2 \in D_f$. Taking into account the remaining instances we have: $T(l) \leq T(l-1) + 3T(l-3)$.
  - If $\{v_1, v_2\}$ is a feasible pair then $|(N_{U^*}(v_1) \cup N_{U^*}(v_2)) \setminus \{u\}| \geq 4$, because each of $v_1, v_2$ has two non-dominated neighbors which are not neighbors of the other. We have $T(l) \leq T(l-1) + 2T(l-3) + T(l-4) + T(l-6)$.

- - $T(l) \leq T(l-1) + 3T(l-3)$ gives $x^3 = x^2 + 3$ with root $r < 1.8638$.
  - $T(l) \leq T(l-1) + 2T(l-3) + T(l-4) + T(l-6)$ gives $x^6 = x^5 + 2x^3 + x^2 + 1$ with root $r < 1.8199$.

We are now at a situation where all vertices $u \in U^*$ have $d_U(u) \geq 3$.

**Branching Rule (B3.1):** If there exists $u \in U^*$ with $d_U(u) = 3$ and $d_{U^*}(u) \leq 2$, then let $N_U(u) = \{v_1, v_2, v_3\}$ and let $v_3 \in U \setminus U^*$. We branch on $(D_f, P_f \cup \{u\})$, $(D_f \cup \{u\}, P_f \cup ((N_U(v_1) \cup N_U(v_2)) \setminus \{u\}))$, and, for each non-empty $S \subseteq \{v_1, v_2, v_3\}$, we branch on $(D_f \cup S, P_f \cup (\{v_1, v_2, v_3\} \setminus S))$.

- We observe that if $u \in D$, then $v_1, v_2$ must be its private neighbors, so again $(N_U(v_1) \cup N_U(v_2)) \setminus \{u\}$ must be a subset of $P$. If $u \in I$, we consider all partitions of $N_U(u)$ into $D$ and $P$, while ensuring that $u$ is dominated.

- First note that, if $v_i \in U^*$, then $d_U(v_i) \geq 3$, since Rules B1-B2.2 do not apply, hence $|N_U(v_i) \setminus \{u\}| \geq 2$. Consider the following subcases:

  - $d_{U^*}(u) \leq 1$. Then the branch where $u \in D_f$ is immediately eliminated by the Sanity Check Rule. For the remaining branches we have $T(l) \leq T(l-1) + 3T(l-4) + 3T(l-5) + T(l-6)$.
  - $d_{U^*}(u) = 2$, so $N_{U^*}(u) = \{v_1, v_2\}$, and $\{v_1, v_2\}$ is a feasible pair. Then $|(N_U(v_1) \cup N_U(v_2)) \setminus \{u\}| \geq 4$. We have $T(l) \leq T(l-1) + 3T(l-4) + 3T(l-5) + 2T(l-6)$
  - $d_{U^*}(u) = 2$, so $N_{U^*}(u) = \{v_1, v_2\}$, and $\{v_1, v_2\}$ is not a feasible pair. Then the Sanity Check Rule eliminates the instances where $\{v_1, v_2\} \subseteq D$. Taking into account the remaining instances we have $T(l) \leq T(l-1) + 4T(l-4) + 2T(l-5)$.

- $T(l) \leq T(l-1) + 3T(l-4) + 3T(l-5) + T(l-6)$ gives $x^6 = x^5 + 3x^2 + 3x + 1$ with root $r < 1.8205$.
- $T(l) \leq T(l-1) + 3T(l-4) + 3T(l-5) + 2T(l-6)$ gives $x^6 = x^5 + 3x^2 + 3x + 2$ with root $r < 1.8393$.
- $T(l) \leq T(l-1) + 4T(l-4) + 2T(l-5)$ gives $x^5 = x^4 + 4x + 2$ with root $r < 1.8305$.

**Branching Rule (B3.2):** If there exists $u \in U^*$ with $d_U(u) = 3$ such that there exist at least two feasible pairs in $N_U(u)$, then we do the following. Let $N_U(u) = \{v_1, v_2, v_3\}$. For each $i, j \in \{1, 2, 3\}$, with $i < j$, branch on $(D_f \cup \{u\}, P_f \cup ((N_U(v_i) \cup N_U(v_j)) \setminus \{u\}))$. Furthermore, branch on the instances $(D_f, P_f \cup \{u\})$ and, for each non-empty $S \subseteq \{v_1, v_2, v_3\}$, on the instance $(D_f \cup S, P_f \cup (\{v_1, v_2, v_3\} \setminus S))$.

- We branch in a similar fashion as in Branching 3.1, except that, for all $i, j \in \{1, 2, 3\}$ with $i < j$, we consider the case that $v_i, v_j$ are private neighbors of $u$, when $u \in D$.
- We have $v_1, v_2, v_3 \in U^*$. We again note that if $\{v_i, v_j\}$ is feasible then $|(N_U(v_i) \cup N_U(v_j)) \setminus \{u\}| \geq 4$. Therefore, a branch corresponding to a feasible pair diminishes $l$ by at least 6. We consider the subcases:
  - All three pairs from $\{v_1, v_2, v_3\}$ are feasible. Then we get $T(l) \leq T(l-1) + 3T(l-4) + 3T(l-5) + 4T(l-6)$.
  - Two of the pairs from $\{v_1, v_2, v_3\}$ are feasible, and one, say $\{v_1, v_2\}$ is not feasible. Then the Sanity Check rule will eliminate the branches that have $\{v_1, v_2\} \subseteq D$. We therefore get $T(l) \leq T(l-1) + 4T(l-4) + 2T(l-5) + 2T(l-6)$.
- $T(l) \leq T(l-1) + 3T(l-4) + 3T(l-5) + 4T(l-6)$ gives $x^6 = x^5 + 3x^2 + 3x + 4$ with root $r < 1.8734$.
- $T(l) \leq T(l-1) + 4T(l-4) + 2T(l-5) + 2T(l-6)$ gives $x^6 = x^5 + 4x^2 + 2x + 2$ with root $r < 1.8672$.

**Branching Rule (B3.3):** If there exists $u \in U^*$ with $d_U(u) = 3$, let $N_U(u) = \{v_1, v_2, v_3\}$ and branch on the following instances: $(D_f, P_f \cup \{u\})$, $(D_f \cup \{u\}, P_f \cup (N_U(v_1) \setminus \{u\}))$, $(D_f \cup \{u\}, P_f \cup ((N_U(v_2) \cup N_U(v_3)) \setminus \{u\}))$, and for each non-empty $S \subseteq \{v_1, v_2, v_3\}$ branch on $(D_f \cup S, P_f \cup (\{v_1, v_2, v_3\} \setminus S))$.

- We make a variation of the previous branching by arguing that, if $u \in D$, then either $v_1$ is its private neighbor, or both $v_2, v_3$ are its private neighbors.

- Now at least two pairs in $\{v_1, v_2, v_3\}$ are not feasible (otherwise we would have applied B3.2). Then, the Sanity Check Rule eliminates all branches where $D$ contains an infeasible pair. From the remaining branches we get $T(l) \leq T(l-1) + 5T(l-4) + T(l-5)$.

- $T(l) \leq T(l-1) + 5T(l-4) + T(l-5)$ gives $x^5 = x^4 + 5x + 1$ with root $r < 1.8603$.

We are now at a situation where all vertices $u \in U^*$ have $d_U(u) \geq 4$. The next case we would like to handle is that of a vertex $u \in U^*$ with $d_U(u) = d_{U^*}(u) = 4$. For such a vertex let $N_U(u) = \{v_1, v_2, v_3, v_4\}$. Let us now give one more helpful definition. For some $i \in \{1, 2, 3, 4\}$, we will say that $v_i$ is *compatible for* $u$ if $N_{U^*}(v_i)$ contains at least two vertices which are not neighbors of any $v_j$, for $j \in \{1, 2, 3, 4\} \setminus \{i\}$. In other words, $v_i$ is compatible if it has two private neighbors, which will remain private even if we put all of $\{v_1, v_2, v_3, v_4\}$ in $D_f$. Using this definition, we distinguish the following two cases:

**Branching Rule (B4.1):** If there exists $u \in U^*$ with $d_U(u) = d_{U^*}(u) = 4$, $N_U(u) = \{v_1, v_2, v_3, v_4\}$, and all $v_i$ are compatible for $u$, then we branch on the following instances: $(D_f, P_f \cup \{u\})$; for each $i, j \in \{1, 2, 3, 4\}$ with $i < j$ we branch on $(D_f \cup \{u\}, P_f \cup ((N_U(v_i) \cup N_U(v_j)) \setminus \{u\}))$; for each non-empty subset $S \subseteq N_U(u)$, let $S^c = N_U(u) \setminus S$, branch on $(D_f \cup S, P_f \cup S^c)$.

- The branching is similar to B3.2: if $u \in D$, then two of its neighbors must be private and we consider all possibilities.

- If all $v_i \in U^*$ and all $v_i$ are compatible for $u$ that means that for all $i, j \in \{1, 2, 3, 4\}$, with $i < j$, we have $|(N_U(v_i) \cup N_U(v_j)) \setminus \{u\}| \geq 5$, where we use that $v_i$ has at least two neighbors in $U$ which are not connected to $v_j$ (and vice-versa) and that $d_U(v_i), d_U(v_j) \geq 4$ since previous Rules do not apply. We therefore have $T(l) \leq T(l-1) + 4T(l-5) + 6T(l-6) + 10T(l-7) + T(l-8)$.

- $T(l) \leq T(l-1) + 4T(l-5) + 6T(l-6) + 10T(l-7) + T(l-8)$ gives $x^8 = x^7 + 4x^3 + 6x^2 + 10x + 1$ with root $r < 1.8595$.

If the previous rule does not apply, vertices $u \in U^*$ with $d_U(u) = 4$ have either $d_{U^*}(u) \leq 3$ or a neighbor $v_i \in N_U(u)$ is not compatible for $u$.

**Branching Rule (B4.2):** If there exists $u \in U^*$ with $d_U(u) = 4$, then let $N_U(u) = \{v_1, v_2, v_3, v_4\}$. Suppose without loss of generality that for all $1 \leq j \leq d_{U^*}(u)$ we have $v_j \in U^*$ (that is, vertices of $N_{U^*}(u)$ are ordered first),

and that, if there exists a feasible pair in $N_{U^*}(u)$, then $\{v_1, v_2\}$ is feasible. We produce the instances: $(D_f, P_f \cup \{u\})$; for each non-empty subset $S \subseteq N_U(u)$, let $S^c = N_U(u) \setminus S$, we branch on $(D_f \cup S, P_f \cup S^c)$; if $d_{U^*}(u) \geq 2$ we produce the branch $(D_f \cup \{u\}, P_f \cup ((N_U(v_1) \cup N_U(v_2)) \setminus \{u\}))$; for $3 \leq j \leq d_{U^*}(u)$ we produce the branch $(D_f \cup \{u\}, P_f \cup (N_U(v_j) \setminus \{u\}))$.

- Either $u \in P$ (which we consider), or $u \in I$, so we consider all partitions of $N_U(u)$ into $D, P$ that dominate $u$, or $u \in D$. For the latter to happen it must be the case that $d_{U^*}(u) \geq 2$. In that case, either $v_1, v_2$ are both private neighbors, or $v_3$ is a private neighbor (if $v_3 \in U^*$), or $v_4$ is a private neighbor (if $v_4 \in U^*$).

- – We first handle the case where $d_{U^*}(u) \leq 3$. Recall that, if $v_i \in U^*$, then $d_U(v_i) \geq 4$ (otherwise one of the previous Rules applies), so the (at most two) branches where $u \in D_f$ diminish $l$ by at least 5. We have $T(l) \leq T(l-1)+6T(l-5)+6T(l-6)+4T(l-7)+T(l-8)$.
  – If $d_{U^*}(u) = 4$, we note that it cannot be the case that $\{v_1, v_2, v_3, v_4\} \subseteq D$, since this would mean that all $v_i$ are compatible for $u$ and Rule B4.1 would have applied. The branch corresponding to $S = \{v_1, v_2, v_3, v_4\}$ is therefore eliminated by the Sanity Check Rule. We consider two subcases:
    * At least one feasible pair exists in $N_{U^*}(u)$, therefore, $\{v_1, v_2\}$ is a feasible pair. Then $T(l) \leq T(l-1) + 6T(l-5) + 6T(l-6) + 5T(l-7)$.
    * No feasible pair exists. In this case the Sanity Check Rule eliminates all sets $S \subseteq N_U(u)$ that contain two or more vertices. We have $T(l) \leq T(l-1) + 7T(l-5)$.

- – $T(l) \leq T(l-1)+6T(l-5)+6T(l-6)+4T(l-7)+T(l-8)$ gives $x^8 = x^7 + 6x^3 + 6x^2 + 4x + 1$ with root $r < 1.8665$.
  – $T(l) \leq T(l-1) + 6T(l-5) + 6T(l-6) + 5T(l-7)$ gives $x^7 = x^6 + 6x^2 + 6x + 5$ with root $r < 1.8700$.
  – $T(l) \leq T(l-1)+7T(l-5)$ gives $x^5 = x^4+7$ with root $r < 1.7487$.

We are now at a situation where all vertices $u \in U^*$ have $d_U(u) \geq 5$.

**Branching Rule (B5):** If there exists $u \in U^*$ with $d_U(u) \in \{5, 6, 7, 8\}$, then select such a $u$ with minimum $d_U(u)$ and let $i = d_U(u)$ and $N_U(u) = \{v_1, \ldots, v_i\}$. Again, without loss of generality we order the vertices of $N_{U^*}(u)$ first, that is, for $1 \leq j \leq d_{U^*}(u)$, we have $v_j \in U^*$. Branch on the following: $(D_f, P_f \cup \{u\})$; for each non-empty subset $S \subseteq N_U(u)$, let $S^c = N_U(u) \setminus S$,

branch on $(D_f \cup S, P_f \cup S^c)$; if $d_{U^*}(u) \geq 2$ branch on $(D_f \cup \{u\}, P_f \cup ((N_U(v_1) \cup N_U(v_2)) \setminus \{u\}))$; for $3 \leq j \leq d_{U^*}(u)$, branch on $(D_f \cup \{u\}, P_f \cup (N_U(v_j) \setminus \{u\}))$.

- We generalize the previous branching to higher degrees in the obvious way: if $u \in D$, either the two first of its $d_{U^*}(u)$ neighbors in $U^*$ are its private neighbors, or one of its remaining $d_{U^*}(u) - 2$ neighbors in $U^*$ is private.
- Let $i = d_U(u)$. Note that we then assume that $d_U(v_j) \geq i$ for all $j \in \{1, \ldots, d_{U^*}(u)\}$, since we selected $u$ with the minimum $d_U(u)$. Hence the branches where $u \in D_f$ diminish $l$ by at least $i + 1$. We then have $T(l) \leq T(l-1) + (i-1)T(l-i-1) + \sum_{j=1}^{i} \binom{i}{j} T(l-i-j)$, which corresponds to the case where $d_{U^*}(u) = d_U(u)$.
- $T(l) \leq T(l-1) + (i-1)T(l-(i+1)) + \sum_{j=1}^{i} \binom{i}{j} \cdot T(l-(i+j))$ gives $x^{2i} = x^{2i-1} + (i-1)x^{i-1} + \sum_{j=1}^{i} \binom{i}{j} \cdot x^{i-j}$.
  - If $i = 5$: $x^{10} = x^9 + 9x^4 + 10x^3 + 10x^2 + 5x + 1$ with root $r < 1.8473$.
  - If $i = 6$: $x^{12} = x^{11} + 11x^5 + 15x^4 + 20x^3 + 15x^2 + 6x + 1$ with root $r < 1.8104$.
  - If $i = 7$: $x^{14} = x^{13} + 13x^6 + 21x^5 + 35x^4 + 35x^3 + 21x^2 + 7x + 1$ with root $r < 1.7816$.
  - If $i = 8$: $x^{16} = x^{15} + 15x^7 + 28x^6 + 56x^5 + 70x^4 + 56x^3 + 28x^2 + 8x + 1$ with root $r < 1.7593$.

We are now at a situation where all vertices $u \in U^*$ have $d_U(u) \geq 9$.

**Branching Rule (B6):** If there exists $u \in U^*$ with $d_U(u) \geq 9$, then let $T = \{v_1, \ldots, v_9\} \subseteq N_U(u)$. Branch on the feasible sub-instances among the following: $(D_f \cup \{u\}, P_f)$; $(D_f, P_f \cup \{u\})$; for each (possibly empty) subset $S \subseteq T$, let $S^c = T \setminus S$, branch on $(D_f \cup S, P_f \cup S^c)$.

- We consider all possibilities (including the $T \subseteq P$), so one produced instance must be good.
- $T(l) \leq T(l-1) + T(l-2) + \sum_{j=0}^{9} \binom{9}{j} \cdot T(l-(9+j))$.
- $T(l) \leq T(l-1) + T(l-2) + \sum_{j=0}^{9} \binom{9}{j} \cdot T(l-(9+j))$ gives $x^{18} = x^{17} + x^{16} + x^9 + 9x^8 + 36x^7 + 84x^6 + 126x^5 + 126x^4 + 84x^3 + 36x^2 + 9x + 1$ with root $r < 1.8640$.

**Step 2:** Branch to eliminate $U$.

If none of the Rules above apply, we enter the second branching step of our algorithm, which only involves one Rule that will be applied exhaustively. We observe that we now have $U^* = \emptyset$:

**Lemma 2.8.** *If none of the Rules from R1 to B6 can be applied, then* $U^* = \emptyset$.

*Proof.* Suppose that none of the Rules up to B6 applies, then $U^* = \emptyset$: indeed all vertices $u \in U^*$ are handled according to whether $d_U(u)$ is 0 (R1), 1 (B1), 2 (B2.2), 3 (B3.3), 4 (B4.2), $5, 6, 7$, or 8 (B5), or higher (B6). □

The fact that $U^* = \emptyset$ means that all remaining undecided vertices belong to $P \cup I$, because they cannot have two private neighbors. We use this observation to branch until we eliminate all vertices of $G[U]$ with degree at least 2.

**Branching Rule (B7):** If there exists $u \in U$ with $d_U(u) \geq 2$, then branch on the following two sub-instances: $(D_f, P_f \cup \{u\})$ and $(D_f, P_f \cup N_U(u))$.

- If $U^* = \emptyset$ then $U \subseteq P \cup I$, because no vertex of $U$ can have two private neighbors. Hence, if $u \in I$ then $N_U(u) \subseteq P$ and the Rule B7 is correct.
- $T(l) \leq T(l-1) + T(l-2)$, because $|N_U(u)| \geq 2$.
- $T(l) \leq T(l-1) + T(l-2)$ gives $x^2 = x + 1$ with root $r < 1.6181$.

**Step 3:** Complete the solution.

We are now in a situation where $U^* = \emptyset$ and $G[U]$ has maximum degree 1. As for the $O^*(1.7321^k)$ FPT algorithm for Max Min Vertex Cover we have presented in Section 1.4, the problem Mixed Dominating Set is polynomially solvable when the vertices left form a set of connected component each of size at most 2. We recall that Theorem 2 of [XS19] showed that this problem can now be solved optimally in polynomial time by collapsing all edges of $G[U]$ and then performing a minimum edge cover computation. We recall the relevant result, translated to our terminology:

**Lemma 2.9** (Theorem 2 of [XS19])**.** *For an instance characterized by* $(D_f, P_f)$ *which has* $U^* = \emptyset$ *and* $G[U]$ *has maximum degree* 1*, we can compute in polynomial time a minimum mixed dominating set* $D \cup M$ *satisfying* $D_f \subseteq D$, $P_f \subseteq V(M)$.

We are now ready to put everything together to obtain the promised algorithm.

**Theorem 2.10.** *Mixed Dominating Set parameterized by the size of the solution $k$ can be solved in time $O^*(3.510^k)$.*

*Proof.* The algorithm applies the Rules exhaustively and when no Rule applies invokes Lemma 2.9. Fix an optimal nice mds partition $V = D \cup P \cup I$. We assume that no isolated vertices exist, so such a nice partition exists by Lemma 2.3.

For correctness, we need to argue that if the cost $|D| + \frac{|P|}{2}$ is at most $k$, the algorithm will indeed output a solution of cost at most $k$. Observe that the initial instance is good, and we always produce a correct instance when we branch since all our Rules are correct, and by Lemma 2.9 the solution is optimally completed when no Rule applies, so if the optimal partition has cost at most $k$, the algorithm will produce a valid solution of cost at most $k$.

Let us now analyse the running time. We observe first that we can decide if a Rule applies in polynomial time, and the algorithm of Lemma 2.9 runs in polynomial time. We therefore only need to bound the number of subinstances the branching step will produce, as a function of $k$.

Of all the branching vectors, the worst case if given by Branching Rule B3.2, which leads to a complexity of $1.8734^l$. Taking into account that $l \leq 2k$, the running time of our algorithm is $O^*(1.8734^l) = O^*(3.510^k)$. □

We have described our improved FPT algorithm parameterized by the solution size $k$ for Mixed Dominating Set. In the next Section, we improve the best FPT algorithm parameterized by the treewidth, and prove that this algorithm and the one of Jain et al. [JJPS17] for pathwidth are optimal under the SETH.

## 2.4 Tight Treewidth Algorithm

We begin this Section with an algorithm for Min Mixed Dominating Set running in time $O^*(5^{tw})$. We rely on three ingredients: (i) the fact that Min Mixed Dominating Set on $G$ is equivalent to Min Distance-2-Dominating Set on the incidence graph of $G$ by a result of Madathil et al. [MPSS19] (ii) the standard fact that the incidence graph of $G$ has the same treewidth as $G$ (iii) and an $O^*(5^{tw})$ algorithm (by Borradaile and Le [BL16]) for Min Distance-2-Dominating Set.

**Theorem 2.11.** *There is an $O^*(5^{tw})$-time algorithm for Min Mixed Dominating Set in graphs of treewidth $tw$.*

*Proof.* We are given an instance of Min Mixed Dominating Set $G = (V, E)$. We first construct the incidence graph of $G$, which has vertex set

$V \cup E$, and has an edge between $v \in V$ and $e \in E$ if $e$ is incident on $v$ in $G$. We denote this graph as $I(G)$. In other words, $I(G)$ is obtained by sub-dividing every edge of $G$ once.

We now note the standard fact that $tw(I(G)) \leq tw(G)$. Indeed, if $G$ is a forest, then $I(G)$ is also a forest; while if $tw(G) \geq 2$, then we can take any tree decomposition of $G$ and for each $e = (u, v)$ we observe that it must contain a bag with both $u$ and $v$. We create a bag containing $u, v, e$ and attach it to the bag containing $u, v$. Note that this does not increase the width of the decomposition. We thus obtain a decomposition of $I(G)$ of width $tw(G)$.

Second, as observed by [MPSS19], every mixed dominating set of $G$ corresponds to a distance-2 dominating set of $I(G)$. Recall that a distance-2 dominating set of a graph is a set of vertices $D$ such that all vertices of $V \setminus D$ are at distance at most 2 from $D$.

Finally, we use the algorithm of [BL16] to solve Min Distance-2-Dominating Set in time $O^*(5^{tw})$ in $I(G)$, which gives us the optimal mixed dominating set of $G$. □

Now that we have presented an improved FPT algorithm parameterized by the treewidth, we show next that this algorithm and the one for pathwidth of Jain et al. [JJPS17] running in time $O^*(5^{pw})$ are optimal under the SETH.

Indeed, we prove that, under SETH, for any $\varepsilon > 0$, there is no algorithm for Min Mixed Dominating Set running in time $O^*((5 - \varepsilon)^{pw})$. The starting point of our reduction is the problem $q$-CSP-5. In this problem we are given a Constraint Satisfaction (CSP) instance with $n$ variables and $m$ constraints. The variables take values in a set of size 5, say $\{0, 1, 2, 3, 4\}$. Each constraint involves at most $q$ variables and is given as a list of acceptable assignments for this variables, where an acceptable assignment is a $q$-tuple of values from the set $\{0, 1, 2, 3, 4\}$ given to the $q$ variables. The following result was shown by Lampis [Lam20] to be a natural consequence of the SETH, and is a special case with $B = 5$ of the result of Lampis we have presented in Section 1.4.

**Lemma 2.12** (Theorem 3.1 by [Lam20])**.** *If the SETH is true, then for all $\varepsilon > 0$, there exists a $q$ such that $n$-variable $q$-CSP-5 cannot be solved in time $O^*((5 - \varepsilon)^n)$.*

As we have mentioned in Section 1.4, in this theorem of Lampis [Lam20] it was shown that, for any alphabet size $B$, $q$-CSP-$B$ cannot be solved in time $O^*\left((B - \varepsilon)^n\right)$ under the SETH, but for our purposes only the case $B = 5$ is relevant, for two reasons: because this corresponds to the base of our target lower bound; and because in our construction we will represent the $B = 5$

possible values for a variable with a path of five vertices in which there exist exactly five different ways of selecting one vertex and one edge among these five vertices.

Our plan is therefore to produce a polynomial-time reduction which, given a $q$-CSP-5 instance with $n$ variables, produces an equivalent Min Mixed Dominating Set instance whose pathwidth is at most $n + O(1)$. Then, the existence of an algorithm for the latter problem running faster than $O^*\left((5-\varepsilon)^{pw}\right)$ would give an $O^*\left((5-\varepsilon)^n\right)$ algorithm for $q$-CSP-5, contradicting the SETH.

Before giving the details of our reduction let us sketch the basic ideas, which follow the pattern of other SETH-based lower bounds which have appeared in the literature: see [HKL$^+$18, JJ17, KLP18, KLP19b, KLP19a, LMS18]. The constructed graph consists of a main selection part of $n$ paths of length $5m$, divided into $m$ sections. Each path corresponds to a variable and each section to a constraint. The idea is that the optimal solution will follow for each path a basic pattern of selecting one vertex and one edge among the first five vertices and then repeat this pattern throughout the path (see Figure 2.2). There are 5 natural ways to do this, so this can represent all assignments to the $q$-CSP-5 instance. We will then add verification gadgets to each section, connected only to the vertices of that section that represent variables appearing in the corresponding constraint (thus keeping the pathwidth under control), in order to check that the selected assignment satisfies the constraint.

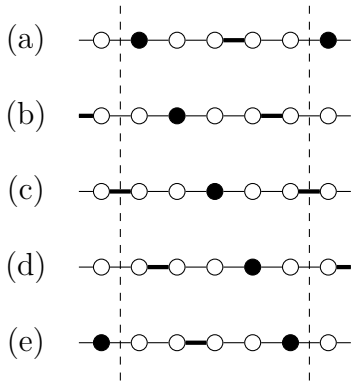


Figure 2.2: Main part of the construction with the five possible configurations. Filled vertices are in $D$, thick edges are in $M$.

The main difficulty in completing the proof is showing that the optimal solution has the desired form, and in particular, that the pattern that is selected for a variable is kept constant throughout the construction. This

is in general not possible to prove, but using a technique introduced by [LMS18], we work around this difficulty by making polynomially many copies of our construction, gluing them together, and arguing that a large enough consistent copy must exist.

We now present the reduction. We are given a $q$-CSP-5 instance $\varphi$ with $n$ variables $x_1, \ldots, x_n$ taking values over the set $\{0, 1, 2, 3, 4\}$, and $m$ constraints $c_0, \ldots, c_{m-1}$. For each constraint we are given a set of at most $q$ variables which are involved in this constraint and a list of satisfying assignments for these variables. Without loss of generality, we make the following assumptions: (i) each constraint involves exactly $q$ variables, because if it has fewer variables, we can add to it new variables and augment the list of satisfying assignments so that the value of the new variables is irrelevant (ii) all constraints have lists of satisfying assignments of size $C = 5^q - 1$; note that this is an upper bound on the size of the list of satisfying assignments since if a constraint has $5^q$ different satisfying assignments then it is always satisfied and thus is redundant; and for each constraint which has fewer we add several copies of one of its satisfying assignments to its list (so the list may repeat an assignment). We define two "large" numbers $F = (4n+1)(2n+1)$ and $A = 12$ and we set our budget to be $k = 8AFmn + 2Fmn + 2Fmq(C-1) + n + 1$.

We now construct our graph as follows:

1. We construct a vertex $s$ and attach to it two leaves $s_1, s_2$.

2. For $i \in \{1, \ldots, n\}$ we construct a path on $5Fm$ vertices: the vertices are labeled $u_{i,j}$, for $j \in \{0, 1, \ldots, 5Fm - 1\}$ and for each $i, j$ the vertex $u_{i,j}$ is connected to $u_{i,j+1}$. We call these paths the *main* part of our construction.

3. For each $j \in \{0, 1, \ldots, Fm - 1\}$, let $j' = j \bmod m$. We construct a checker gadget $H_j$ as follows (see Figure 2.3):

   (a) For each satisfying assignment $\sigma$ in the list of the constraint $c_{j'}$, we construct an independent set $Z_{\sigma,j}$ of size $2q$ (therefore, $C$ such independent sets). The $2q$ vertices are partitioned so that for each of the $q$ variables involved in $c_{j'}$ we reserve two vertices. In particular, if $x_i$ is involved in $c_{j'}$ we denote by $z^1_{\sigma,j,i}, z^2_{\sigma,j,i}$ its two reserved vertices in $Z_{\sigma,j}$.

   (b) For each $i \in \{1, \ldots, n\}$ such that $x_i$ is involved in $c_{j'}$, for each satisfying assignment $\sigma$ in the list of $c_{j'}$, if $\sigma$ sets $x_i$ to value $\alpha \in \{0, 1, 2, 3, 4\}$ we add the following edges:

      i. $(u_{i,5j+\alpha}, z^1_{\sigma,j,i})$ and $(u_{i,5j+\alpha}, z^2_{\sigma,j,i})$.

ii. Let $\beta = (\alpha + 2) \bmod 5$ and $\gamma = (\alpha + 3) \bmod 5$. We add the edges $(u_{i,5j+\beta}, z^1_{\sigma,j,i})$ and $(u_{i,5j+\gamma}, z^2_{\sigma,j,i})$.

(c) For all assignments $\sigma \neq \sigma'$ of $c_{j'}$, add all edges between $Z_{\sigma,j}$ and $Z_{\sigma',j}$.

(d) We construct an independent set $W_j$ of size $2q(C-1)$.

(e) Add all edges between $W_j$ and $Z_{\sigma,j}$, for all assignments $\sigma$ of $c_{j'}$.

(f) For each $w \in W_j$, we construct an independent set of size $2k+1$ whose vertices are all connected to $w$ and to $s$.

4. We define the consistency gadget $Q_{i,j}$, for $i \in \{1, \ldots, n\}$ and $j \in \{0, \ldots, Fm-1\}$ which consists of (see Figure 2.3):

(a) An independent set of size 8 denoted $A_{i,j}$.

(b) Five independent sets of size 2 each, denoted $B_{i,j,0}, B_{i,j,1}, \ldots, B_{i,j,4}$.

(c) For each $\ell, \ell' \in \{0, \ldots, 4\}$ with $\ell \neq \ell'$ all edges from $B_{i,j,\ell}$ to $B_{i,j,\ell'}$.

(d) For each $\ell \in \{0, \ldots, 4\}$ all possible edges from $B_{i,j,\ell}$ to $A_{i,j}$.

(e) For each $a \in A_{i,j}$, $2k+1$ vertices connected to $a$ and to $s$.

(f) For each $\ell \in \{0, \ldots, 4\}$ both vertices of $B_{i,j,\ell}$ are connected to $u_{i,5j+\ell}$.

(g) For each $\ell \in \{0, \ldots, 4\}$ let $\ell' = (\ell + 2) \bmod 5$ and $\ell'' = (\ell + 3) \bmod 5$. One vertex of $B_{i,j,\ell}$ is connected to $u_{i,5j+\ell'}$ and the other to $u_{i,5j+\ell''}$.

5. For each $i \in \{1, \ldots, n\}$ and $j \in \{0, \ldots, Fm-1\}$ construct $A$ copies of the gadget $Q_{i,j}$ and connect them to the main part as described above.

This completes the construction. The target size is $k$, as defined above. We now argue that the reduction is correct and $G$ has the desired pathwidth.

**Lemma 2.13.** *If $\varphi$ is satisfiable, then there exists a mixed dominating set in $G$ of size at most $k$.*

*Proof.* Assume that $\varphi$ admits some satisfying assignment $\rho : \{x_1, \ldots, x_n\} \to \{0, 1, 2, 3, 4\}$. We construct a solution as follows:

1. For each $i \in \{1, \ldots, n\}$ let $\alpha = \rho(x_i)$. For each $j \in \{0, \ldots, Fm-1\}$, we select in the dominating set the vertex $u_{i,5j+\alpha}$.

2. Let $U'$ be the set of vertices $u_{i,j}$ of the main part which were not selected in the previous step and which do not have a neighbor selected in the previous step. We add to the solution all edges of a maximum matching of $G[U']$, as well as all vertices of $U'$ left unmatched by this matching.

3. For each $j \in \{0, \ldots, Fm-1\}$, $G$ contains a gadget $H_j$. Consider the constraint $c_{j'}$ for $j' = j \bmod m$. Let $\sigma$ be an assignment in the list of $c_{j'}$ that agrees with $\rho$ (such a $\sigma$ must exist, since the constraint is satisfied by $\rho$). We add to the solution the edges of a perfect matching from $W_j$ to $\bigcup_{\sigma' \neq \sigma} Z_{\sigma',j}$.

4. For each $j \in \{0, \ldots, Fm-1\}$ and $i \in \{1, \ldots, n\}$ we have added to the graph $A$ copies of the consistency gadget $Q_{i,j}$. For each copy we add to the solution a perfect matching from $A_{i,j}$ to $\bigcup_{\ell \neq \rho(x_i)} B_{i,j,\ell}$.

5. We set $s \in D$.

Let us first argue why this solution has size at most $k$. In the first step we select $Fnm$ vertices. In the second step we select at most $Fnm+n$ elements. To see this, note that if $u_{i,j}$ is taken in the previous step, then $u_{i,j+5}$ is also taken (assuming $j+5 < 5Fm$), which leaves two adjacent vertices $u_{i,j+2}$ and $u_{i,j+3}$. These vertices will be matched in $G[U']$ and in our solution. Note that, for a variable $x_i$, if $\rho(x_i) \neq 2$, then at most one vertex is left unmatched by the matching taken, so the cost for this variable is at most $Fm+1$. If $\rho(x_i) = 2$, then at most two vertices are left matched by the matching taken, so the cost for this variable is at most $(Fm-1)+2$. Furthermore, for each $H_j$ we select $|W_j| = 2q(C-1)$ edges. For each copy of $Q_{i,j}$ we select 8 edges, for a total cost of $8AFmn$. Taking into account $s$, the total cost is at most $Fnm + Fnm + n + 2Fmq(C-1) + 8AFmn + 1 = k$.

Let us argue why the solution is feasible. First, all vertices $u_{i,j}$ and all edges connecting them to each other are dominated by the first two steps of our selection since we have taken a maximum matching in $G[U']$ and all vertices left unmatched by this matching. Second, for each $H_j$, the vertex $s$ together with the endpoints of selected edges form a vertex cover of $H_j$, so all internal edges are dominated. Furthermore, $s$ dominates all vertices which are not endpoints of our solution, except $Z_{\sigma,j}$, where $\sigma$ is the selected assignment of $c_{j'}$, with $j' = j \bmod m$. We then need to argue that the vertices of $Z_{\sigma,j}$ and the edges connecting it to the main part are covered.

Recall that the $2q$ vertices of $Z_{\sigma,j}$ are partitioned into pairs, with each pair $z^1_{\sigma,j,i}, z^2_{\sigma,j,i}$ reserved for the variable $x_i$ involved in $c_{j'}$. We now claim that $z^1_{\sigma,j,i}, z^2_{\sigma,j,i}$ are dominated by our solution, since we have selected the vertex $u_{i,5j+\alpha}$, where $\alpha = \rho(x_i)$. Furthermore, $u_{i,5j+\beta}, u_{i,5j+\gamma}$, where $\beta = (a+$

2) mod $m$, $\gamma = (a+3) \bmod m$, belong in $U'$ and therefore the edges incident to them are covered. Finally, to see that the $Q_{i,j}$ gadgets are covered, observe that for each such gadget only 2 vertices of some $B_{i,j,\ell}$ are not endpoints of selected edges. The common neighbor of these vertices has been selected in the first step, and their other neighbors in the main part belong to $U'$. □

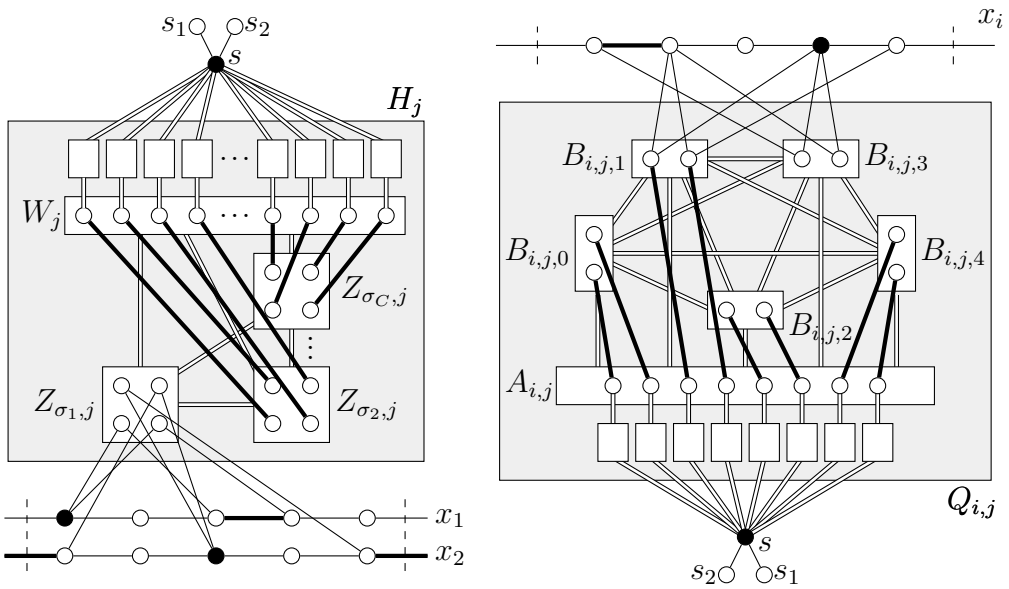


Figure 2.3: (Double edges between two sets of vertices represent all edges between the two sets.) Left: Checker gadget $H_j$ connected to the main part. Here we have considered an instance where the clause $c_{j'}$ has only two variables, $x_1$ and $x_2$. Moreover, only the independent set $Z_{\sigma_1,j}$ is shown connected to the main part. The possible assignment $\sigma_1$ of $c_{j'}$ is $(x_1 = 0, x_2 = 2)$. We have supposed that this assignment is satisfiable, and we have marked the corresponding mixed dominating set: filled vertices are in $D$, thick edges are in $M$. Right: Checker gadget $Q_{i,j}$ connected to the main part, that is to the path corresponding to the variable $x_i$. Only the independent sets $B_{i,j,1}$ and $B_{i,j,3}$ are shown connected to the main part. We have supposed that the assignment $(x_i = 3)$ is satisfiable, and we have marked the corresponding mixed dominating set: filled vertices are in $D$, thick edges are in $M$.

We can now prove the other direction. The idea of the proof of the next lemma is the following: by partitioning the graph into different parts and lower bounding the cost of these parts, we prove that if a mixed dominating set in $G$ has not the same form as in Lemma 2.13 in a sufficiently large copy, then it has size strictly greater than $k$, enabling us to produce a satisfiable

assignment for $\varphi$ using the mixed dominating set which has the desired form. Note that we use the definition of nice mixed dominating set we have given in Section 2.1 (Definition 2.2 and Lemma 2.3), and in particular the fact that $D \cap V(M) = \emptyset$ in order to have a sufficiently large copy where a nice mixed dominating set of minimum size has the same form as in Lemma 2.13.

**Lemma 2.14.** *If there exists a mixed dominating set in $G$ of size at most $k$, then $\varphi$ is satisfiable.*

*Proof.* Suppose that we are given, without loss of generality (Lemma 2.3), a nice mixed dominating set $D \cup M$ of $G$ of minimum size. We therefore have a partition of $V$ into $V = D \cup P \cup I$. Before proceeding, let us define for a set $S \subseteq V$ its *cost* as $\text{cost}(S) = |S \cap D| + \frac{|S \cap P|}{2}$. Clearly, $\text{cost}(V) \leq k$ since $|M| \geq |P|/2$, and for disjoint sets $S_1, S_2$ we have $\text{cost}(S_1 \cup S_2) = \text{cost}(S_1) + \text{cost}(S_2)$. Our strategy will therefore be to partition $V$ into different parts and lower bound their cost.

First, we give some notation. Consider some $j \in \{0, \ldots, Fm-1\}$ and $i \in \{1, \ldots, n\}$: recall that we have constructed $A$ copies of the gadget $Q_{i,j}$, call them $Q^1_{i,j}, \ldots, Q^A_{i,j}$; also we define the sets $S_{i,j} = \{u_{i,5j}, u_{i,5j+1}, \ldots, u_{i,5j+4}\}$. Now, for some $j \in \{0, \ldots, Fm-1\}$, let:

$$S_j = H_j \cup \bigcup_{i \in \{1,\ldots,n\}} \left( S_{i,j} \cup \bigcup_{r \in \{1,\ldots,A\}} Q^r_{i,j} \right) \tag{2.1}$$

**Claim 2.15.** $\text{cost}(S_j) \geq 2q(C-1) + 2n + 8An$.

*Proof.* We begin with some easy observations. First, it must be the case that $s \in D$. If not, either $s_1$ or $s_2$ are in $D$, which contradicts the niceness of the solution, i.e. the fact that every vertex of $D$ has two private neighbors.

Consider some $j \in \{0, \ldots, Fm-1\}$ and $i \in \{1, \ldots, n\}$. We will say that, for $1 \leq r \leq A$, $Q^r_{i,j}$ is *normal* if we have the following: $Q^r_{i,j} \cap D = \emptyset$ and there exists $\ell \in \{0, \ldots, 4\}$ such that $Q^r_{i,j} \cap P = A_{i,j} \cup \bigcup_{\ell' \neq \ell} B_{i,j,\ell'}$. In other words, $Q^r_{i,j}$ is normal if locally the solution has the form described in Lemma 2.13.

We now observe that for all $i, j, r$ we have $\text{cost}(Q^r_{i,j}) \geq 8$. To see this, observe that, if there exists $a \in A_{i,j} \cap I$, then the $2k+1$ neighbors of $a$ must be in $D \cup P$, so the solution cannot have cost $k$. Hence, $A_{i,j} \subseteq D \cup P$. Furthermore, the maximum independent set of $\bigcup_{\ell \in \{0,\ldots,4\}} B_{i,j,\ell}$ is 2, so $|(\bigcup_{\ell \in \{0,\ldots,4\}} B_{i,j,\ell}) \cap (D \cup P)| \geq 8$. So $\text{cost}(Q^r_{i,j}) \geq 8$. Following this reasoning we also observe that if $Q^r_{i,j}$ is not normal, then we have $\text{cost}(Q^r_{i,j}) > 8$. In other words, 8 is a lower bound for the cost of every copy of $Q_{i,j}$, which can only be attained if a copy is normal.

Consider some $j \in \{0, \ldots, Fm-1\}$ and $i \in \{1, \ldots, n\}$ and suppose that none of the $A$ copies of $Q_{i,j}$ is normal. We will then arrive at a contradiction. Indeed, we have $\text{cost}(\bigcup_r Q^r_{i,j}) \geq 8A + A/2 \geq 8A + 6$. We create another solution by doing the following: take the five vertices $u_{i,5j}, u_{i,5j+1}, \ldots, u_{i,5j+4}$, and take in all $Q_{i,j}$ a matching so that $Q_{i,j}$ is normal. This has decreased the total cost, while keeping the solution valid, which should not be possible.

We can therefore assume from now on that for each $i, j$ at least one copy of $Q_{i,j}$ is normal, hence, there exists $\ell \in \{0, \ldots, 4\}$ such that $B_{i,j,\ell} \subseteq I$ in that copy.

Recall that $S_{i,j} = \{u_{i,5j}, u_{i,5j+1}, \ldots, u_{i,5j+4}\}$. We claim that for all $i \in \{1, \ldots, n\}, j \in \{0, \ldots, Fm-1\}$, we have $\text{cost}(S_{i,j}) \geq 2$. Indeed, if we consider the normal copy of $Q_{i,j}$ which has $B_{i,j,\ell} \subseteq I$, the two vertices of $B_{i,j,\ell}$ have three neighbors in $S_{i,j}$, and at least one of them must be in $D$ to dominate the vertices of $B_{i,j,\ell}$.

In addition, we claim that for all $j \in \{0, \ldots, Fm-1\}$ we have $\text{cost}(H_j) \geq 2q(C-1)$. The reasoning here is similar to $Q_{i,j}$, namely, the vertices of $W_j$ cannot belong to $I$ (otherwise we get $2k+1$ vertices in $D \cup P$); and from the $2qC$ vertices in $\bigcup_\sigma Z_{\sigma,j}$ at most $2q$ can belong to $I$.

We now have the lower bounds we need: $\text{cost}(S_j) \geq 2q(C-1) + 2n + 8An$. □

Now, if for some $j$ we have $\text{cost}(S_j) > 2q(C-1) + 2n + 8An$ we will say that $j$ is *problematic*.

**Claim 2.16.** *There exists a contiguous interval $J \subseteq \{0, \ldots, Fm-1\}$ of size at least $m(4n+1)$ in which all $j \in J$ are not problematic.*

*Proof.* Let $L \subseteq \{0, \ldots, Fm-1\}$ be the set of problematic indices. We claim that $|L| \leq 2n$. Indeed, we have $\text{cost}(V) = 1 + \sum_{j \in \{0,\ldots,Fm-1\}} \text{cost}(S_j) \geq 1 + Fm(2q(C-1) + 2n + 8An) + |L|/2 = k - n + |L|/2$. But since the total cost is at most $k$, we have $|L|/2 \leq n$.

We will now consider the longest contiguous interval $J \subseteq \{0, \ldots, Fm-1\}$ such that all $j \in J$ are not problematic. We have $|J| \geq Fm/(|L|+1) \geq m(4n+1)$. □

Before we proceed further, we note that if $j$ is not problematic, then for any $i \in \{1, \ldots, n\}$, all edges of $M$ which have an endpoint in $S_{i,j}$ must have their other endpoint also in the main part, that is, they must be edges of the main paths. To see this note that, if $j$ is not problematic, all $Q_{i,j}$ are normal, so there are 8 vertices in $A_{i,j} \cap P$ which must be matched to the 8 vertices of $(\bigcup_\ell B_{i,j,\ell}) \cap P$. Similarly, in $H_j$ the $2q(C-1)$ vertices of $W_j \cap P$

must be matched to the $2q(C-1)$ vertices of $(\bigcup_\sigma Z_{\sigma,j}) \cap P$, otherwise we would increase the cost and $j$ would be problematic.

Consider now a non-problematic $j \in J$ and $i \in \{1, \ldots, n\}$ such that $\text{cost}(S_{i,j}) = 2$. We claim that the solution must follow one of the five configurations below (see also Figure 2.2):

(a) $u_{i,5j} \in D$ and $(u_{i,5j+2}, u_{i,5j+3}) \in M$.

(b) $u_{i,5j+1} \in D$ and $(u_{i,5j+3}, u_{i,5j+4}) \in M$.

(c) $u_{i,5j+2} \in D$, $(u_{i,5j+4}, u_{i,5j+5}) \in M$, and $(u_{i,5j-1}, u_{i,5j}) \in M$.

(d) $u_{i,5j+3} \in D$ and $(u_{i,5j}, u_{i,5j+1}) \in M$.

(e) $u_{i,5j+4} \in D$ and $(u_{i,5j+1}, u_{i,5j+2}) \in M$.

Indeed, these configurations cover all the cases where exactly one vertex of $S_{i,j}$ is in $D$ and exactly two are in $P$ and are endpoints of an edge of $M$. This is a condition enforced by the fact that all of the $Q_{i,j}$ copies are normal, and that $\text{cost}(S_{i,j}) = 2$.

**Claim 2.17.** *There exists a contiguous interval $J' \subseteq \{0, \ldots, Fm-1\}$ of size at least $m$ in which all $j \in J'$ are not problematic and, for all $j_1, j_2 \in J'$, $S_{i,j_1}$ and $S_{i,j_2}$ are in the same configuration.*

*Proof.* Given the five configurations, we now make the following simple observations, where statements apply for all $i \in \{1, \ldots, n\}$ and $j$ such that $j, j+1 \in J$:

- If $S_{i,j}$ is in configuration (a), then $S_{i,j+1}$ is also in configuration (a).
- If $S_{i,j}$ is in configuration (b), then $S_{i,j+1}$ is in configuration (a), (b), (c) or (d).
- If $S_{i,j}$ is in configuration (c), then $S_{i,j+1}$ is in configuration (c) or (d).
- If $S_{i,j}$ is in configuration (d), then $S_{i,j+1}$ is in configuration (a) or (d).
- If $S_{i,j}$ is in configuration (e), then $S_{i,j+1}$ is in configuration (a), (b), (d) or (e).

For the first claim, we note that in configuration (a) vertex $u_{i,5j+4}$ is not dominated, so $S_{i,j+1}$ cannot be in configuration (b), (d) and (e) by this fact, and cannot be in configuration (c) because otherwise $\text{cost}(S_{i,j}) > 2$. For the second claim, we note that $S_{i,j+1}$ cannot be in configuration (e)

because otherwise the vertex $u_{i,5(j+1)}$ is not dominated. For the third claim, we note that $S_{i,j+1}$ cannot be in configuration (a) since $D \cap P = \emptyset$, nor in configuration (b) and (e) because otherwise $\text{cost}(S_{i,j+1}) > 2$. For the fourth claim, $S_{i,j+1}$ cannot be in configuration (b) and (e) because otherwise the edge $(u_{i,5j+4}, u_{i,5(j+1)})$ is not dominated, nor in configuration (c) because otherwise $\text{cost}(S_{i,j}) > 2$. For the last claim, note that $S_{i,j+1}$ cannot be in configuration (c) since $D \cap P = \emptyset$.

We will now say for some $i \in \{1, \ldots, n\}$, $j \in J$, that $j$ is *shifted* for variable $i$ if $j+1 \in J$ but $S_{i,j}$ and $S_{i,j+1}$ do not have the same configuration. We observe that there cannot exist distinct $j_1, j_2, j_3, j_4, j_5 \in J$ such that all of them are shifted for variable $i$. Indeed, if we draw a directed graph with a vertex for each configuration, and an arc $(u, v)$ expressing the property that the configuration represented by $v$ can follow the one represented by $u$, if we take into account the observations above, the graph will be a DAG with maximum path length 4. Hence, a configuration cannot shift 5 times, as long as we stay in $J$ (the part of the graph where the minimum local cost is attained everywhere).

By the above, the number of shifted indices $j \in J$ is at most $4n$. Hence, the longest contiguous interval without shifted indices has length at least $|J|/(4n+1) \geq m$. Let $J'$ be this interval. □

We are now almost done: we have located an interval $J' \subseteq \{0, \ldots, Fm-1\}$ of length at least $m$ where for all $i \in \{1, \ldots, n\}$ and all $j_1, j_2 \in J'$ we have the same configuration in $S_{i,j_1}$ and $S_{i,j_2}$. We now extract an assignment from this in the natural way: if $u_{i,5j+\ell} \in D$, for some $j \in J', \ell \in \{0, \ldots, 4\}$, then we set $x_i = \ell$. We claim this satisfies $\varphi$. Consider a constraint $c_{j'}$ of $\varphi$. There must exist $j \in J'$ such that $j' = j \bmod m$, because $|J'| \geq m$ and $J'$ is contiguous. We therefore check $H_j$, where there exists $\sigma$ such that $Z_{\sigma,j} \subseteq I$ (this is because $j$ is not problematic, that is, $H_j$ attains the minimum cost). But because the vertices and incident edges of $Z_{\sigma,j}$ are dominated, it must be the case that the assignment we extracted agrees with $\sigma$, hence $c_{j'}$ is satisfied. □

We now show that the pathwidth of $G$ is at most $n + O(1)$.

**Lemma 2.18.** *The pathwidth of $G$ is at most $n + O(q5^q)$.*

*Proof.* We will show how to build a path decomposition. First, we can add $s$ to all bags, so we focus on the rest of the graph. Second, after removing $s$ from the graph, some vertices become leaves. It is a well-known fact that removing all leaves from a graph can only increase the pathwidth by at most 1. To see this, let $G'$ be the graph obtained after deleting all leaves of $G$ and

suppose we have a path decomposition of $G'$ of width $w$. We obtain a path decomposition of $G$ by doing the following for every leaf $v$: find a bag of width at most $w$ that contains the neighbor of $v$ and insert after this bag, a copy of the bag with $v$ added. Clearly, the width of the new decomposition is at most $w+1$. Because of this we will ignore all vertices of $G$ which become leaves after the removal of $s$.

For all $j \in \{0, \ldots, Fm-1\}$, let $S_j$ be defined as in Equation (2.1). We will show how to build a path decomposition of $G[S_j]$ with the following properties:

- The first bag of the decomposition contains vertices $u_{i,5j}$, for all $i \in \{1, \ldots, n\}$.
- The last bag of the decomposition contains vertices $u_{i,5j+4}$, for all $i \in \{1, \ldots, n\}$.
- The width of the decomposition is $n + O(q5^q)$.

If we achieve the above then we can obtain a path decomposition of the whole graph: indeed, the sets $S_j$ partition all remaining vertices of the graph, while the only edges not covered by the above decompositions are those between $u_{i,5j+4}$ and $u_{i,5(j+1)}$. We therefore place the decompositions of $S_j$ in order, and then between the last bag of the decomposition of $S_j$ and the first bag of the decomposition of $S_{j+1}$ we have $2n$ "transition" bags, where in each transition step we add a vertex $u_{i,5(j+1)}$ in the bag, and then remove $u_{i,5j+4}$.

Let us now show how to obtain a decomposition of $G[S_j]$, having fixed the contents of the first and last bag. First, $H_j$ has order $O(q5^q)$ after handling the leaves, so we place all its vertices to all bags. The remaining graph is a union of paths of length 4 with the $Q_{i,j}$ gadgets attached. We therefore have a sequence of $O(n)$ bags, where for each $i \in \{1, \ldots, n\}$ we add to the current bag the vertices of $S_{i,j}$, then add and remove one after another whole copies of $Q_{i,j}$, then remove $S_{i,j}$ except for $u_{i,5j+4}$. □

We are now ready to present the main result of this section. By putting together Lemmas 2.13, 2.14, 2.18 and the negative result for $q$-CSP-5 (Lemma 2.12), we get the following theorem:

**Theorem 2.19.** *Under SETH, for all $\varepsilon > 0$, no algorithm solves* Min Mixed Dominating Set *in time $O^*((5-\varepsilon)^{pw})$, where $pw$ is the pathwidth of the input graph.*

*Proof.* Fix $\varepsilon > 0$ and let $q$ be sufficiently large so that Lemma 2.12 is true. Consider an instance $\varphi$ of $q$-CSP-5. Using our reduction, create an instance $(G, k)$ of Min Mixed Dominating Set. Thanks to Lemmas 2.13 and 2.14, we know that $\varphi$ is satisfiable if and only if there exists a mixed dominating set of size at most $k$ in $G$.

Suppose there exists an algorithm which solves Min Mixed Dominating Set in time $O^*((5-\varepsilon)^{pw})$. With this algorithm and our reduction, we can determine if $\varphi$ is satisfiable in time $O^*((5-\varepsilon)^{pw})$, where $pw = n + O(q5^q) = n + O(1)$, so the total running time of this procedure is $O^*((5-\varepsilon)^n)$, contradicting the SETH. □

With this theorem, we proved that our algorithm for Min Mixed Dominating Set for treewidth and the one of Jain et al. [JJPS17] for pathwidth of complexity $O^*(5^{tw})$ and $O^*(5^{pw})$, respectively, are optimal under the SETH.

# Chapter 3

# Max Min Feedback Vertex Set

In a graph $G = (V, E)$ with $|V| = n$, a set $S \subseteq V$ is called a *feedback vertex set* (fvs for short) if the subgraph induced by $V \setminus S$ is a forest. Typically, Min Feedback Vertex Set is studied with a minimization objective: given a graph we are interested in finding the best (that is, smallest) fvs. In this chapter we are interested in an objective which is, in a sense, the inverse: we seek an fvs $S$ which is as *large* as possible, while still being minimal. We call this problem Max Min Feedback Vertex Set. Since it is the Max-Min version of Min Feedback Vertex Set, the problem has a mandatory private structure, given as follows: for every vertex $u$ in the solution, $u$ must have at least one private cycle only dominated by $u$.

Max-Min and Min-Max versions of many famous optimization problems have recently attracted much interest in the literature, such as Upper Dominating Set [BBC+18a] Max Min Separator [HBvdZO19], Max Min Cut [EHKK19], Min Max Knapsack (also known as Lazy Bureaucrat) [ABMS03, FLS17, GMP13], and some variants of Max Min Edge Cover [KGMS20, HGM+20]. Some problems in this area also arise naturally in other forms and have been extensively studied, such as Min Edge Dominating Set [IN16], Grundy Coloring, which can be seen as a Max-Min version of Coloring [ABKS20, BKL+20], and Max Min Vertex Cover in hypergraphs, which is known as Upper Transversal [MRRS12, HY18, HY19, HY20]. Although the initial motivation for studying such problems was a desire to analyze the worst possible performance of a naive heuristic, these problems have gradually been revealed to possess a rich combinatorial structure that makes them interesting in their own right.

Max Min Feedback Vertex Set, which is a member of this framework of Max-Min and Min-Max problems, was first considered by Mishra and Sikdar [MS01], who showed that the problem does not admit an $n^{1/2-\epsilon}$-approximation under $\mathbf{P} \neq \mathbf{NP}$, and that it remains **APX**-hard for $\Delta \geq 9$. On

this last front, we will first improve the **NP**-hardness of Max Min Feedback Vertex Set and show that it is **NP**-hard for $\Delta \geq 6$ on planar bipartite graphs. We present this result in the next section.

On the other hand, the two Max-Min problems Upper Dominating Set and Max Min Vertex Cover are well-studied problems, in the context of approximation [AHL+18, BBC+18b, BCP15, BM17, CFHJ90, Dem99, HP20, JP90, ZZ95]. Our motivation for focusing on Max Min Feedback Vertex Set is the contrast between these two well-studied cousins: the Max Min Vertex Cover and Upper Dominating Set problems, where the objective is to find the largest minimal vertex cover or dominating set, respectively. At first glance, one would expect Max Min Vertex Cover to be the easiest of these two problems: both problems can be seen as trying to find the largest minimal hitting set of a hypergraph, but in the case of Max Min Vertex Cover the hypergraph has a very restricted structure, while in Upper Dominating Set the hypergraph is essentially arbitrary. This intuition turns out to be correct: while Upper Dominating Set admits no $n^{1-\epsilon}$-approximation (see [BBC+18a]), Max Min Vertex Cover admits a $\sqrt{n}$-approximation (but no $n^{1/2-\epsilon}$-approximation) [BCP15], as we have presented in Section 1.3.

This background leads us to the natural question of the approximability of Max Min Feedback Vertex Set. On an intuitive level, one may be tempted to think that this problem should be harder than Max Min Vertex Cover, since hitting cycles is more complex than hitting edges, but easier than Upper Dominating Set, since hitting cycles still offers us more structure than an arbitrary hypergraph. However, to the best of our knowledge, no $n^{1-\epsilon}$-approximation algorithm is currently known for Max Min Feedback Vertex Set (so the problem could be as hard as Upper Dominating Set), and the best hardness result of approximation known is $n^{1/2-\epsilon}$ [MS01] (so the problem could be as easy as Max Min Vertex Cover).

We fully answer this question in this chapter, confirming and precisely quantifying the intuition that Max Min Feedback Vertex Set is a problem that lies "between" Max Min Vertex Cover and Upper Dominating Set: we give in Section 3.3 a polynomial-time approximation algorithm with ratio $O(n^{2/3})$ and a hardness of approximation in Section 3.4 which shows that under $\mathbf{P} \neq \mathbf{NP}$ no polynomial-time algorithm can obtain a ratio of $n^{2/3-\epsilon}$, for any $\epsilon > 0$. This completely settles the approximability of the problem in polynomial time. Along the way, we also prove in Section 3.2 that Max Min Feedback Vertex Set has an approximation algorithm with ratio $O(\Delta)$, show in Section 3.4 that no algorithm can achieve ratio $\Delta^{1-\epsilon}$ for any $\epsilon > 0$, and give in Section 3.3 a cubic kernel when parameterized by the

solution size.

One interesting aspect of our results is that they have an interpretation from extremal combinatorics which nicely mirrors the situation for Max Min Vertex Cover. Recall that a corollary of the $\sqrt{n}$-approximation for Max Min Vertex Cover [BCP15], which we have mentioned in Section 1.3, is that any graph without isolated vertices has a minimal vertex cover of size at least $\sqrt{n}$, and this is tight (see Remark 3.15). Hence, the algorithm only needs to trivially preprocess the graph (deleting isolated vertices) and then find this set, which is guaranteed to exist, in polynomial time, as we have showed in Section 1.3. Our algorithm can be seen in a similar light: we prove that if one applies two almost trivial preprocessing rules we present in Section 3.2 to a graph (deleting leaves and contracting edges between degree-two vertices), a minimal fvs of size at least $n^{1/3}$ (and $\Omega(n/\Delta)$) is always guaranteed to exist, and this is tight (Corollary 3.12 and Remark 3.13). Thus, the approximation ratio of $n^{2/3}$ is automatically guaranteed for any graph where we exhaustively apply these very simple rules and our algorithms only have to work to construct the promised set. This makes it somewhat remarkable that the ratio of $n^{2/3}$ turns out to be best possible.

Concerning the super-polynomial approximation of Max Min Feedback Vertex Set, no result is currently known to the best of our knowledge. We thus have studied this framework for this problem, and in Section 3.5, we generalize our $O(n^{2/3})$-approximation described in Section 3.3 to obtain a super-polynomial approximation algorithm which is able to guarantee any desired performance, at the cost of increased running time. More precisely, we describe an $r$-approximation algorithm for any $r \leq n^{2/3}$ running in time $n^{O(n/r^{3/2})}$. To obtain this algorithm, we use an algorithm we describe in Section 3.5 that finds a constant factor approximation in time exponential in the size of a given fvs. Finally, we present in Section 3.6 a matching lower bound: for any $\varepsilon > 0$ and any sufficiently large $r$, there is no $r$-approximation algorithm for Max Min Feedback Vertex Set running in time $n^{(n/r^{3/2})^{1-\varepsilon}}$ under the randomized ETH. To derive this super-polynomial inapproximability for our considered problem Max Min Feedback Vertex Set, we make a reduction from Max Independent Set, and we use the inapproximability of Chalermsook et al. [CLN13] for this problem and some Chernoff bounds, as we have described in Section 1.5.

In the next section, we begin by presenting the improved **NP**-hardness of Max Min Feedback Vertex Set.

## 3.1 Improved NP-Hardness

Let us present our **NP**-hardness of Max Min Feedback Vertex Set on planar bipartite graphs of maximum degree 6, which improves on the **NP**-hardness with $\Delta \geq 9$ of Mishra and Sikdar [MS01].

To obtain this improved **NP**-hardness, we make a reduction from Max Min Vertex Cover, for which there exists an **NP**-hardness on planar bipartite graphs of maximum degree 3 [ZZ95], to Max Min Feedback Vertex Set. We recall the mandatory private structure of the Max Min Vertex Cover problem, that we have extensively presented in the Introduction: every vertex $u$ in the solution has at least one private edge incident on itself. To derive the **NP**-hardness of Max Min Vertex Cover to our considered problem Max Min Feedback Vertex Set, for which the mandatory private structure is that every vertex $u$ in the solution has at least one private cycle, we naturally create a cycle for each edge of the original graph, in order to have a one-to-one correspondence between the vertices taken in the maximum minimal vertex cover and the ones taken in the maximum minimal feedback vertex set. Nonetheless, we also need to add some extra vertices and cycles in order to prove our **NP**-hardness.

**Theorem 3.1.** *Max Min Feedback Vertex Set is **NP**-hard on planar bipartite graphs with $\Delta = 6$.*

*Proof.* We give a reduction from Max Min Vertex Cover, which is **NP**-hard on planar bipartite graphs of maximum degree 3 [ZZ95]. Note that the **NP**-hardness in [ZZ95] is stated for Min Independent Dominating Set, but any independent dominating set is also a maximal independent set (and vice-versa) and the complement of a minimum maximal independent set of any graph is a maximum minimal vertex cover. Thus, we also obtain an **NP**-hardness for Max Min Vertex Cover on the same instances.

We are given a graph $G = (V, E)$ of Max Min Vertex Cover and we construct a graph $G = (V', E')$ of Max Min Feedback Vertex Set in the following way (see Figure 3.1): we keep the graph $G$; and for every edge $e = (u, v) \in E$, we add a path of length three from $u$ to $v$ going through two new vertices $e^1, e^2$; and for $i \in \{1, 2\}$, we add two cycles of length 4, $e^i, c_1^i, c_2^i, c_3^i, e^i$ and $e^i, c_4^i, c_5^i, c_6^i, e^i$. Note that $u, e^1, e^2, v, u$ forms a cycle of length 4. Because $\Delta(G) = 3$, we have $\Delta(G') = 6$. Moreover, since $G$ is planar and bipartite, $G'$ is also planar and bipartite. We will show that there is a minimal vertex cover of size at least $k$ in $G$ if and only if there is a minimal fvs of size at least $k + 4|E|$ in $G'$.

For the first direction, and given a minimal vertex cover $C$ of size at least $k$ in $G$, we construct the set $S = C \cup \bigcup_{e \in E} \{c_1^1, c_4^1, c_1^2, c_4^2\}$. We have

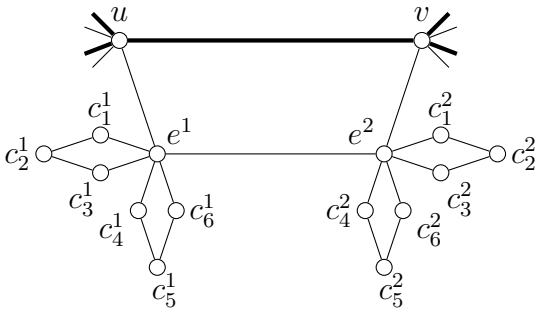


Figure 3.1: Edge gadget of $e = (u, v)$ in the constructed graph $G'$. Thick edges are originally in $G$.

$|S| \geq k+4|E|$. Let us first argue that $S$ is an fvs of $G'$. First, since $G'[V \setminus C]$ is an independent set, it is also acyclic. So if there is a cycle in $G'[V' \setminus S]$, then it is going through at least one newly added vertices. For each edge $e = (u, v) \in E$, we have at least one of $u$ and $v$ in $C$, so without loss of generality, let $u \in C$. Now, in $G'[V' \setminus S]$, the edges $(e^1, e^2)$ and $(e^2, v)$ are bridges, and therefore cannot be part of any cycle. The remaining cycles going through $e^1$ and $e^2$ are handled by the vertices $c_1^1, c_4^1, c_1^2$ and $c_4^2$. To see that $S$ is minimal, we remark first that since $C$ is a minimal vertex cover of $G$, for each $u \in C$ there exists $v \notin C$ with $e = (u, v) \in E$, and thus $u \in S$ has a private cycle, namely the cycle $u, e^1, e^2, v, u$. Now, we remark that for each $c_1^i, c_4^i$, for $i \in \{1, 2\}$, there is a private cycle, namely the cycle going through $e^i$.

For the other direction, suppose we are given a minimal fvs $S$ of $G'$ with $|S| \geq k + 4|E|$. We will edit $S$ so that is contains only vertices in $V' \setminus \bigcup_{e \in E} \{e^1, e^2\}$, without decreasing its size.

First, suppose that $e^1, e^2 \in S$ for some $e \in E$. We construct a new minimal fvs $S' = (S \setminus \{e^2\}) \cup \{c_1^2, c_4^2\}$. The set $S'$ is larger than $S$ since by the minimality of $S$ we have that $c_j^2 \notin S$ for $j \in \{1, \ldots, 6\}$. Moreover, $S'$ is an fvs, since the cycles dominated by $e^2$ in $S$ were the cycles $u, e^1, e^2, v, u$, which is dominated by $e^1$, and the two cycles $e^2, c_1^2, c_2^2, c_3^2, e^2$ and $e^2, c_4^2, c_5^2, c_6^2, e^2$, which are dominated by $c_1^2$ and $c_4^2$, respectively. And the two added vertices $c_1^2$ and $c_4^2$ both have a private cycle, namely the only cycle to which they belong. Thus, in the remainder, we assume that $S$ contains at most one of $e^1$ and $e^2$ for all $e \in E$.

Suppose now that for some $e = (u, v) \in E$, we have $S \cap \{u, v\} \neq \emptyset$ and $S \cap \{e^1, e^2\} \neq \emptyset$. Without loss of generality, let $e^1 \in S$. We construct a new set $S' = (S \setminus \{e^1\}) \cup \{c_1^1, c_4^1\}$. The set $S'$ is larger than $S$ since by the minimality of $S$ we have that $c_j^1 \notin S$ for $j \in \{1, \ldots, 6\}$. Moreover, $S'$ is

an fvs, since the cycle $u, e^1, e^2, v, u$ is already dominated by $S \cap \{u, v\}$, and the two cycles $e^1, c_1^1, c_2^1, c_3^1, e^1$ and $e^1, c_4^1, c_5^1, c_6^1, e^1$ are dominated by $c_1^1$ and $c_4^1$, respectively. And the two added vertices $c_1^2$ and $c_4^2$ both have a private cycle, namely the only cycle to which they belong. Thus, in the remainder, we assume that if for some $e = (u, v) \in E$, we have $S \cap \{e^1, e^2\} \neq \emptyset$, then $u, v \notin S$.

Now, suppose that for some $e = (u, v) \in E$, we have $u, v \notin S$ and, without loss of generality, $e^1 \in S$. We construct a new minimal fvs $S' = (S \setminus \{e^1\}) \cup \{u, c_1^1, c_4^1\}$. Note first that $|S'| \geq |S| + 2$. The set $S'$ is an fvs, since the cycles dominated by $e^1$ in $S$ were the cycle $u, e^1, e^2, v, u$, which is dominated by $u$, and the two cycles $e^1, c_1^1, c_2^1, c_3^1, e^1$ and $e^1, c_4^1, c_5^1, c_6^1, e^1$, which are dominated by $c_1^1$ and $c_4^1$, respectively. Nonetheless, the set $S'$ is not necessarily minimal. So we greedily remove vertices from $S'$ to obtain a minimal fvs $S^*$. We claim that in this process we cannot remove more than two vertices. To see this, we first note that $u, c_1^1$ and $c_4^1$ cannot be removed from $S'$ since they all have a private cycle, namely the cycle $u, e^1, e^2, v, u$, $e^1, c_1^1, c_2^1, c_3^1, e^1$ and $e^1, c_4^1, c_5^1, c_6^1, e^1$, respectively. Suppose now that $w_1 \in S' \setminus S^*$ is the first vertex to be removed from $S'$. Note that, since $e^1$ is the vertex removed from $S$ to obtain $S'$, it follows that the redundant vertices of $S'$ belong to cycles $u, e^{1\prime}, e^{2\prime}, v', u$ for $(u, v') \in E$. Moreover, $w_1$ cannot be a vertex $e^{1\prime}$ or $e^{2\prime}$ because these vertices already have a private cycle, going through the vertices $c_1^{i\prime}, c_2^{i\prime}, c_3^{i\prime}$ and $c_4^{i\prime}, c_5^{i\prime}, c_6^{i\prime}$. So $w_1 = v'$. Now, let $w_2 \in S' \setminus S^*$ be the second vertex to be removed from $S'$. This vertex $w_2$ cannot belong to the cycle $u, e^{1\prime}, e^{2\prime}, u, v'$, because since $w_1 = v'$ it must be $e^{1\prime}$ or $e^{2\prime}$ and these two vertices have a private cycle. So $w_2$ belong to another cycle $u, e^{1\prime\prime}, e^{2\prime\prime}, v'', u$ for $(u, v'') \in E$, and by the same argument it must be the case that $w_2 = v''$. Since $|N(u) \setminus \{v\}| \leq 2$, it cannot be the case that a third vertex is removed from $S'$. As a result, at most two vertices of $S'$ are removed in the process to obtain the minimal fvs $S^*$, and, since $|S'| \geq |S| + 2$, we obtain another minimal fvs of the same size. Thus, in the remainder, we assume that $S$ does not contain $e^1, e^2$ for any $e \in E$.

Now, given a minimal fvs $S$ of $G'$ of size at least $k + 4|E|$ and with $S \cap (\bigcup_{e \in E} \{e^1, e^2\}) = \emptyset$, we construct a minimal vertex cover $C = S \cap V$. First, observe that $C$ is a vertex cover, since, for each edge $e = (u, v) \in E$, if $u, v \notin S$ then we would have the cycle $u, e^1, e^2, v, u$. Moreover, suppose that $C$ is not minimal, that is suppose there exists a vertex $u \in V$ such that $N_G[u] \subseteq C \subseteq S$. In that case, the vertex $u \in S$ has no private cycle since all the cycles $u, e^1, e^2, v, u$ with $(u, v) \in E$ are also dominated by $v \in S$. So we obtain a contradiction, and indeed $u$ has a private edge $(u, v)$ in $G$. Finally, we argue that $|S \setminus V| \leq 4|E|$. Consider an edge $e = (u, v) \in E$. The set $S$ cannot contain more than four vertices among the vertices $c_j^i$, for $i \in \{1, 2\}$

and $j \in \{1, \ldots, 6\}$, since otherwise a vertex of $S$ would not have a private cycle. So indeed $|S \setminus V| \leq 4|E|$ and so $|C| = |S \cap V| \geq k$.

□

Now that we have proven an improved **NP**-hardness from $\Delta \geq 9$ [MS01] to $\Delta \geq 6$ on planar bipartite graphs, we will in the following section present the few reduction rules we will use later in our polynomial-time approximation algorithm.

## 3.2 Reduction Rules and Combinatorial Tools

We begin by making two basic observations about our problem: deleting vertices or contracting edges can only decrease the size of the optimal solution. We let mmfvs($G$) be the size of the largest minimal fvs of $G$.

**Lemma 3.2.** *Let $G = (V, E)$ be a graph and $u \in V$. Then,* mmfvs($G$) $\geq$ mmfvs($G-u$). *Furthermore, given any minimal feedback vertex set $S$ of $G-u$, it is possible to construct in polynomial time a minimal feedback vertex set of $G$ of the same size or larger.*

*Proof.* Let $S$ be a minimal fvs of $G - u$. We observe that $S \cup \{u\}$ is an fvs of $G$. If $S \cup \{u\}$ is minimal, we are done. If not, we delete vertices from it until it becomes minimal. We now note that the only vertex which may be deleted in this process is $u$, since all vertices of $S$ have a private cycle in $G - u$ and thus in $G$. Hence, the resulting set is a superset of $S$. □

**Lemma 3.3.** *Let $G = (V, E)$ be a graph, $u, v \in V$ with $N(u) \cap N(v) = \emptyset$ and $(u, v) \in E$. Then* mmfvs($G$) $\geq$ mmfvs($G/uv$). *Furthermore, given any minimal feedback vertex set $S$ of $G/uv$, it is possible to construct in polynomial time a minimal feedback vertex set of $G$ of the same size or larger.*

*Proof.* Before we prove the lemma we note that the contraction operation, under the condition that $N(u) \cap N(v) = \emptyset$, preserves acyclicity in a strong sense: $G$ is acyclic if and only if $G/uv$ is acyclic. Indeed, if we contract an edge that is part of a cycle, this cycle must have length at least 4, and will therefore give a cycle in $G/uv$. Of course, contractions never create cycles in acyclic graphs.

Let $G' = G/uv$ and $w$ be the vertex of $G'$ which has replaced $u, v$. Let $V' = V(G')$, and $S$ be a minimal fvs of $G'$. We have two cases: $w \in S$ or $w \notin S$.

In the first case $w \in S$, we start with the set $S' = (S \setminus \{w\}) \cup \{u, v\}$. $S'$ is an fvs of $G$. Furthermore, no vertex of $S' \setminus \{u, v\}$ is redundant: for all

$z \in S \setminus \{w\}$, there is a cycle in $G'[(V' \setminus S) \cup \{z\}]$, therefore there is also a cycle in $G[(V \setminus S') \cup \{z\}]$. Furthermore, we claim that $S' \setminus \{u, v\}$ is not a valid fvs. Indeed, there must be a cycle contained (due to minimality) in $G_1 = G'[(V' \setminus S) \cup \{w\}]$. Therefore, if there is no cycle in $G_2 = G[(V \setminus S') \cup \{u, v\}]$, we get a contradiction, as $G_1$ can be obtained by $G_2$ by contracting the edge $(u, v)$ and contracting edges preserves acyclicity. We conclude that, even if $S'$ is not minimal, if we remove vertices until it becomes minimal, we will remove at most one vertex, so the size of the fvs obtained is at least $|S|$.

In the second case $w \notin S$, we will return the same set $S$. Let $F = V \setminus S$ and $F' = V' \setminus S$. By definition, $G'[F']$ is acyclic. To see that $G[F]$ is also a forest, we note that $G'[F']$ is obtained from $G[F]$ by contracting $(u, v)$, and as we noted in the beginning, the contractions we use strongly preserve acyclicity. To see that $S$ is minimal, take $z \in S$ and consider the graphs $G_1 = G[(V \setminus S) \cup \{z\}]$ and $G_2 = G'[(V' \setminus S) \cup \{z\}]$. We see that $G_2$ can be obtained from $G_1$ by contracting $(u, v)$. But $G_2$ must have a cycle, by the minimality of $S$, so $G_1$ also has a cycle. Thus, $S$ is minimal in $G$. □

We now show two *safe* versions of Lemmas 3.2 and 3.3.

**Lemma 3.4.** *Let $G, u$ be as in Lemma 3.2 with $d(u) \leq 1$. Then* $\text{mmfvs}(G - u) = \text{mmfvs}(G)$.

*Proof.* We only need to show that $\text{mmfvs}(G) \leq \text{mmfvs}(G - u)$, since the other direction is given by Lemma 3.2. Let $S$ be a minimal fvs of $G$. Then, $S$ is an fvs of $G - u$. Furthermore, $u \notin S$, as $S$ is minimal in $G$. To see that $S$ is also minimal in $G - u$, note that any cycle of $G$ also exists in $G - u$, since no cycle contains $u$. □

**Lemma 3.5.** *Let $G, u, v$ be as in Lemma 3.3 with $d(u) = d(v) = 2$. Then* $\text{mmfvs}(G/uv) = \text{mmfvs}(G)$.

*Proof.* Let $G' = G/uv$, $w$ be the vertex that replaced $u, v$ in $G'$, and $V' = V(G')$.

We only need to show that $\text{mmfvs}(G) \leq \text{mmfvs}(G')$, since the other direction is given by Lemma 3.3. Let $S$ be a minimal fvs of $G$. We consider two cases:

If $u, v \notin S$, then we claim that $S$ is also a minimal fvs of $G'$. Indeed, $G'[V' \setminus S]$ is obtained from $G[V \setminus S]$ by contracting $(u, v)$, so both are acyclic. Furthermore, for all $z \in S$, $G'[(V' \setminus S) \cup \{z\}]$ is obtained from $G[(V \setminus S) \cup \{z\}]$ by contracting $(u, v)$, therefore both have a cycle, hence no vertex of $S$ is redundant in $G'$.

If $\{u, v\} \cap S \neq \emptyset$, we claim that exactly one of $u$ and $v$ is in $S$. Indeed, if $u, v \in S$, then $G[(V \setminus S) \cup \{u\}]$ does not contain a cycle going through $u$, as

$u$ has degree 1 in this graph. Without loss of generality, let $u \in S$ and $v \notin S$. We set $S' = (S \setminus \{u\}) \cup \{w\}$ and claim that $S'$ is a minimal fvs of $G'$. $S'$ is an fvs of $G'$, since it corresponds to deleting $S \cup \{v\}$ from $G$. To see that it is minimal, for all $z \in S' \setminus \{w\}$ we observe that $G'[(V' \setminus S') \cup \{z\}]$ is obtained from $G'[(V \setminus S) \cup \{z\}]$ by deleting $v$, which has degree 1. Therefore, this deletion strongly preserves acyclicity and $z$ has a private cycle in $G'$. Finally, to see that $w$ is not redundant for $S'$ we observe that $G[(V \setminus S) \cup \{u\}]$ has a cycle, and a corresponding cycle must be present in $G'[(V' \setminus S') \cup \{w\}]$, which is obtained from the former graph by contracting $(u, v)$. □

We now give a definition, which defines the notion of *reduced* graph concerning the Max Min Feedback Vertex Set problem. Informally, a reduced graph is a graph that has been preprocessed in such a way that our approximation algorithm we will present in the next section always constructs on it a minimal fvs that has size at least $\Omega(n^{1/3})$, thus giving us our $O(n^{2/3})$-approximation. Moreover, as we have mentioned in Section 1.3 and at the beginning of this chapter, a reduced graph is a graph which has been preprocessed by applying Lemmas 3.4 and 3.5 exhaustively for Max Min Feedback Vertex Set, whereas the preprocessing step for Max Min Vertex Cover we have presented in Section 1.3 is to simply remove isolated vertices.

**Definition 3.6.** *For a graph $G = (V, E)$ we say that $G$ is reduced if Lemma 3.4 and Lemma 3.5 have been exhaustively applied.*

We now present a counting argument which will be useful in our algorithm, and states, roughly, that if in a reduced graph we find a (not necessarily minimal) fvs, that fvs must have many neighbors in the corresponding forest.

**Lemma 3.7.** *Let $G = (V, E)$ be a reduced graph and $S \subseteq V$ a feedback vertex set of $G$. Let $F = V \setminus S$. Then, $|N(S) \cap F| \geq \frac{|F|}{4}$.*

*Proof.* Let $n_0$ be the number of isolated vertices in $G[F]$, $n_1$ be the number of leaves of $F$, $n_3$ the number of vertices of $F$ with at least three neighbors in $F$, $n_{2a}$ the number of vertices of $F$ with two neighbors in $F$ and at least one neighbor in $S$, and $n_{2b}$ the number of remaining vertices of $F$, that is the number of vertices of $F$ with two neighbors in $F$ and no neighbor in $S$. We have $n_0 + n_1 + n_{2a} + n_{2b} + n_3 = |F|$. Furthermore, $n_3 \leq n_1$ because the average degree of any forest is less than 2.

We observe that all leaves of the tree have a neighbor in $S$, because otherwise we would have applied Lemma 3.4. And all isolated vertices in

$G[F]$ have a neighbor in $S$ since $G$ is reduced and thus does not have any isolated vertex. This gives $|N(S) \cap F| \geq n_0 + n_1 + n_{2a}$.

Furthermore, none of the $n_{2b}$ vertices which have degree two in the tree and no neighbors in $S$ can be adjacent to each other, since then Lemma 3.5 would apply. Therefore, $n_{2b} \leq n_1 + n_{2a} + n_3$. Indeed, if $n_{2b} > n_1 + n_{2a} + n_3$, then $n_{2b} > |F|/2$ and, since these $n_{2b}$ vertices form an independent set, we would have $|E(F)| \geq 2n_{2b} > |F|$, contradicting the assumption that $F$ is a forest.

Putting things together we get $|F| = n_0 + n_1 + n_{2a} + n_{2b} + n_3 \leq 2n_0 + 2n_1 + 2n_{2a} + 2n_3 \leq 2n_0 + 4n_1 + 2n_{2a} \leq 4|N(S) \cap F|$. □

We note that Lemma 3.7 immediately gives an approximation algorithm with ratio $O(\Delta)$.

**Lemma 3.8.** *In a reduced graph $G$ with $n$ vertices and maximum degree $\Delta \geq 1$, every feedback vertex set has size at least $\frac{n}{5\Delta}$.*

*Proof.* Let $S$ be a feedback vertex set of $G$ and $F$ the corresponding forest. If $|S| < \frac{n}{5\Delta}$ then $|N(S) \cap F| < \frac{n}{5}$ since the maximum degree is $\Delta$. So by Lemma 3.7, we have $|F| < \frac{4n}{5}$. But then $|V| = |S| + |F| < n$, which is a contradiction. □

**Remark 3.9.** *Lemma 3.7 is tight.*

*Proof.* Take two copies of a complete rooted binary tree with $n$ leaves and connect their roots. The resulting tree has $2n$ leaves and $2n - 2$ vertices of degree 3. Subdivide every edge of this tree once. Add two vertices $u, v$ connected to every leaf. In the resulting graph $S = \{u, v\}$ is an fvs. The corresponding forest has $8n-5$ vertices. Indeed, we have: $2n-3$ new vertices obtained from the subdivisions between the degree-3 vertices; $2n-2$ vertices of degree 3; and $2(2n)$ leaves and their adjacent new vertices. And we have $2n$ vertices connected to $S$. The graph is reduced. □

We finish this section with a final intermediate Lemma that allows us to construct a large minimal fvs in any reduced graph that is a forest plus one vertex.

**Lemma 3.10.** *Let $G = (V, E)$ be a reduced graph and $u \in V$ such that $G - u$ is acyclic. Then it is possible to construct in polynomial time a minimal feedback vertex set $S$ of $G$ with $|S| \geq d(u)/2$.*

*Proof.* Let $F = V \setminus \{u\}$. Since the graph is reduced, all trees of $G[F]$ contain at least two neighbors of $u$. Indeed, every tree $T$ of $G[F]$ contains at least two vertices, because otherwise Lemma 3.4 would apply. Thus every tree $T$

contains at least two leaves, and all leaves must be neighbors of $u$, because otherwise Lemma 3.4 would apply.

Now, we edit the graph. As long as there exist $v, w \in F$ with $(v, w) \in E$ and $\{v, w\} \not\subseteq N(u)$, we contract the edge $(v, w)$. Note that we can apply Lemma 3.3 since $v$ and $w$ do not have any common neighbors ($u$ is not a common neighbor by assumption, and they cannot have a common neighbor in the forest without forming a cycle). Furthermore, this operation does not change $d(u)$, since for two neighbors $v, w$ in $F$ neighbors of $u$, the edge $(v, w)$ is not contracted. Therefore, it will be sufficient to construct a minimal fvs in the resulting graph after applying this operation exhaustively, by Lemma 3.3.

Suppose now that we have applied this operation exhaustively. We eventually arrive at a graph where $u$ is connected to all vertices of $F$, since every tree of $F$ initially contains at least two neighbors of $u$, since all the non-neighbors of $u$ are absorbed by the contraction operation (each contraction decreases $|F \setminus N(u)|$), and since neighbors of $u$ in $F$ are never absorbed. Therefore, we arrive at a graph with $d(u) = |F'|$ for the new forest $F'$. And every tree of $F'$ contains at least two vertices.

Now, since $G[F']$ is a forest, it is bipartite, so there is a bipartition $F' = L \cup R$. Without loss of generality, $|L| \leq |R|$. We return the set $S = R$. First, $S$ does have the promised size, since $|S| \geq |F'|/2 = d(u)/2$. Second, $S$ is an fvs, as $L$ is an independent set and hence $L \cup \{u\}$ is a star. Finally, $S$ is minimal, because every $v \in S$ is connected to $u$, and also has at least one neighbor $w \in L$ which is also connected to $u$. □

Now that we have presented the two rules that are applied to obtain a reduced graph, and two intermediate lemmas (Lemmas 3.7 and 3.10) which we will use in our polynomial-time approximation, we give in the following section our approximation algorithm.

## 3.3 Polynomial-Time Approximation

In this section we present our polynomial-time algorithm which guarantees an approximation ratio of $O(n^{2/3})$. As we will show later in the next section, this ratio is the best that can be hoped for in polynomial time.

On a high level, our algorithm proceeds as follows: first we apply the reduction rules of Lemmas 3.4 and 3.5 where the value of the optimal is guaranteed to stay constant to obtain a reduced graph; then we compute a minimal fvs $S$ in an arbitrary way. If $S$ is large enough (larger than $n^{1/3}$), we simply return this set.

If not, we apply some counting arguments (by Lemma 3.7) to show that a vertex $u \in S$ with high degree ($\geq n^{2/3}$) must exist. We then have two cases: either we are able to construct a large minimal fvs just by looking at the neighborhood of $u$ in the forest (and ignoring $S \setminus \{u\}$) (by Lemma 3.10), or $u$ must share many neighbors with another vertex $v \in S$, in which case we construct a large minimal fvs in the common neighborhood of $u$ and $v$.

Because our algorithm is constructive (and runs in polynomial time), we find it interesting to remark an interpretation from the point of view of extremal combinatorics, given in Corollary 3.12.

We can now present the main theorem of this section:

**Theorem 3.11.** *There is a polynomial time approximation algorithm for* Max Min Feedback Vertex Set *with ratio* $O(n^{2/3})$.

*Proof.* We are given a graph $G = (V, E)$. We begin by applying Lemmas 3.4 and 3.5 exhaustively in order to obtain a reduced graph $G' = (V', E')$. Clearly, if we obtain a $|V'|^{1/3}$ solution in $G'$, since the transformations applied do not change the optimal, and since we can construct a solution of the same size in $G$ by Lemmas 3.2, 3.3, 3.4 and 3.5, we get a $|V'|^{2/3} \leq |V|^{2/3}$ approximation ratio in $G$. So, in the remainder, to ease presentation, we assume that $G$ is already reduced and has $n$ vertices.

Our algorithm begins with an arbitrary minimal fvs $S$. It can be constructed, for example, by starting with $S = V$, and by removing vertices from $S$ until it becomes minimal. If $|S| \geq n^{1/3}$, then we return $S$. Since the optimal solution cannot have size more than $n$, we already have a $n^{2/3}$-approximation.

So suppose that $|S| < n^{1/3}$. Let $F$ be the corresponding forest. We have $|F| > n - n^{1/3} > n/2$ for $n$ sufficiently large. By Lemma 3.7, $|N(S) \cap F| \geq n/8$. Since $|S| < n^{1/3}$, there must exist a vertex $u \in S$ with at least $\frac{|N(S) \cap F|}{|S|} > \frac{n^{2/3}}{8}$ neighbors in $F$.

Now, let $w \in F \cap N(u)$. We say that $w$ is a *good* neighbor of $u$ if there exists another vertex $w' \in F \cap N(u)$ with $w' \neq w$ and $w'$ is in the same tree of $G[F]$ as $w$. Otherwise, we say that $w$ is a *bad* neighbor of $u$. By extension, a tree of $G[F]$ that contains a good (resp. bad) neighbor of $u$ will be called *good* (resp. *bad*). Note that every vertex of $N(u) \cap F$ is either good or bad.

Recall that $|N(u) \cap F| \geq \frac{n^{2/3}}{8}$. We distinguish between the following two cases: either $u$ has at least $\frac{n^{2/3}}{16}$ good neighbors in $F$, or it has at least that many bad neighbors in $F$.

In the former case, we delete from the graph the set $S \setminus \{u\}$, and apply Lemmas 3.4 and 3.5 exhaustively again. We claim that the number of good neighbors of $u$ does not decrease in this process. Indeed, two good neighbors

of $u$ cannot be contracted using Lemma 3.5, since they have a common neighbor, namely $u$. Furthermore, suppose $w$ is the first good neighbor of $u$ to be deleted using Lemma 3.4. This would mean that $w$ currently has no other neighbor except $u$. However, since $w$ is good, there initially was a vertex $w' \in N(u)$ in the same tree of $G[F]$ as $w$. And since $w'$ has not been deleted yet, since we assumed that $w$ was the first to be deleted, and since Lemmas 3.4 and 3.5 cannot disconnect two vertices which are in the same component, we obtain that the vertex $w$ cannot be removed by Lemma 3.4. Thus, we have a reduced graph, where $\{u\}$ is an fvs, and with $d(u) \geq \frac{n^{2/3}}{16}$. So, by Lemma 3.10, we obtain a minimal fvs of size at least $\frac{n^{2/3}}{32}$, which is an $O(n^{1/3})$-approximation.

In the latter case, $u$ has at least $\frac{n^{2/3}}{16}$ bad neighbors in $F$. Consider such a bad tree $T$. The tree $T$ must have a neighbor in $S \setminus \{u\}$. Indeed, if $|T| = 1$, then the vertex in $T$ must have another neighbor in $S$, because otherwise it should have been deleted by Lemma 3.4. And if $|T| \geq 2$, then one vertex is a neighbor of $u$ and at least one leaf is connected to $S$, because otherwise this leaf should have been deleted by Lemma 3.4. Furthermore, since $u$ is connected to one vertex in each bad tree, $u$ is connected to at least $\frac{n^{2/3}}{16}$ bad trees. We now find a vertex $v \in S \setminus \{u\}$ such that $v$ is connected to the maximum number of bad trees connected to $u$. Since $|S| < n^{1/3}$, $v$ must be connected to at least $\frac{n^{2/3}}{16|S|} \geq \frac{n^{1/3}}{16}$ bad trees connected to $u$.

Now, we delete from the graph the set $S \setminus \{u, v\}$ as well as all trees of $G[F]$, except the bad trees connected to $u$ and $v$. Consider such a bad tree $T$ connected to both $u$ and $v$, and let $u' \in T \cap N(u)$ and $v' \in T \cap N(v)$ such that $u'$ and $v'$ are as close as possible in $T$ (note that perhaps $u' = v'$). We delete all vertices of the tree $T$ except those on the path from $u'$ to $v'$, and then we contract all internal edges of this path (note that internal vertices of this path are not connected to $u$ and $v$ by the selection of $u', v'$). By Lemma 3.2 and 3.3, if we are able to produce a large minimal fvs in the resulting graph, we obtain a solution of the same size or larger for $G$. We have that in the resulting graph, every bad tree $T$ connected to $u$ and $v$ has been reduced to a single vertex connected to $u$ and $v$. So the graph is either a $K_{2,s}$ with $s \geq \frac{n^{1/3}}{16}$, or a $K_{2,s}$ with the addition of the edge $(u, v)$. In either case, by starting with the fvs that contains all vertices except $u$ and $v$, and making it minimal, we obtain a solution of size at least $s - 1$, which gives a $O(n^{2/3})$-approximation. □

Now that we have proved our $O(n^{2/3})$-approximation algorithm for Max Min Feedback Vertex Set, we show that this algorithm always creates a minimal fvs of size at least $\Omega(n^{1/3})$ if the graph is reduced, that is if it has been properly preprocessed. Note the difference with Max Min Vertex

Cover, for which there always exists a $\Omega(n^{1/2})$ minimal vertex cover in a graph which does not contain any isolated vertices, as we have shown in Section 1.3.

**Corollary 3.12.** *For any reduced graph $G$ on $n$ vertices we have* $\text{mmfvs}(G) = \Omega(n^{1/3})$.

*Proof.* We simply note that the algorithm of Theorem 3.11 always constructs a solution of size at least $\frac{n^{1/3}}{c}$, where $c$ is a constant, assuming that the original $n$-vertex graph $G$ was reduced. □

**Remark 3.13.** *Corollary 3.12 is tight.*

*Proof.* Take a $K_n$ and, for every pair of vertices $u, v$ in the clique, add $2n$ new vertices connected only to $u$ and $v$. The graph has order $n + 2n\binom{n}{2} = n + n^2(n-1) = n^3 - n^2 + n \geq n^3/2$ for $n$ sufficiently large. Any minimal fvs of this graph must contain at least $n - 2$ vertices of the clique. As a result its maximum size is at most $n - 2 + 2n \leq 3n$. We have $\text{mmfvs}(G) \leq 3n$ so $\text{mmfvs}(G) = O(|V(G)|^{1/3})$. □

We briefly mention that Theorem 3.11 also implies the existence of a cubic kernel of Max Min Feedback Vertex Set when parameterized by the solution size $k$. We did not mention the notion of *kernel* in the Introduction since it is the only place where we use it, but a kernel is formally defined through the notion of *kernelization*. A kernelization for a decision problem $\Pi$ is an algorithm that takes an instance $(I, k)$ and maps it in polynomial time in $|I|$ and $k$ to another instance $(I', k')$ such that: $I$ is a Yes-instance if and only if $I'$ is a Yes-instance; $|I'| \leq f(k)$ for a function $k$; and $k' \leq g(k)$ for a function $g$. If these conditions are satisfied, the instance $(I', k')$ is called a kernel.

Recall that the reduction rules do not change the solution size. We suppose that the reduced graph has $n$ vertices. For a constant $c$, if $n \geq c^3k^3$, then we can always produce a solution of size at least $n^{1/3}/c = k$, and thus the answer is Yes. Otherwise, we have a cubic kernel.

**Corollary 3.14.** *Max Min Feedback Vertex Set admits a cubic kernel when parameterized by the solution size.*

Finally, we remark that a similar combinatorial point of view can be taken for the related problem of Max Min Vertex Cover, giving another intuitive explanation for the difference in approximability between the two problems.

**Remark 3.15.** *Any graph $G = (V, E)$ with $n$ vertices and without isolated vertices has a minimal vertex cover of size at least $\sqrt{n}$, and this is asymptotically tight.*

*Proof.* We have shown that a graph without isolated vertices always has a minimal vertex cover of size at least $\sqrt{n}$ in Section 1.3.

To see that the bound given is tight, take a $K_n$ and attach $n$ leaves to each of its vertices. This graph has $n^2$ vertices, but any minimal vertex cover has size at most $(n-1)+n = 2n-1$. □

Now that we have described our $O(n^{2/3})$-approximation algorithm for Max Min Feedback Vertex Set, we will in the following section prove that this algorithm is asymptotically the best we could hope.

## 3.4 Matching Polynomial-Time Inapproximability

We begin by showing that the best approximation ratio achievable in polynomial time is indeed (essentially) $n^{2/3}$. For this, we rely on the celebrated result of Håstad [Hås96] on the hardness of approximating Max Independent Set, which was later derandomized by Zuckerman [Zuc05], that we have presented in Section 1.3.

**Theorem 3.16** ( [Hås99,Zuc05])**.** *For any $\epsilon > 0$, there is no polynomial time algorithm which approximates Max Independent Set with a ratio of $n^{1-\epsilon}$ if $\boldsymbol{P \neq NP}$.*

Starting from this result, we present a reduction to Max Min Feedback Vertex Set. Note that our reduction is similar to the reduction of Boria et al. [BCP13] we have presented in Section 1.3 from Max Independent Set to Max Min Vertex Cover which gave the $n^{1/2-\varepsilon}$-inapproximability for the latter problem.

**Theorem 3.17.** *For any $\epsilon > 0$, Max Min Feedback Vertex Set is inapproximable within a factor of $n^{2/3-\epsilon}$ if $\boldsymbol{P \neq NP}$.*

*Proof.* We give a gap-preserving reduction from Max Independent Set, which cannot be approximated within a factor of $n^{1-\epsilon}$ under $\mathbf{P} \neq \mathbf{NP}$ [Zuc05]. We are given a graph $G = (V, E)$ on $n$ vertices as an instance of Max Independent Set. Recall that $\alpha(G)$ denotes the size of the maximum independent set of $G$.

We transform $G$ into an instance of Max Min Feedback Vertex Set as follows: we keep the graph $G$; and for every pair of $u, v \in V$, we add $n$ vertices such that they are adjacent only to $u$ and $v$. We denote by $I_{uv}$ the set of such vertices. Then $I_{uv}$ is an independent set. Let $G' = (V', E')$ be the constructed graph.

We now make the following two claims:

**Claim 3.18.** $\text{mmfvs}(G') \geq (n-1)\binom{\alpha(G)}{2}$

*Proof.* We construct a minimal fvs of $G'$ as follows: let $C$ be a minimum vertex cover of $G$. Then we begin with the set that contains $C$ and the union of all $I_{uv}$ (which is clearly an fvs) and remove vertices from it until it becomes minimal. Let $S$ be the final minimal fvs. We observe that for all $u, v \in V \setminus C$, $S$ contains at least $n-1$ of the vertices of $I_{uv}$. Since $C$ is a minimum vertex cover of $G$, there are $\binom{\alpha(G)}{2}$ pairs $u, v \in V \setminus C$. □

**Claim 3.19.** $\text{mmfvs}(G') \leq n\binom{2\alpha(G)}{2} + n$

*Proof.* Let $S$ be a minimal fvs of $G'$ and $F$ be the corresponding forest. It suffices to show that $|S \setminus V| \leq n\binom{2\alpha(G)}{2}$, since $|S \cap V| \leq n$. Consider now a set $I_{uv}$. If $u \in S$ or $v \in S$, then $I_{uv} \cap S = \emptyset$, because all vertices of $I_{uv}$ have at most one neighbor in $F$, and are therefore redundant. So, $I_{uv}$ contains (at most $n$) vertices of $S$ only if $u, v \in F$. However, $|F \cap V| \leq 2\alpha(G)$, because $F$ is bipartite and $F \cap V$ induces two independent sets, both of which must be at most equal to the maximum independent set of $G$. So the number of pairs $u, v \in F \cap V$ is at most $\binom{2\alpha(G)}{2}$ and, since each corresponding $I_{uv}$ has size $n$, we get the promised bound. □

The two claims together imply that there exist constants $c_1, c_2$ such that, for $n$ sufficiently large, we have $c_1 n(\alpha(G))^2 \leq \text{mmfvs}(G') \leq c_2 n(\alpha(G))^2$. That is, $\text{mmfvs}(G') = \Theta(n(\alpha(G))^2)$.

Suppose now that there exists a polynomial-time approximation algorithm which, given a graph $G'$, produces a minimal fvs $S$ with the property $\frac{\text{mmfvs}(G')}{r} \leq |S| \leq \text{mmfvs}(G')$, that is, there exists an $r$-approximation for Max Min Feedback Vertex Set. Running this algorithm on the instance we have constructed, we obtain that $\frac{c_1 n(\alpha(G))^2}{r} \leq |S| \leq c_2 n(\alpha(G))^2$. Therefore, $\frac{\alpha(G)}{\sqrt{rc_2/c_1}} \leq \sqrt{\frac{|S|}{c_2 n}} \leq \alpha(G)$. As a result, we obtain an $O(\sqrt{r})$-approximation for the value of $\alpha(G)$. We therefore conclude that, if $\mathbf{P} \neq \mathbf{NP}$, any such algorithm must have $\sqrt{r} > n^{1-\epsilon}$, for any $\epsilon > 0$, hence $r > n^{2-\epsilon}$ by choosing $\varepsilon$ correctly. Since the graph $G'$ has $N = \Theta(n^3)$ vertices, we get that no approximation algorithm can achieve a ratio of $N^{2/3-\epsilon}$ for Max Min Feedback Vertex Set. □

We notice that, in the construction of the previous theorem, the maximum degree of the graph is approximately equal to the approximation gap. Thus, the following corollary also holds.

**Corollary 3.20.** *For any positive constant $\epsilon$, Max Min Feedback Vertex Set is inapproximable within a factor of $\Delta^{1-\epsilon}$ if $\boldsymbol{P} \neq \boldsymbol{NP}$.*

Thus, we have completely settled the polynomial-time approximability of Max Min Feedback Vertex Set, and along the way we have proved that the problem also has an asymptotically optimal $O(\Delta)$ algorithm, that the problem admits a cubic kernel, and finally that this problem and Max Min Vertex Cover have some interesting combinatorial extremal solutions. In the next two sections, we focus on super-polynomial approximation.

## 3.5 Super-Polynomial Approximation

In this section we give an approximation algorithm that generalizes our $O(n^{2/3})$-approximation algorithm described in Section 3.3 and which is able to guarantee any desired performance, at the cost of increased running time. On a high level, our initial approach again constructs an arbitrary minimal fvs $S$ and, if $S$ is clearly large enough, returns it. However, things become more complicated from then on, as it is no longer sufficient to consider vertices of $S$ individually or in pairs. We therefore need several new ideas, one of which is given in the following lemma which states that we can find a constant factor approximation in time exponential in the size of a given fvs. This will be useful as we will use the assumption that $S$ is "small", and then cut it up into even smaller pieces to allow us to use this lemma.

**Lemma 3.21.** *Given a graph $G = (V, E)$ on $n$ vertices and a feedback vertex set $S_0 \subseteq V$ of size $k$, it is possible to produce a minimal fvs $S'$ of $G$ of size $|S'| \geq \frac{\text{mmfvs}(G)}{3}$ in time $n^{O(k)}$.*

Before we prove this lemma, let us point out that, for $k = 1$, Max Min Feedback Vertex Set can be solved optimally in time $O(n)$, using standard arguments from parameterized complexity. Indeed, in this case, the graph $G$ has treewidth 2, so by invoking Courcelle's Theorem and since the properties "$S$ is an fvs" and "$S$ is minimal" are MSO-expressible [CFK+15], we can solve the problem optimally in time $O(n)$. Unfortunately, this type of argument is not good enough for larger values of $k$, as the running time guaranteed by Courcelle's Theorem could depend super-exponentially on $k$, as we have mentioned in Section 1.4. We could try to avoid this by formulating a treewidth-based dynamic programming algorithm to obtain a better

running time, but we prefer to give a more direct branching algorithm, since this is good enough for the super-polynomial approximation algorithm we seek to design.

*Proof.* We will assume that $S_0$ is minimal, because otherwise we can remove vertices from it to make it minimal, and this only decreases the running time of our algorithm. As a result, we assume also that $\text{mmfvs}(G) \geq 3k$, as otherwise $S_0$ is already a 3-approximation.

Let $S^*$ be a maximum minimal fvs in $G$, and let $F^* = V \setminus S^*$. We formulate an algorithm that maintains two disjoint sets of vertices $S$ and $F$, which, intuitively, correspond to the vertices we have decided to place in the fvs or in the induced forest, respectively. We will denote $U = V \setminus (S \cup F)$ the set of "undecided" vertices. Our algorithm will sometimes "guess' some vertices of $U$ to be placed in $S$ or $F$, and we will upper-bound the guessing possibilities by $n^{O(k)}$.

Throughout the algorithm, we will work to maintain the following four invariants:

1. $S \cup F$ is an fvs of $G$;
2. $S \subseteq S^*$ and $F \subseteq F^*$;
3. $G[F]$ is acyclic and has at most $2k$ components;
4. All vertices of $S$ have at least two neighbors in $F$.

We begin by guessing a set $F_0 \subseteq S_0$ such that $G[F_0]$ is acyclic and we set $F = F_0$ and $S = S_0 \setminus F_0$. Property 1 is satisfied as $S \cup F = S_0$ is an fvs of $G$. Property 2 is satisfied for the right guess $F_0 = F \cap S_0$. Now, if there exists a vertex $u \in S$ which does not satisfy Property 4, we guess one or two vertices from $N(u) \cap U$ and place them into $F$, so that $u$ has indeed two neighbors in $F$. Since $u$ has a private cycle in $G[F^*]$, by assuming that we have rightfully guessed $F_0$, if the vertices we guessed are the neighbors of $u$ in that cycle, we maintain Property 2. We continue in this way until Property 4 is satisfied for all vertices of $S$. We now observe that $F$ is acyclic, since for the right guess we have $F \subseteq F^*$. And since we have added at most 2 vertices for each vertex of $S$, it follows that $F$ contains at most $2k$ vertices, hence at most $2k$ components, so Property 3 is satisfied also. So far, the total running time is upper-bounded by $2^k n^{2k}$: $2^k$ for guessing $F_0 \subseteq S_0$ and $n^{2k}$ for guessing at most two neighbors for every $u \in S$.

We now define the notion of "connector". Formally, a connector is a path $P$ with $V(P) \subseteq F^* \setminus F$ such that $G[F \cup P]$ has strictly fewer components than

$G[F]$. Our algorithm will now repeateadly guess if a connector exists, and if it does it will guess the first and last vertices $u$ and $v$ of $P$. Note that $u, v \in U$, and, if we rightfully guess $u$ and $v$, then we can infer all of $P$, since $G[U]$ is acyclic and there is at most one path from $u$ to $v$ in $G[U]$. Then, we set $F = F \cup P$, and we continue guessing until we guess that no connector exists. Note that guessing the two endpoints of a connector gives $n^2$ possibilities, and that adding a connector to $F$ decreases the number of connected components of $F$ by at least one, which can happen at most $2k$ times by Property 3. So the total running time of this procedure is upper-bounded by $n^{O(k)}$. We now show that the four properties are satisfied so far. Property 1 is satisfied since $S \cup F$ was already an fvs of $G$ before adding the connectors to $F$. Property 2 is satisfied since a connector $P$ verifies $V(P) \subseteq F^* \setminus F$. Property 3 is satisfied since adding a connector decreases the number of components by at least one. And Property 4 is satisfied since every vertex of $S$ already had two neighbors in $F$ before adding the connectors.

We now consider every vertex $u \in U$ that has at least two neighbors in $F$ and place all such vertices in $S$. Properties 1, 3 and 4 are still satisfied. Furthermore, if our guesses so far are correct, all such vertices $u$ belong to $S$. Indeed, they have at least two neighbors in $F$ for which there is a path between them in $F$, because otherwise they would function as connectors in $F$, and we assume that we have correctly guessed that no more connector exist. Thus, these vertices $u$ must be in $S^*$ in order to dominate the cycle that go through their two neighbors in $F$.

We are now in a situation where every vertex left in $U$ has at most one neighbor in $F$. We construct a new graph $H$ by deleting from $G$ all of $S$ and replacing $F$ by a single vertex $f$ that is connected to $N(F) \cap U$. Note that $H$ is a simple graph, i.e. it has no parallel edges, because otherwise a vertex of $U$ would have two neighbors in $F$, and we have put all these vertices of $U$ in $S$. Moreover, $H$ has an fvs of size 1, namely the set $\{f\}$. We therefore use the aforementioned algorithm implied by Courcelle's Theorem to produce a maximum minimal fvs of $H$, which, without loss of generality, does not contain $f$. Let $S_H \subseteq U$ be this fvs. In $G$, the set $S \cup S_H$ is an fvs. But it might be not minimal, so we remove vertices from it until it is minimal. Let $S'$ be this minimal fvs obtained.

We now prove that this solution $S'$ is a 3-approximation. To see that the resulting solution has the desired size, we focus on the case where all guesses were correct, and therefore where Properties 1-4 were maintained throughout the execution of the algorithm. As mentioned earlier, the total running time of this algorithm is $n^{O(k)}$.

We first observe that $\text{mmfvs}(H) \geq \text{mmfvs}(G) - |S|$. Indeed, the set $S_1 = S^* \setminus S$ is a minimal fvs of $H$. To see that $S_1$ is an fvs, suppose that

$H$ contains a cycle after deleting $S_1$. This cycle must necessarily go through $f$, since $G[U]$ is acyclic. Now, let $P$ be the vertices of this cycle except $f$. We have $P \subseteq U \setminus S^*$, so $P \subseteq F^*$. However, this means that either $P$ forms a cycle with a component of $F$, which contradicts the acyclicity of $F^*$ by Property 2, or $P$ is a connector, which contradicts our guess that no other connector exists. Therefore, we obtain a contradiction, and $S_1$ must be an fvs of $H$. To see that it is minimal, we note that for every $u \in S_1$, there is a private cycle in $G[U \cup F \cup \{u\}]$, since $S_1 = S^* \setminus S$ and $S \subseteq S^*$ by Property 2. And this private cycle is not destroyed by contracting the vertices of $F$ into $f$, since $F \subseteq F^*$ by Property 2.

We now have that $|S_H \cup S| \geq |S^*|$, because $|S_H| \geq |S^* \setminus S|$. We argue that in the process of making $S_H$ minimal to obtain $S'$, we delete at most $2k$ vertices. Indeed, every time a vertex $u$ of $S$ is removed from $S \cup S_H$ as redundant, since $u$ has at least two neighbors in $F$ by Property 4, the number of components of $G[F]$ must decrease. Similarly, every time a vertex $u \in S_H$ is removed as redundant, consider the private cycle of $u$ in $H \setminus S_H$. All of the vertices of this cycle are present in $G$ after we remove $S_H$, except $f$. Therefore, this cycle must form a path between two distinct components of $G[F]$, since $G[U]$ is acyclic and because $u$ has been considered redundant. We conclude that, since removing a vertex from $S \cup S_H$ decreases the number of components in $G[F]$, and since they are at most $2k$ such components in $G[F]$ by Property 3, we have $|S'| \geq |S^*| - 2k$. But recall that we have assumed $k \leq \frac{|S^*|}{3}$, so we obtain $|S'| \geq \frac{|S^*|}{3}$. □

Now that we have proved the existence of such a constant ratio approximation algorithm running in time exponential of a given fvs, we present our super-polynomial approximation algorithm that uses this algorithm as subroutine and which generalizes our $O(n^{2/3})$-approximation algorithm described in Section 3.3.

Recall that our approach is the following. First we construct an arbitrary minimal fvs $S$ and if $S$ is clearly large enough, return it. However, as it is no longer sufficient to consider vertices of $S$ individually or in pairs, we therefore use a divide-and-conquer approach where in each subgraph considered we use Lemma 3.21 giving us constant approximation ratio on this subgraph, enabling us to obtain the desired $r$ approximation ratio for the whole graph.

As we have mentioned in Section 1.5, instead of considering each subgraph individually, as what is done by Bonnet et al. [BLP16] for Max Min Vertex Cover, we need in our super-polynomial approximation for Max Min Feedback Vertex Set to consider pairs of subgraphs and run the sub-exponential algorithm of Lemma 3.21 on each such pairs.

**Theorem 3.22.** *There is an algorithm which, given an $n$-vertex graph $G = (V, E)$ and a value $r \leq n^{2/3}$, produces an $r$-approximation for* Max Min Feedback Vertex Set *in $G$ in time $n^{O(n/r^{3/2})}$.*

*Proof.* First, let us note that we may assume that $r$ is $\omega(1)$, because if $r$ is bounded above by a constant, then we can solve the problem exactly in the given time. To ease presentation, we will give an algorithm with approximation ratio $O(r)$. A ratio of approximation ratio exactly $r$ can be obtained by multiplying $r$ with an appropriate small constant.

Our algorithm borrows several of the basic ideas from Theorem 3.11, but requires some new ingredient, including the algorithm of Lemma 3.21. The first step is, again, to construct a minimal fvs $S$ in some arbitrary way, for example by setting $S = V$ and then removing vertices from $S$ until it becomes minimal. If $|S| \geq n/r$, then we already have an $r$-approximation, so in this case we simply return $S$. So we assume that $|S| < n/r$. From this point, this algorithm departs from the algorithm of Theorem 3.11, because it is no longer sufficient to compare the size of the output solution with a function of $n$, we need to compare it to the actual optimal value in order to obtain a ratio of $r$.

Let us now present our algorithm. Let $k = \lceil \sqrt{r} \rceil$. Partition $S$ into $k$ almost equal-sized parts $S_1, \ldots, S_k$. Our algorithm proceeds as follows: for each $i, j \in \{1, \ldots, k\}$ with $i$ and $j$ not necessarily distinct, consider the graph $G_{i,j}$ obtained by deleting all vertices of $S \setminus (S_i \cup S_j)$. Compute, using the algorithm of Lemma 3.21, a solution for $G_{i,j}$, taking into account that $S_i \cup S_j$ is an fvs of this graph, though not necessarily minimal. Then, for each of the solutions found, extend it to a solution of $G$ using Lemma 3.2. Finally, output the largest solution encountered.

The algorithm runs in the promised time: we have $|S_i \cup S_j| < \frac{2n}{rk}$, so the algorithm of Lemma 3.21 runs in time $n^{O(n/r^{3/2})}$, and the rest of the algorithm runs in polynomial time.

Let us now analyze the approximation ratio of the produced solution. Let $S^*$ be an optimal solution, and let $F = F \setminus S$ and $F^* = V \setminus S^*$ be the induced forests corresponding to $S$ and $S^*$, respectively. We would like to argue that one of the considered sub-graphs contains at least $1/r$ fraction of $S^*$, and that most of these vertices form part of a minimal fvs of that subgraph.

We will define the notion of "type" for every $u \in S^* \cap F$. For each such $u$, there must exist a private cycle in the graph $G[F^* \cup \{u\}]$, since $S^*$ is a minimal fvs. Call this cycle $c(u)$, and consider one such cycle if several exist. The cycle $c(u)$ must intersect with $S$ since $S$ is an fvs. So let $v$ be the vertex of $c(u) \cap S$ closest to $u$ on this cycle, and let $v'$ be the vertex of $c(u) \cap S$ closest to $u$ if we traverse the cycle in the opposite direction. Note

that perhaps $v = v'$. Suppose that $v \in S_i$ and $v' \in S_j$, and, without loss of generality, $i \leq j$. We then say that $u \in S^* \cap F$ has type $(i, j)$. In this way, we define a type for every $u \in S^* \cap F$. Note that, according to our definition, all internal vertices of the paths in $c(u)$ from $u$ to $v$ and from $u$ to $v'$ belong to $F^* \cap F$.

According to the definition of the previous paragraph, there are $\binom{k}{2} + k = k(k+1)/2 \leq r$ possible types of vertices in $S^* \cap F$. Therefore, there must exist a type $(i, j)$ such that at least $\frac{|S^* \cap F|}{r}$ vertices have this type. We now concentrate on the corresponding graph $G_{i,j}$, for the type $(i, j)$ that satisfies this condition. Our algorithm has constructed $G_{i,j}$ by deleting all vertices of $S \setminus (S_i \cup S_j)$. We will prove that this graph has a minimal feedback vertex set of size comparable to $\frac{|S^* \cap F|}{r}$.

For the sake of the analysis, construct a minimal feedback vertex set $S_{i,j}$ of $G_{i,j}$ as follows: start with the fvs $S_{i,j} = S^* \cap (F \cup S_i \cup S_j)$. Let $F_{i,j}$ be the corresponding induced forest $F_{i,j} = F^* \cap (F \cup S_i \cup S_j)$. The set $S_{i,j}$ is a feedback vertex set of $G_{i,j}$ as it contains all vertices of $S^*$ found in $G_{i,j}$ and $S^*$ is a feasible fvs of all of $G$. We then make $S_{i,j}$ minimal by removing vertices from it until it becomes minimal. Call the resulting set $S'_{i,j} \subseteq S_{i,j}$ and the corresponding induced forest $F'_{i,j} \supseteq F_{i,j}$.

We will prove now that the number of vertices of $S^* \cap F$ of type $(i, j)$ which have been deleted in the process of making $S_{i,j}$ minimal is upper-bounded by $|S_i \cup S_j|$. Consider such a vertex $u \in (S_{i,j} \cap F) \setminus S'_{i,j}$ of type $(i, j)$, and let $c(u)$ be the cycle that defines the type of $u$, and $v, v'$ be the vertices of $S_i \cup S_j$ which are closest to $u$ on the cycle in either direction. As we have mentioned earlier, all vertices of $c(u)$ in the paths from $u$ to $v$ and from $u$ to $v'$ belong to $F^* \cap F$, and therefore to $F_{i,j}$. If $u$ was removed as redundant, this means that $v$ and $v'$ must have been in distinct connected components at the moment $u$ was removed from the fvs $S_{i,j}$, because since $c(u)$ is a private cycle of $u$, if $u$ has been removed, it means that $v$ and $v$' are only connected by a path going through $u$ and no other vertex. However, the addition of $u$ to the induced forest creates a path from $v$ to $v'$ in the induced forest, and hence decreases the number of connected components containing vertices of $S_i \cup S_j$. The number of such connected components cannot decrease more than $|S_i \cup S_j|$ times. Thus, in the process of making $S_{i,j}$ minimal, we have removed at most $|S_i \cup S_j|$ vertices of type $(i, j)$ from $S_{i,j} \cap F$.

Using the above analysis, and the assumption that $S_{i,j}$ contains at least $\frac{|S^* \cap F|}{r}$ vertices of type $(i, j)$, we have that $\text{mmfvs}(G_{i,j}) \geq |S'_{i,j}| \geq \frac{|S^* \cap F|}{r} - |S_i \cup S_j|$. Now, we can assume that $|S^* \cap S| < \frac{|S^*|}{r}$, because otherwise $S$ is already an $r$-approximation. So we can assume that $|S^* \cap F| \geq \frac{(r-1)|S^*|}{r}$. Furthermore, we obtain $|S_i \cup S_j| \leq \frac{2|S|}{\sqrt{r}} \leq \frac{2|S^*|}{r\sqrt{r}}$, where again we assume that

$S$ is not already an $r$-approximation. Putting things together, we obtain $\text{mmfvs}(G_{i,j}) \geq \frac{(r-1)|S^*|}{r^2} - \frac{2|S^*|}{r\sqrt{r}} \geq \frac{|S^*|}{r}$, for $r$ sufficiently large. Hence, since our algorithm will return a solution that is at least as large as $\frac{\text{mmfvs}(G_{i,j})}{3}$, we obtain an $O(r)$-approximation. □

Now that we have presented our super-polynomial approximation algorithm for Max Min Feedback Vertex Set, we will in the following Section prove that it is asymptotically optimal under the randomized ETH.

## 3.6 Matching Lower Bound

In this Section, we extend Theorem 3.17 to the realm of super-polynomial time algorithms. We recall the following result of Chalermsook et al. [CLN13] we have presented in Section 1.5.

**Theorem 3.23** (Theorem 1.2 from [CLN13])**.** *For any $\epsilon > 0$ and any sufficiently large $r$, if there exists an $r$-approximation algorithm for Max Independent Set running in time $2^{(n/r)^{1-\epsilon}}$, then the randomized ETH is false.*

We remark that Theorem 3.23, which gives an almost tight running time lower bound for Max Independent Set, has already been used as a starting point to derive a similarly tight bound for the running time of any subexponential time approximation for Max Min Vertex Cover, inapproximability result due to Bonnet et al. [BLP16] that we have presented in Section 1.5. Here, we modify the proof of Theorem 3.17 to obtain a similarly tight result for Max Min Feedback Vertex Set. Nevertheless, the reduction for Max Min Feedback Vertex Set is significantly more challenging, because the ideas used in Theorem 3.17 involve an inherent quadratic blow-up in the size of the instance. As a result, in addition to executing an appropriately modified version of the reduction of Theorem 3.17, we are forced to add an extra "sparsification" step, and use a probabilistic analysis with Chernoff bounds to argue that this step does not destroy the inapproximability gap, as we have described in Section 1.5.

**Theorem 3.24.** *For any $\epsilon > 0$ and any sufficiently large $r$, if there exists an $r$-approximation algorithm for Max Min Feedback Vertex Set running in time $n^{(n/r^{3/2})^{1-\epsilon}}$, then the randomized ETH is false.*

*Proof.* We recall some details about the reduction used to prove Theorem 3.23. The reduction of Chalermsook et al. [CLN13] begins from a 3-Sat instance $\phi$ on $n$ variables, and for any $\epsilon, r$, constructs a graph $G$ with $n^{1+\epsilon}r^{1+\epsilon}$ vertices which, with high probability, satisfies the following properties: if $\phi$

is satisfiable, then $\alpha(G) \geq n^{1+\epsilon}r$; otherwise $\alpha(G) \leq n^{1+\epsilon}r^{2\epsilon}$. Hence, any approximation algorithm with ratio $r^{1-2\epsilon}$ for Max Independent Set would be able to distinguish between the two cases, and thus solve the initial 3-Sat instance. If, furthermore, this algorithm runs in time $2^{(|V(G)|/r)^{1-2\epsilon}}$, we get a sub-exponential algorithm for 3-Sat.

Suppose we are given $\epsilon, r$, and we want to prove the claimed lower bound on the running time of any algorithm that $r$-approximates Max Min Feedback Vertex Set. To ease presentation, we will assume that $r$ is the square of an integer (this can be achieved without changing the value of $r$ by more than a small constant). We will also perform a reduction from 3-Sat to show that an algorithm that achieves this ratio too rapidly would give a sub-exponential randomized algorithm for 3-Sat. We begin by executing the reduction of [CLN13], starting from a 3-Sat instance $\phi$ on $n$ variables, but adjusting their parameters $r$ and $\varepsilon$ appropriately so we obtain a graph $G$ with the following properties, with high probability:

- $|V(G)| = n^{1+\epsilon}r^{1/2+\epsilon}$
- If $\phi$ is satisfiable, then $\alpha(G) \geq n^{1+\epsilon}r^{1/2}$
- If $\phi$ is not satisfiable, then $\alpha(G) \leq n^{1+\epsilon}r^{2\epsilon}$

We now construct a graph $G'$ as follows: we keep the graph $G$; and for each pair $u, v \in V(G)$, we introduce an independent set $I_{uv}$ of size $\sqrt{r}$ connected to $u, v$. We claim that $G'$ has the following properties:

- $|V(G')| = \Theta(n^{2+2\epsilon}r^{3/2+2\epsilon})$
- If $\phi$ is satisfiable, then $\text{mmfvs}(G') = \Omega(n^{2+2\epsilon}r^{3/2})$
- If $\phi$ is not satisfiable, then $\text{mmfvs}(G') = O(n^{2+2\epsilon}r^{1/2+4\epsilon})$

Before proceeding, let us establish the properties mentioned above. The size of $|V(G')|$ is easy to bound, as for each of the $\binom{|V(G)|}{2}$ pairs of vertices of $G$ we have constructed an independent set of size $\sqrt{r}$.

For the second property, if $\phi$ is satisfiable, we construct a minimal fvs of $G'$ by starting with a minimum vertex cover of $G$ to which we add all vertices of all $I_{uv}$. We then make this fvs minimal, for example by removing vertices until it becomes minimal. We have that, for each $I_{uv}$ for which $u, v \in V \setminus C$, our set will in the end contain all of $I_{uv}$, except maybe at most one vertex. Furthermore, if one vertex of $I_{uv}$ is removed from the fvs as redundant, this decreases the number of components of the induced forest that contain

vertices of $V$, since $u, v$ are now in the same component. This cannot happen more than $|V(G)|$ times. The number of $I_{uv}$ with $u, v \in V \setminus C$ is $\binom{\alpha(G)}{2} = \Omega(n^{2+2\epsilon}r)$. So, $\text{mmfvs}(G') = \Omega(n^{2+2\epsilon}r^{3/2} - |V(G)|) = \Omega(n^{2+2\varepsilon}r^{3/2})$.

For the third property, take any minimal fvs $S$ of $G'$ and let $F$ be the corresponding forest. We have $|F \cap V| \leq 2\alpha(G)$, because $F$ is bipartite. It is sufficient to bound $|S \setminus V|$ to obtain the bound, since $|S \cap V|$ is already small enough. To do this, we note that in a set $I_{uv}$ where $u, v$ are not both in $F$, we have $I_{uv} \cap S = \emptyset$, as all vertices of $I_{uv}$ are redundant. So, the number of sets $I_{uv}$ which contain vertices of $S$ is at most $\binom{|F \cap V|}{2} = O(n^{2+2\epsilon}r^{4\epsilon})$. Each such set has size $\sqrt{r}$, giving the claimed bound.

We have now constructed an instance where the gap between the values for $\text{mmfvs}(G')$, depending on whether $\phi$ is satisfiable, is almost $r$. In fact, it is $r^{1-4\epsilon}$, but we can make it equal to $r$ by adjusting the parameters accordingly. The problem is that the order of the new graph depends quadratically on $n$. This blow-up makes it impossible to obtain a running time lower bound, as a fast approximation algorithm for Max Min Feedback Vertex Set, for example with running time $2^{n/r^2}$, would not result in a sub-exponential algorithm for 3-Sat. We therefore need to "sparsify" our instance.

We construct a graph $G''$ by taking $G'$ and deleting every vertex of $V(G') \setminus V(G)$ with probability $\frac{n-1}{n}$. That is, every vertex of the independent sets $I_{uv}$ we added survives (independently) with probability $1/n$. We now claim the following properties hold with high probability:

- $|V(G'')| = \Theta(n^{1+2\epsilon}r^{3/2+2\epsilon})$
- If $\phi$ is satisfiable, then $\text{mmfvs}(G'') = \Omega(n^{1+2\epsilon}r^{3/2})$
- If $\phi$ is not satisfiable, then $\text{mmfvs}(G'') = O(n^{1+2\epsilon}r^{1/2+4\epsilon})$

Before we proceed, let us explain why if we establish that $G''$ satisfies these properties, then we obtain the theorem. Indeed, suppose that, for some sufficiently large $r$ and $\epsilon > 0$, there exists an approximation algorithm for Max Min Feedback Vertex Set with ratio $r^{1-5\epsilon}$ running in time $N^{(N/r^{3/2})^{1-5\epsilon}}$ for graphs with $N$ vertices. The algorithm has sufficiently small ratio to distinguish between the two cases in our constructed graph $G''$, since the ratio between $\text{mmfvs}(G'')$ when $\phi$ is satisfiable or not is $\Omega(r^{1-4\epsilon})$, and since $r$ is sufficiently large. So we can use the approximation algorithm to solve 3-Sat. Furthermore, to compute the running time we see that $N/r^{3/2} = \Theta(n^{1+2\epsilon}r^{2\epsilon}) = O(n^{1+4\epsilon})$. Therefore, $(N/r^{3/2})^{1-5\epsilon} = o(n)$ and we get a sub-exponential time algorithm for 3-Sat. We conclude that for any sufficiently large $r$ and any $\epsilon > 0$, no algorithm achieves ratio $r^{1-5\epsilon}$ in time

$N^{(N/r^{3/2})^{1-5\epsilon}}$. By adjusting $r, \epsilon$ appropriately we get the statement of the theorem.

Let us therefore try to establish that the three claimed properties all hold with high probability. We will use the same Chernoff bound we have presented in Sections 1.5 and 1.6: suppose $X = \sum_{i=1}^{p} X_i$ is the sum of $p$ independent random 0/1 variables $X_i$ and that $E[X] = \sum_{i=1}^{p} E[X_i] = \mu$. Then, for all $0 \leq \delta \leq 1$ we have $Pr[|X - \mu| \geq \delta\mu] \leq 2e^{-\mu\delta^2/3}$.

The first property is easy to establish: we define a random variable $X_i$ for each vertex of each $I_{uv}$ of $G'$. This variable takes value 1 if the corresponding vertex appears in $G''$ and 0 otherwise. Let $X$ be the sum of the $X_i$ variables, which corresponds to the number of such vertices appearing in $G''$. Suppose that the number of vertices in sets $I_{uv}$ of $G'$ is $cn^{2+2\epsilon}r^{3/2+2\epsilon}$, where $c$ is a constant. Then, $E[X] = cn^{1+2\epsilon}r^{3/2+2\epsilon}$. Also, $Pr[|X - E[X]| \geq \frac{E[X]}{2}] \leq 2e^{-E[X]/12} = o(1)$. So with high probability, $|V(G'')|$ is of the promised magnitude.

For the second property, we consider a maximum minimal fvs $S$ of $G'$ of size $cn^{2+2\epsilon}r^{3/2}$. Again, we define an indicator variable for each vertex of this set in sets $I_{uv}$. The expected number of such vertices that survive in $G''$ is $cn^{1+2\epsilon}r^{3/2}$. As in the previous paragraph, with high probability the actual number will be close to this bound. We now need to argue that almost the same set is a minimal fvs of $G''$. We start in $G''$ with the surviving vertices of $S$, which is clearly an fvs of $G''$, and delete vertices until the set is minimal. We claim that the size of the set will decrease by at most $|V(G)| = n^{1+\epsilon}r^{1/2+\epsilon}$. Indeed, if $S \cap I_{uv} \neq \emptyset$, then $u, v \notin S$. The two vertices $u, v$ are (deterministically) included in $G''$ and start out in the corresponding induced forest. If a vertex of $S \cap I_{uv}$ is deleted as redundant, placing that vertex in the forest will put $u$ and $v$ in the same component, reducing the number of components of the forest with vertices from $|V(G)|$. This can happen at most $|V(G)|$ times. Since $|V(G)| < c(n^{1+2\epsilon}r^{3/2})$, for $n$ and $r$ sufficiently large and $c$ a constant, deleting these redundant vertices will not change the order of magnitude of the solution.

Finally, in order to establish the third property we need to consider every possible minimal fvs of $G''$ and show that none of them end up being too large. Consider a set $F \subseteq V(G)$ that induces a forest in $G$. Our goal is to prove that any minimal fvs $S$ of $G''$ that satisfies $V(G) \setminus S = F$ has a probability of being "too large" (that is, violating our claimed bound) much smaller than $2^{-|V(G)|}$. If we achieve this, then we can take a union bound over all sets $F$ and conclude that with high probability no minimal fvs of $G''$ is too large.

Suppose then that we have fixed an acyclic set $F \subseteq V(G)$. We have $|F| \leq 2\alpha(G) \leq 2n^{1+\epsilon}r^{2\epsilon}$. Any minimal fvs with $V(G) \setminus S = F$ can only

contain vertices from a set $I_{uv}$ if $u, v \in F$. The total number of such vertices in $G'$ is at most $O(n^{2+2\epsilon}r^{1/2+4\epsilon})$. The expected number of such vertices that survive in $G''$ is, for some constant $c$, at most $\mu = cn^{1+2\epsilon}r^{1/2+4\epsilon}$. Now, using the Chernoff bound cited above we have $Pr[|X-\mu| \geq \frac{\mu}{2}] \leq 2e^{-\mu/12}$. We claim $2e^{-\mu/12} = o(2^{-|V(G)|})$. Indeed, this follows because $|V(G)| = n^{1+\epsilon}r^{1/2+\epsilon} = o(\mu)$. As a result, the probability that a large minimal fvs exists for a fixed set $F \subseteq V(G)$ is low enough that taking the union bound over all possible sets $F$ we have that with high probability, that is with probability $o(1)$, that no minimal fvs exists with value higher than $3\mu/2$, which establishes the third property. □

With the two Theorems 3.22 and 3.24, we have completely settled the super-polynomial approximation of Max Min Feedback Vertex Set.

# Chapter 4

# Upper Dominating Set

In a graph $G(V, E)$ with $|V| = n$, a set $D \subseteq V$ is called a *dominating set* if all vertices of $V$ are dominated by $D$, that is for every $u \in V$ either $u$ belongs to $D$ or $u$ is a neighbor of some vertex in $D$. The well-known Min Dominating Set problem is studied with a minimization objective: given a graph, we are interested in finding the smallest dominating set. In this chapter, we consider *upper dominating sets*, that is dominating sets that are minimal, where a dominating set $D$ is minimal if no proper subset of it is a dominating set, that is if it does not contain any redundant vertex. We study the problem of finding an upper dominating set of maximum size. This problem is called Upper Dominating Set, and is the Max-Min version of the Min Dominating Set problem. Since it is the Max-Min version of Min Dominating Set, its private structure is mandatory, and, in contrary to Max Min Feedback Vertex Set, this structure is rather surprising. Indeed, for a dominating set to be minimal, each vertex $u$ in the solution must have a private vertex associated to it, but this vertex can either be a neighbor of $u$ *or* $u$ itself.

Studying Max-Min and Min-Max versions of some famous optimization problems is not a new idea, and it has recently attracted some interest in the literature: see the beginning of Chapter 3 for references on the most well-studied Max-Min and Min-Max problems. The Upper Dominating Set problem can be seen as a member of this framework, and studying it within this framework is one of our motivations.

This problem is also one of the six problems of the well-known *domination chain* (see [HHS98, BBCF19]) and is somewhat one which has fewer results, compared to the famous Min Dominating Set and Max Independent Set problems. Increasing our understanding of the Upper Dominating Set problem compared to these two famous problems is another motivation.

Upper Dominating Set was first considered from an algorithmic point

of view by Cheston et al. [CFHJ90], where they showed that the problem is **NP**-hard. In the more extensive paper considering this problem, Bazgan et al. [BBC+18a] studied approximability, and classical and parameterized complexity of the Upper Dominating Set problem. In the polynomial approximation paradigm, they proved that the problem does not admit an $n^{1-\varepsilon}$-approximation for any $\varepsilon > 0$, unless $\mathbf{P} = \mathbf{NP}$, making the problem as hard as Max Independent Set, whereas there exists a greedy $\ln n$-approximation algorithm for the Min version Min Dominating Set.

Considering the parameterized complexity, they proved that the problem is as hard as the Independent Set problem: Upper Dominating Set is $\mathbf{W[1]}$-hard parameterized by the standard parameter $k$. Nonetheless, in their reduction, there is an inherent quadratic blow-up in the size of the solution $k$, so they essentially proved that there is no algorithm solving Upper Dominating Set in time $O(n^{o(\sqrt{k})})$. In the next section, we improve this FPT intractability by proving that Upper Dominating Set cannot be solved in time $O(n^{o(k)})$ under the ETH, by making a linear fpt reduction from Independent Set to Upper Dominating Set and using the FPT intractability result of the former problem we have presented in Section 1.4. We mention that more recently, and independently from our work, this result was also obtained by Araujo et al. [ABCS21]. In the same section, we give an FPT inappproximability result for Upper Dominating Set: with the same reduction, we prove that, under the ETH, for any $\varepsilon > 0$ and any constant $r > 0$, there is no $r$-approximation algorithm for Upper Dominating Set running in time $O(n^{k^{1-\varepsilon}})$, using the same inapproximability result for Independent Set due to Bonnet et al. [BEKP13] we have presented in Section 1.5.

Bazgan et al. [BBC+18a] also gave FPT algorithms parameterized by the pathwidth $pw$ and the treewidth $tw$ of the graph, running in time $O^*(7^{pw})$ and $O^*(10^{tw})$, respectively. In Section 4.2, we improve the FPT algorithm parameterized by the pathwidth by giving a $O^*(6^{pw})$ algorithm. Surprisingly, this result is obtained by slightly modifying the algorithm of Bazgan et al. [BBC+18a]. Then, in Section 4.3, we give a tight lower bound: under the SETH, and for any $\varepsilon > 0$, there is no algorithm solving Upper Dominating Set in time $O^*((6-\varepsilon)^{pw})$, by making a reduction from the $q$-CSP-6 problem, as we have presented in the Introduction and similarly to what we have done for Min Mixed Dominating Set in Section 2.4.

Concerning the super-polynomial approximation of Upper Dominating Set, no result is currently known to the best of our knowledge. We thus have studied this framework for this problem, and in Section 4.4 we give an $r$-approximation algorithm for any $r \leq n$ running in time $n^{O(n/r)}$, thus matching the best trivial $n$-approximation algorithm. To obtain this al-

gorithm, we use the divide-and-conquer method, where in each subgraph we consider all independent sets, similarly to the super-polynomial approximation algorithm for Max Min Vertex Cover we have presented in Section 1.5, before we extend these independent sets to upper dominating sets of the whole graph. We also need to consider all subsets of such subgraphs in order to prove the claimed approximation ratio. Finally, we present in Section 4.5 a matching lower bound: for any $\varepsilon > 0$ and any sufficiently large $r$, there is no $r$-approximation algorithm for Upper Dominating Set running in time $n^{(n/r)^{1-\varepsilon}}$ under the randomized ETH. To derive this super-polynomial inapproximability result, and similarly to what we have done for Max Min Feedback Vertex Set in Section 3.6, we make a reduction from Max Independent Set and use the inapproximability result of Chalermsook et al. [CLN13] for this problem along with some Chernoff bounds.

## 4.1 FPT and FPT-Approximation Hardness

In this section, we present two hardness results for the Upper Dominating Set problem in the parameterized paradigm: we prove first that the considered problem is **W[1]**-hard, and more precisely that there is no FPT algorithm solving the Upper Dominating Set problem in time $O(n^{o(k)})$ under the ETH; and we prove then that for any approximation ratio $0 < r < 1$ and any $\varepsilon > 0$, there is no FPT algorithm giving an $r$-approximation for the Upper Dominating Set problem in time $O(n^{k^{1-\varepsilon}})$, again under the ETH.

In their extensive paper on the Upper Dominating Set problem, Bazgan et al. [BBC+18a] proved that the problem is as hard as the Independent Set problem: Upper Dominating Set is **W[1]**-hard parameterized by the standard parameter $k$. Nonetheless, in their reduction, from the Multicolored Clique problem to Upper Dominating Set, there is an inherent quadratic blow-up in the size of the solution $k$, so they essentially proved that there is no algorithm solving Upper Dominating Set in time $O(n^{o(\sqrt{k})})$. In this section, we present an improved hardness result for the Upper Dominating Set problem in the parameterized paradigm: we prove that the considered problem cannot be solved in time $O(n^{o(k)})$ under the ETH.

To obtain our desired negative results, we will make a reduction from the Independent Set problem to our problem. So recall that we have the following hardness result for the Independent Set problem that we have mentioned in Section 1.4 :

**Lemma 4.1** (Theorem 5.5 from [CHKX06])**.** *Under the ETH, Independent Set cannot be solved in time $O(n^{o(k)})$.*

We will obtain a similar result for the Upper Dominating Set by doing a linear fpt reduction from Independent Set. This reduction will linearly increase the size of the solutions between the two problems, so this hardness result for the latter problem will hold for the former problem.

Before we proceed further in the description of our reduction, note that we will use the variant of the Independent Set problem due to the inapproximability result of Bonnet et al. [BEKP13] we presented in Section 1.5. In this variant, the graph $G$ contains $k$ cliques which are connected to each other, and, if a solution of size $k$ exists, then this solution takes exactly one vertex per clique. Note that Lemma 4.1 holds on such particular instances, since this is a case where the problem remains hard to solve in FPT time.

Let us now present our reduction. We are given a Independent Set instance $G$ where the vertices are partitioned into $k$ distinct cliques $V_1, \ldots, V_k$. We define $A = 3$ and we set our budget to be $k' = Ak$. We construct our instance $G'$ of Upper Dominating Set as follows (see Figure 4.1):

1. For any vertex $u \in V(G)$, create an independent set $Z_u$ of size $A$.

2. For any edge $(u, v) \in E(G)$, add all edges between the vertices of $Z_u$ and the vertices of $Z_v$.

3. For any $i \in \{1, \ldots, n\}$, let $W_i$ be the *group* associated to the clique $V_i$, which contains all vertices of all independent sets $Z_u$ such that the vertex $u$ belongs to the clique $V_i$. For any $i \in \{1, \ldots, k\}$, create a vertex $z_i$ connected to all vertices of the group $W_i$.

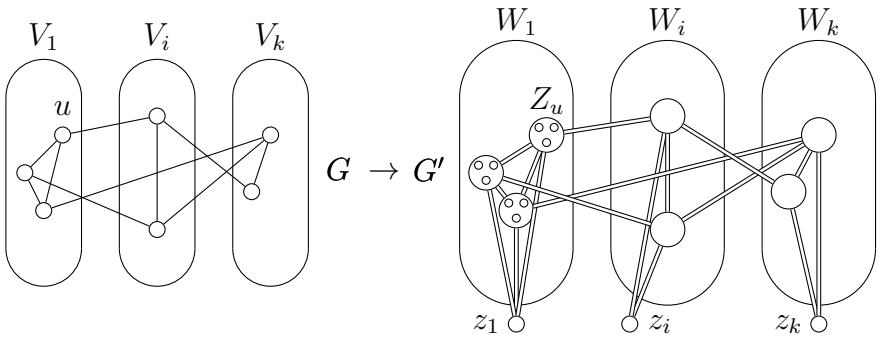


Figure 4.1: (Double edges between two sets of vertices represent all edges between the two sets.) Reduction from an instance $G$ of Independent Set to the instance $G'$ of Upper Dominating Set. Here, the reduction is for $A = 3$. Only 3 cliques $V_i$ are drawn ($V_1, V_i$ and $V_k$), and only the $A$ vertices of the independent sets $Z_u$ of the group $W_1$ are shown.

Now that we have presented our reduction, we argue that it is correct. Recall that the target size of an optimal solution in $G'$ is $k'$ as defined above. We will first prove that, given an independent set $I$ of size at least $k$ in $G$, we can construct an upper dominating set of size at least $Ak$ in $G'$ by taking the $A$ vertices of the independent set $Z_u$ for any vertex $u \in I$. Note that in our reduction, we can say that we use only one part of the mandatory private structure of the Upper Dominating Set problem. Indeed, we construct an upper dominating set which is also an independent dominating set, that is we use the fact that the private vertex of every taken vertex $u$ is itself.

**Lemma 4.2.** *If $G$ has an independent set of size at least $k$, then $G'$ has an upper dominating set of size at least $k'$.*

*Proof.* Assume $G$ admits an independent set $I$ of size at least $k$. We construct an upper dominating set $D$ of size at least $k'$ in $G'$ as follows: for any vertex $u \in V(G) \cap I$, put the $A$ vertices of the corresponding independent set $Z_u$ in $D$.

Clearly, the size of $D$ is at least $k' = Ak$ since $I$ is of size at least $k$ and every independent set $Z_u$ is of size $A$.

Consider any $i \in \{1, \ldots, k\}$, and observe that $I \cap V_i \leq 1$ since $V_i$ is a clique. But there is $k$ such cliques $V_i$ and $|I| \geq k$, so necessarily $|I| = k$ and $|I \cap V_i| = 1$ for all $i \in \{1, \ldots, k\}$.

Consider again any $i \in \{1, \ldots, k\}$ and the corresponding vertex $u$ which belongs to $I \cap V_i$. The vertices of the independent set $Z_u$ have been put in $D$. By the construction, the vertices of $Z_u$ are connected to all remaining vertices of $W_i$ and to the vertex $z_i$. So the vertices of $W_i \cup \{z_i\}$ are dominated. This is true for all $i \in \{1, \ldots, k\}$, so the graph $G'$ is dominated by $D$.

Moreover, since $I$ is an independent set, and by the construction, it follows that, for any two vertices $u, u' \in I$, there is no edge between the vertices of $Z_u$ and the vertices of $Z_{u'}$. And, since the sets $Z_u$ are independent sets, it follows that $D$ is an independent set of $G'$, which means that all vertices of $D$ are their own private vertices. □

We can now prove the other direction of the reduction. The idea of the proof is the following: if an upper dominating set in $G'$ of size at least $k'$ has not the form described in Lemma 4.2, then it cannot have size at least $k'$, enabling us to construct an independent set of size at least $k$ in $G$ from an upper dominating set which has the desired form.

**Lemma 4.3.** *If $G'$ has an upper dominating set of size at least $k'$, then $G$ has an independent set of size at least $k$.*

*Proof.* Assume $G'$ admits an upper dominating set $D$ of size at least $k'$. For any $i \in \{1, \ldots, k\}$, we give the following notations:

- If there exists at least three vertices $u_1, u_2, u_3$ in $V_i$ such that $Z_{u_j} \cap D \neq \emptyset$ for all $j \in \{1, 2, 3\}$, then we call the group $W_i$ *very bad*.
- If there exists exactly two vertices $u$ and $u'$ in $V_i$ such that $Z_u \cap D \neq \emptyset$ and $Z_{u'} \cap D \neq \emptyset$, then we call the group $W_i$ *bad*.
- Otherwise, that is if there exists at most one vertex $u \in V_i$ such that $Z_u \cap D \neq \emptyset$, then $W_i$ is called *good*.

Our proof will therefore be to consider that there is some bad and very bad groups in $G'$ and we will arrive at a contradiction on the size of $D$, which will prove that the solution $D$ as the form described in Lemma 4.2.

So suppose there exists a bad or very bad group $W_i$. Suppose first that there exists $u \in V_i$ such that $|Z_u \cap D| \geq 2$. Observe that the vertices of $Z_u$ which belong to $D$ have the same neighborhood (since they are in the same independent set $Z_u$), and observe that there exists at least one vertex $u' \in V_i \setminus \{u\}$ with $Z_{u'} \cap D \neq \emptyset$ (since $W_i$ is bad or very bad). So the vertices of $Z_u$ which belong to $D$ are dominated and share the same neighborhood, which contradicts the fact that $D$ is an upper dominating set. So, for a bad or very bad group $W_i$ and any $u \in V_i$ such that $Z_u \cap D \neq \emptyset$, we have $|Z_u \cap D| = 1$.

Consider now a bad group $W_i$ and the two vertices $u, u' \in V_i$ such that $Z_u \cap D \neq \emptyset$ and $Z_{u'} \cap D \neq \emptyset$. Let $v = Z_u \cap D$ and $v' = Z_{u'} \cap D$. The two vertices $v$ and $v'$ dominate each other since they belong to two distinct independent set $Z_u$ and $Z_{u'}$ in the group $W_i$. Observe now that $z_i$ cannot be in $D$ because otherwise it is dominated and have no private neighbor (since all vertices of $W_i$ are dominated by $v$ or $v'$). Moreover, since $W_i$ is a bad group and $|Z_u \cap D| = 1$ and $|Z_{u'} \cap D| = 1$, we have $|(W_i \cup \{z_i\}) \cap D| = 2$.

Consider now a very bad group $W_i$ and three vertices $u_1, u_2, u_3$ in $V_i$ such that $Z_{u_j} \cap D \neq \emptyset$ for all $j \in \{1, 2, 3\}$. Let $v_j = Z_{u_j} \cap D$ (since $|Z_{u_j} \cap D| = 1$) for every $j \in \{1, 2, 3\}$. Consider now any $j \in \{1, 2, 3\}$. Observe that $W_i \cup \{z_i\}$ is dominated by the two vertices $v_{j'}$ and $v_{j''}$, for $j', j'' \in \{1, 2, 3\}$ and $j', j'' \neq j$. Indeed, $(W_i \cup \{z_i\}) \setminus Z_{u_{j'}}$ is dominated by $v_{j'}$ and $Z_{u_{j'}}$ is dominated by $v_{j''}$. Since $v_j$ is dominated by both $v_{j'}$ and $v_{j''}$, it follows that $v_j$ necessarily have a private neighbor outside $W_i \cup \{z_i\}$. It is true for any vertex $u \in V_i$ such that $|Z_u \cap D| \neq \emptyset$.

Now consider such a vertex $v \in \{v_1, v_2, v_3\}$ and his private neighbor $w$, which let us say is in $W_{i'}$, for $i' \in \{1, \ldots, k\} \setminus \{i\}$, and in $Z_{u_w}$, for $u_w \in V_{i'}$. Since $w$ is a private neighbor of $v$, it follows that, for any $u' \in V_{i'} \setminus \{u_w\}$, no

vertex of $Z_{u'}$ is in $D$, because otherwise $w$ would not be a private neighbor of $v$. Moreover, the vertex $z_{i'}$ cannot be in $D$ by the same argument. So, necessarily, the group $W_{i'}$ is a good group.

Now suppose that there exists at least two vertices $w_1, w_2 \in (Z_{u_w} \setminus \{w\}) \cap D$. Observe that both $w_1$ and $w_2$ are dominated by $v$, since there is all edges between the vertices of $Z_u$ (where $v$ belongs) and the vertices of $Z_{u_w}$. But $w_1$ and $w_2$ have the same neighborhood, which contradicts the fact that $D$ is an upper dominating set. So $|Z_{u_w} \cap D| \leq 1$.

Consider any $i \in \{1, \ldots, k\}$ and observe that the vertex $z_i$ has to be dominated, and its neighborhood is the group $W_i$, so $|(W_i \cup \{z_i\}) \cap D| \geq 1$.

Now reconsider the vertex $v$. Since $W_{i'}$ is a good group, since $z_i$ does not belong to $D$, and since $|Z_{u_w} \cap D| \leq 1$, we obtain $|Z_{u_w} \cap D| = 1$.

Now consider two vertices $v$ and $v'$ belonging to $D$ and which are in the same very bad group or in two distinct very bad groups, and consider their corresponding private neighbors $w$ and $w'$. Clearly, $w$ and $w'$ do not belong to the same independent set $Z_{u_w}$, since $v$ dominates all vertices of this independent set. Moreover, since $|Z_{u_w} \cap D| = 1$ for the independent set $Z_{u_w}$ which contains the vertex $w$, the two vertices $w$ and $w'$ cannot belong to the same good group since the vertex in $Z_{u_w} \cap D$ dominates all remaining vertices of $W_{i'} \cup \{z_{i'}\}$. So, for any two such vertices $v$ and $v'$ belonging to $D$ (whether they are in the same very bad group or not), their corresponding private neighbors are in distinct good groups.

But, since in these good groups (which contain a private neighbor $w$) we have $|W_{i'} \cap D| = 1$, we have the following: for any vertex $v \in D$ which belongs to a very bad group, there exists at least one distinct good group $W_{i'}$ where a single vertex is in $D$.

Now, let $b$ be the number of bad groups and $B$ be the number of vertices in the very bad groups.

Observe now that, in a good group $W_i$ which does not contain a private neighbor $w$ of a vertex $v$ belonging to a very bad group, we have that at most $A$ vertices of $W_i \cup \{z_i\}$ are in $D$: since $W_i$ is a good group, there exists at most one vertex $u \in V_i$ such that $Z_u \cap D \neq \emptyset$, and if the $A$ vertices of $Z_u$ are in $D$, then the vertex $z_i$ cannot be in $D$ since it is dominated and all its neighbors are either in $D$ or are dominated.

So the total number of vertices in $D$ is upper-bounded by $2b + 2B + A(k - b - B) = Ak + b(2 - A) + B(2 - A)$. Indeed, we have the following: in a bad group, exactly two vertices are in $D$ and they can have their private neighbors in the group; for every vertex in $D$ in a very bad group, it has one private neighbor outside the group in a good group and a single vertex is taken in $D$ in the corresponding good group; it remains $k - b - B$ good groups in which at most $A$ vertices are in $D$, since there are $b$ bad groups,

and since the private neighbor of each vertex in a very bad group is in a distinct good group where exactly one vertex is in $D$.

But, since $A = 3 > 2$, the set $D$ has at least $k' = Ak$ vertices if and only if $b = B = 0$. So there exists no bad or very bad group in $G'$ associated to $D$.

Since there is at most $A$ vertices in $D$ for a single good group, since there is $k$ such groups, and since $|D| \geq Ak$, it follows that $|W_i \cap D| = A$ for all $i \in \{1, \ldots, k\}$.

We now construct a solution $I$ of the instance $G$ in a natural way: for any $i \in \{1, \ldots, k\}$, there exists a unique $u \in V_i$ such that $W_i \cap D = Z_u$, so take $u$ in the solution $I$.

For any $u \in I$, since $Z_u \subseteq D$ and since all vertices of $Z_u$ have the same neighborhood, it follows that the vertices of $Z_u$ are their own private vertices. So, for any two $u, u' \in I$, there is no edge between the two independent sets $Z_u$ and $Z_{u'}$. So, by the construction, the set $I$ is an independent set in $G$, of size at least $k$. □

Now that we have proved the correctness of our reduction and since the blow-up of the reduction is linear in both the size of the instance and the size of the solution, we can now present one of the main result of this section:

**Theorem 4.4.** *Under the ETH, Upper Dominating Set cannot be solved in time $O(n^{o(k)})$.*

*Proof.* Consider an instance $(G, k)$ of Independent Set. Apply our reduction to obtain an instance $(G', k')$ of Upper Dominating Set. Thanks to Lemmas 4.2 and 4.3, we know that $G$ has an independent set of size at least $k$ if and only if $G'$ has an upper dominating set of size at least $k' = Ak$.

Now suppose that there exists an algorithm that solves Upper Dominating Set in time $O(n^{o(k')})$. With this algorithm and our reduction, we can solve Independent Set in time $O(n^{o(k')})$, where $k' = Ak = O(k)$, so the total running time of this procedure is $O(n^{o(k)})$, contradicting Lemma 4.1 and the ETH. □

Thus, we cannot hope for an algorithm of complexity better than $O(n^k)$ for the Upper Dominating Set problem assuming the ETH.

We have shown that the problem Upper Dominating Set cannot be solved in time $O(n^{o(k)})$ under the ETH. Thus, a natural question is to determine if the problem admits an FPT-approximation algorithm with this complexity. We answer negatively to this question and prove the following: under the ETH, there is no $r$-approximation running in time $O(n^{k^{1-\varepsilon}})$ for Upper Dominating Set for any constant ratio $r > 0$ and any $\varepsilon > 0$.

To obtain this hardness result, we will slightly modify the reduction from the Independent Set problem to our problem. So recall that we have the following hardness result for the Independent Set problem that we have presented in Section 1.5:

**Lemma 4.5** (Corollary 2 from [BEKP13])**.** *Under the ETH, for any constant $r > 0$ and any $\varepsilon > 0$, there is no $r$-approximation algorithm for Independent Set running in time $O(n^{k^{1-\varepsilon}})$.*

Let us redefine the gap-preserving reduction from Independent Set to Upper Dominating Set. We will reuse the variant of the Independent Set problem: in this variant, the graph $G$ contains $k$ cliques which are connected to each other, and if a solution of size $k$ exists, then this solution takes exactly one vertex per clique. Note that this variant corresponds to the instance of Independent Set Bonnet et al. [BEKP13] obtained for the inapproximability of Independent Set of the previous lemma.

From now one and to obtain the FPT-approximation hardness result, we now consider our reduction above with $A$ being sufficiently large. Note that all the properties we have found before still hold since $A$ remains a constant.

Let $0 < r < 1$. To obtain the FPT-approximation hardness result for the Independent Set problem (see Lemma 4.5), Bonnet et al. [BEKP13] made a gap-amplification reduction from an instance $\phi$ of 3-Sat to an instance $(G, k)$ of Independent Set problem. Essentially, this reduction gives the following gap:

- Yes-instance: If $\phi$ is satisfiable, then $\alpha(G) = k$.
- No-instance: If $\phi$ is not satisfiable, then $\alpha(G) \leq rk$.

In this gap, recall that $\alpha(G)$ is the size of a maximum independent set in $G$, and $k$ corresponds in fact to a value which depends on the reduction, but designating it by $k$ ease our purpose.

To obtain a similar result for the Upper Dominating Set problem, and by using our reduction above, we have to prove that our reduction keep a gap of value $r$. Thus, we need to prove the following:

- Yes-instance: If $\phi$ is satisfiable, then $\alpha(G) = k$ and $\Gamma(G') = Ak$.
- No-instance: If $\phi$ is satisfiable, then $\alpha(G) \leq rk$ and $\Gamma(G') \leq rAk$.

Here, $\Gamma(G')$ is the size of a maximum upper dominating set in $G'$.

Note that we have proved the first condition in Lemma 4.2, since an independent set of size at least $k$ in $G$ necessarily has size exactly $k$ and

since all the properties we have found before in Lemmas 4.2 and 4.3 still hold because $A$ remains a constant.

Thus, we just need to prove the second condition. To prove it, we will in fact prove the contraposition. The proof of this lemma uses some arguments made in the proof of Lemma 4.3, and by choosing carefully which vertices we can put in the independent set we want to construct.

**Lemma 4.6.** *If there exists an upper dominating set in $G'$ of size $> rAk$, then there exists an independent set in $G$ of size $> rk$.*

*Proof.* Assume $G'$ admits an upper dominating set $D$ of size $> rAk$. We will use some properties we have proven in Lemma 4.3. Recall that, for any $i \in \{1, \ldots, k\}$, we make the difference whether the group $W_i$ is bad, very bad, or good. Now consider a bad group $W_i$ and recall that $|W_i \cap D| = 2$ and that there exists two vertices $u, u' \in V_i$ such that $Z_u \cap D = \{v\}$ and $Z_{u'} \cap D = \{v'\}$. For a bad group, we will distinguish between three different types of group:

- For any $j \in \{0, 1, 2\}$, we say that the group $W_i$ is *bad of type $j$* if there exists exactly $j$ vertices among $v$ and $v'$ which have their private neighbor outside of $W_i \cup \{z_i\}$.

Now, given our upper dominating set $D$, we construct an independent set $I$ of $G$ as follows:

- For a bad group $W_i$ of type 0, recall that there exists exactly two vertices $u, u' \in V_i$ such that $|Z_u \cap D| = 1$ and $|Z_{u'} \cap D| = 1$. Put in the solution either $u$ or $u'$.

- For a bad group $W_i$ of type 1, and without loss of generality, let $u \in V_i$ be such that $v = Z_u \cap D$ is the unique vertex of $W_i$ which have its private neighbor outside of $W_i \cup \{z_i\}$. Let $w$ be the private neighbor of $v$, and let $u_w$ be the vertex of $V(G)$ for which $w \in Z_{u_w}$. Put the vertex $u_w$ in the solution.

- For a bad group $W_i$ of type 2, let $u, u' \in V_i$ be the two vertices such that $Z_u \cap D \neq \emptyset$ and $Z_{u'} \cap D \neq \emptyset$. Let $v = Z_u \cap D$ and $v' = Z_{u'} \cap D$, let $w$ and $w'$ be the private neighbors of $v$ and $v'$, respectively, and let $u_w$ and $u_{w'}$ be the vertices of $V(G)$ for which $w \in Z_{u_w}$ and $w' \in Z_{u_{w'}}$, respectively. Put the vertices $u_w$ and $u_{w'}$ in the solution.

- For a very bad group $W_i$, let $v$ be any vertex in $W_i \cap D$. Let $w$ be the private neighbor of $v$ and let $u$ be the vertex of $V(G)$ for which $w \in Z_u$. Put $u$ in the solution. Do this for all the vertices $v \in W_i \cap D$.

- For a good group $W_i$, let $u \in V_i$ such that $Z_u \cap D \neq \emptyset$. If $|Z_u \cap D| \geq 2$, then put $u$ in the solution.

We will prove first that the solution $I$ we have constructed is an independent set of $V(G)$.

Consider a bad group $W_i$ of type 0, and let $v = Z_u \cap D$ and $v' = Z_{u'} \cap D$. Since $W_i$ is of type 0, it means that the private neighbor of $v$ is in $Z_{u'}$ and the private neighbor of $v'$ is in $Z_u$, since $(W_i \cup \{z_i\}) \setminus (Z_u \cup Z_{u'})$ are dominated by both $v$ and $v'$. It means that the vertices of $Z_u$ are only dominated by $v'$ and the vertices of $Z_{u'}$ are only dominated by $v$. So, since we put either $u$ or $u'$ in the solution, the vertex selected in $I$ has no neighbor in $I$.

Consider now a bad group $W_i$ of type 1. Since $w$ is the private neighbor of $v$, it means that the vertices of $Z_{u_w}$ are only dominated by $v$. So the selected vertex $u_w$ has no neighbor in $I$.

Consider now a bad group $W_i$ of type 2. By a similar argument as for a bad group of type 1, the vertices of $Z_{u_w}$ are only dominated by $v$, and the vertices of $Z_{u_{w'}}$ are only dominated by $v'$. So, the two selected vertices $u_w$ and $u_{w'}$ have no neighbor in $I$.

Consider now a very bad group $W_i$. We have proven in Lemma 4.3 that, for any vertex $v \in W_i \cap D$, it has a private neighbor outside $W_i \cup \{z_i\}$ in an independent set $Z_u$. So the vertices of $Z_u$ are only dominated by the vertex $v$. So the selected vertex $u$ has no neighbor in $I$. This is true for all vertices $v \in W_i \cap D$.

Consider now a good group $W_i$. Since $W_i$ is a good group, there exists at most one $u \in V_i$ such that $Z_u \cap D \neq \emptyset$. If $|Z_u \cap D| \geq 2$, then the vertices of $Z_u$ in $D$ are their own private vertices. So the selected vertex $u$ has no neighbor in $I$.

So the solution $I$ we have constructed is an independent set in $G$.

We will now show that $|I| > rk$.

First, consider a vertex $u \in V_i$ such that $W_i$ is bad of type 1 or 2, $Z_u \cap D = \{v\}$, and the private neighbor of $v$ is outside $W_i \cup \{z_i\}$. By the same arguments as for the very bad groups in Lemma 4.3, we have that the private neighbor $w$ of $v$ is in a good group $W_{i'}$ and in a set $Z_{u'}$ such that $|(W_{i'} \cup \{z_{i'}\}) \cap D| = 1$ and this unique vertex must be in $Z_{u'}$.

Now, we give the following notations: let $b_j$ be the number of bad groups of type $j$, for $j \in \{0, 1, 2\}$; let $B$ be the number of vertices in all the very bad groups; let $B'$ be the number of very bad groups; let $F$ be the number of good groups which have at least two vertices in $D$; and let $f$ be the number of good groups which have at most one vertex in $D$ and which do not contain private neighbors of vertices in the bad and very bad groups.

We have $|I| \geq (b_0 + b_1 + 2b_2) + B + F$. This is easy to see from how we constructed $I$.

Recall that there exists exactly $k$ groups $W_i$. We have the following: $F \geq k - ((b_0 + 2b_1 + 3b_2) + (B + B') + f)$. Indeed, from $k$ groups $W_i$, we subtract the following: the $b_0$ bad groups of type 0; the $b_1$ bad groups of type 1 and the corresponding good groups (there is one such good group for each bad group of type 1); the $b_2$ bad groups of type 2 and the corresponding good groups (there is two such good groups for each bad group of type 2); the $B$ good groups which contain the private neighbors of the vertices of the very bad groups; the $B'$ very bad groups; and the $f$ remaining good groups.

From these two inequalities, we obtain the following: $|I| \geq k - ((b_1 + b_2) + B' + f)$.

Now, we will upper-bound the size of $D$ to upper-bound $B'$. Recall that for a good group counted in $F$, there are at most $A$ vertices which can be in $D$. We have: $|D| \leq (2b_0 + 3b_1 + 4b_2) + 2B + f + AF$. To see this, make the following observations: two vertices are taken for each bad group of type 0; two vertices and a single vertex from a good group are taken for each bad group of type 1; two vertices and a single vertex from two good groups are taken for each bag group of type 2; $B$ vertices and a single vertex from $B$ good groups are taken for the very bad groups; at most $A$ vertices are taken for the good groups with $|W_i \cap D| \geq 2$ ; and at most one vertex is taken for each remaining good group.

But $D$ is of size $> rAk$, so we obtain:

$$Ak + b_0(2 - A) + b_1(3 - 2A) + b_2(4 - 3A) + B(2 - A) - AB' + f(1 - A) > rAk$$

Which is equivalent to:

$$B' < b_0(2/A - 1) + b_1(3/A - 2) + b_2(4/A - 3) + B(2/A - 1) + f(1/A - 1) - (r - 1)k$$

For $A$ sufficiently large, we obtain: $B' < -b_1 - b_2 - f - (r - 1)k$

With this inequality and the one on the size of $I$, we obtain:

$$|I| > k - ((b_1 + b_2) + f) + (b_1 + b_2 + f + (r - 1)k) = k + (r - 1)k = rk$$

So we have constructed an independent set $I$ of $G$ of size $> rk$. □

Now that we have proved the correctness of our gap-preserving reduction, we can present the main result of this section:

**Theorem 4.7.** *Under the ETH, for any constant $r > 0$ and any $\varepsilon > 0$, there is no $r$-approximation algorithm for* Upper Dominating Set *running in time $O(n^{k^{1-\varepsilon}})$.*

*Proof.* Fix $0 < r < 1$ and $\varepsilon > 0$. Consider an instance $\phi$ of 3-Sat. Apply the reduction of Bonnet et al. [BEKP13] to obtain an instance $(G, k)$ of Independent Set, and then apply our reduction to obtain an instance $(G', k')$ of Upper Dominating Set. Thanks to Lemmas 4.2 and 4.6, and to the gap-amplification reduction of Bonnet et al. [BEKP13] (see Lemma 4.5), we know the following:

- Yes-instance: If $\phi$ is satisfiable, then $\alpha(G) = k$ and thus $\Gamma(G') = Ak$.
- No-instance: If $\phi$ is not satisfiable, then $\alpha(G) \leq rk$ and thus $\Gamma(G') \leq rAk$.

Now suppose there exists an algorithm that outputs an $r$-approximation for Upper Dominating Set in time $O(n^{k^{1-\varepsilon}})$. With this algorithm and our reduction, we can obtain an $r$-approximation for Independent Set, and thus determine if $\phi$ is satisfiable or not in time $O(n^{k^{1-\varepsilon}})$. But, by Lemma 4.5, this would contradict ETH. □

By this theorem, we cannot hope under the ETH for an FPT-approximation algorithm with constant ratio running in time $O(n^{k^{1-\varepsilon}})$. In the next Section, we present an improved FPT algorithm parameterized by the pathwidth for Upper Dominating Set.

## 4.2 Improved Pathwidth Algorithm

In this Section, we prove that, given a graph $G = (V, E)$ and a path decomposition $(T, \{X_t\}_{t \in V(T)})$ of width $pw$, there exists a dynamic programming algorithm that solves Upper Dominating Set problem in time $O(6^{pw} \cdot pw)$.

We now suppose that we are given a path decomposition $(T, \{X_t\}_{t \in V(T)})$ of the given graph $G = (V, E)$. Recall, as we have presented in Section 1.4, that in such a path decomposition, we have three types of bag: the *Initialization* bags, the *Forget* bags, and the *Introduce* bags. We can assume that we are given a nice path decomposition, where a vertex is introduced only once, and forgotten only once also. So we only need to describe the Initialization, Forget and Introduce nodes.

Our algorithm essentially works as the $O^*(7^{pw})$ algorithm of Bazgan et al. [BBC+18a], but we are more careful in the Introduce bags, which enables us to get the desired complexity.

We will now present how our dynamic programming works for each type of bag. To do so, we first distinguish between six different colors for each vertex. We define a *coloring* of a bag $X_t$ to be a mapping $f : X_t \rightarrow$

$\{I, F, F^*, O^*, O, P\}$ assigning six different colors to the vertices of the bag $X_t$. These six colors are defined below, and directly come from the private structure of Upper Dominating Set, i.e. the fact the every vertex $u$ in the solution either has a private neighbor, or is its own private vertex.

- $I$: the set of vertices which are in the dominating set and which forms an independent set, i.e. the vertices of the solution which are their own private vertices.
- $F$: the set of vertices which are in the dominating set and which are already matched to a private neighbor.
- $F^*$: the set of vertices which are in the dominating set and which have no private neighbor yet.
- $O^*$: the set of vertices which are not in the dominating set and which are not dominated yet.
- $O$: the set of vertices which are not in the dominating set, which are dominated, but which are not private neighbors of vertices of the solution.
- $P$: the set of vertices which are not in the dominating set, which are dominated, and which are private neighbors of some vertex of the solution.

Note that, since $f^{-1}(I) \cup f^{-1}(F) \cup f^{-1}(F^*) \cup f^{-1}(O^*) \cup f^{-1}(O) \cup f^{-1}(P)$ is a partition of $X_t$, there are $6^{|X_t|}$ colorings of $X_t$. These colorings form the space of states of the node $X_t$, and we will use this fact to improve the algorithm of Bazgan et al. [BBC+18a] from $O^*(7^{pw})$ to $O(6^{pw} \cdot pw)$.

For a bag $X_t$, we denote by $c[t, f]$ the maximum size of an upper dominating set $D \subseteq V_t$ (where $V_t$ denotes the set of vertices belonging to any bag $X_t$ of the subtree $T_t$ rooted at the node $t$), such that:

- $f^{-1}(I) \cup f^{-1}(F) \cup f^{-1}(F^*) = D \cap X_t$;
- $f^{-1}(O) \cup f^{-1}(P)$ is dominated by $D$.

We call such a set $D$ a *maximum compatible* set for $t$ and $f$. If no maximum compatible set for $t$ and $f$ exists, then we put $c[t, f] = -\infty$.

Let us now define some useful notations. For a subset $X \subseteq V$, consider a coloring $f : X \to \{I, F, F^*, O^*, O, P\}$. For a vertex $v \in V$, and a color $\alpha \in \{I, F, F^*, O^*, O, P\}$, we define a new coloring $f_{v \to \alpha} : X \cup \{v\} \to \{I, F, F^*, O^*, O, P\}$ as follows:

$$f_{v\to\alpha}(x) = \begin{cases} f(x) & \text{if } x \neq v \\ \alpha & \text{if } x = v \end{cases}$$

We now proceed to present the recursive formulas for the values of $c$.

**Initialization node**. For a node $X_t = \{v\}$ which initializes the table, we make the following observations: the vertex $v$ cannot be in $F$ since it cannot have a private neighbor; it cannot be neither in $O$ nor in $P$ since it cannot be dominated; and for the three other cases (for $I$, $F^*$ and $O^*$), we just have to give the size of the corresponding solution. We obtain:

$$c[t,f] = \begin{cases} 1 & \text{if } v \in I \cup F^* \\ 0 & \text{if } v \in O^* \\ -\infty & \text{otherwise} \end{cases}$$

**Forget node**. Let $t$ be a forget node with a unique child $t'$ such that $X_t = X_{t'} \setminus \{v\}$ for some $v \in X_{t'}$. We make the following observations: the vertex $v$ cannot be forgotten if it belongs to $F^*$ since it contradicts the fact that the solution is minimal; $v$ cannot be forgotten if it belongs to $O^*$ since in this case it remains not dominated; the four other cases are valid and we just need to take the maximum value between these four cases. We obtain:

$$c[t,f] = \max\{c[t', f_{v\to I}], c[t', f_{v\to F}], c[t', f_{v\to O}], c[t', f_{v\to P}]\}$$

Note that for these Initialization and Forget nodes, since in the worst case we go through all possible colorings of the bag $X_t$, the running-time for these two types of bags is $O(6^{pw})$.

**Introduce node**. Let $t$ be an introduce node with a unique child $t'$ such that $X_t = X_{t'} \cup \{v\}$ for some $v \notin X_{t'}$. Here, instead of going through all possible colorings of the bag $t$ and considering every subset of the neighborhood of $v$ to put in $O^*$, as Bazgan et al. did [BBC+18a], we go through all possible colorings of the bag $t'$ and update the value of $c[t,f]$ depending on the corresponding coloring and any color affected to $v$. This enables us to lower the complexity to $O(6^{pw} \cdot pw)$ since we don't need anymore to go through every subset of the neighborhood of $v$. First, we affect the value $-\infty$ to $c[t,f]$ for every coloring $f$ of the bag $t$. Then, for every coloring $f : X_{t'} \to \{I, F, F^*, O^*, O, P\}$ and any color $\alpha : \{v\} \to \{I, F, F^*, O^*, O, P\}$, we will define a new coloring $f_{new}$ which corresponds to the coloring $f$ of the vertices of $X_{t'}$ plus the coloring $\alpha$ of $v$, and an associated value $k_{new}$ which will be the size of the corresponding upper dominating set. To get the final value $c[t, f_{new}]$ for the bag $t$, we just update it by $k_{new}$ if $c[t, f_{new}] < k_{new}$ for

the new coloring $f_{new}$, so at the end each entry of the table of $t$ will have the maximum size for the corresponding coloring $f_{new}$. Now, for each coloring $f$ of the bag $t'$ and each color $\alpha \in \{I, F, F^*, O^*, O, P\}$ of the vertex $v$, we have the following cases:

- If $\alpha = I$ and $N(v) \cap (f^{-1}(I) \cap f^{-1}(F) \cap f^{-1}(F^*) \cap f^{-1}(P)) = \emptyset$, then:

$$f_{new}(u) = \begin{cases} O & \text{if } f(u) = O^* \text{ and } (u,v) \in E \\ f(u) & \text{otherwise} \end{cases}$$

For $v$ to be in $I$, we need that all its neighbors are either in $O$ or in $O^*$. Then, if this condition is satisfied, we can give the color $O$ to all neighbors of $v$ which are not dominated in the bag $t'$ since they become dominated in the new bag.

- If $\alpha = F$ and $N(v) \cap (f^{-1}(I) \cup f^{-1}(P)) = \emptyset$ and $\exists w \in N(v) \cap f^{-1}(O^*)$, then:

$$f_{new}(u) = \begin{cases} P & \text{if } u = w \\ O & \text{if } f(u) = O^* \text{ and } u \neq w \text{ and } (u,v) \in E \\ f(u) & \text{otherwise} \end{cases}$$

For $v$ to be in $F$, note that its neighbors cannot be in $I$ or in $P$, because otherwise its neighbors in $I$ are dominated and its neighbors in $P$ cannot be private neighbors of other vertices of $F$ anymore. Note also that at least one neighbor of $v$ has to be in $O^*$ in the bag $t'$ in order to become the private neighbor of $v$. Moreover, $v$ belongs to $F$ if at least one of its neighbors belongs to $P$ in the new coloring, so we can take any neighbor of $v$ which is not dominated in $t'$ and put it in $P$ in the new coloring, since it is enough to have just one neighbor of $v$ being its private neighbor. For the other neighbors of $v$ which are not dominated in $t'$, we can give them the color $O$ since they become dominated. Finally, all other vertices keep the same color from the coloring $f$.

- If $\alpha = F^*$ and $N(v) \cap (f^{-1}(I) \cup f^{-1}(P)) = \emptyset$, then:

$$f_{new}(u) = \begin{cases} O & \text{if } f(u) = O^* \text{ and } (u,v) \in E \\ f(u) & \text{otherwise} \end{cases}$$

For $v$ to be in $F^*$, we need that its neighbors are neither in $I$ nor in $P$, because otherwise its neighbors in $I$ are dominated and its neighbors in

$P$ cannot be private neighbors of vertices of $F$ anymore. If this condition is satisfied, we can give the color $O$ to all neighbors of $v$ which are not dominated in the bag $t'$ since they become dominated in the new bag.

- If $\alpha = O^*$ and $N(v) \cap (f^{-1}(I) \cup f^{-1}(F) \cup f^{-1}(F^*)) = \emptyset$, then: $f_{new}(u) = f(u)\ \forall u \in X_{t'}$

For $v$ to be in $O^*$, we just need to check that all its neighbors are not in the solution, that is they do not have the colors $I$, $F$ or $F^*$. If this condition is satisfied, $v$ can be added as a non dominated vertex and the coloring $f_{new}$ is the coloring $f$.

- If $\alpha = O$ and $N(v) \cap (f^{-1}(I) \cup f^{-1}(F) \cup f^{-1}(F^*)) \neq \emptyset$, then: $f_{new}(u) = f(u)\ \forall u \in X_{t'}$

For $v$ to be in $O$, we just need that at least one of its neighbors is in the solution, that is one of its neighbor is in $I$, in $F$ or in $F^*$. If this condition is satisfied, $v$ can get color $O$ and the coloring $f_{new}$ is the coloring $f$.

- If $\alpha = P$ and $N(v) \cap (f^{-1}(I) \cup f^{-1}(F)) = \emptyset$ and $\exists! w \in N(v) \cap f^{-1}(F^*)$, then:

$$f_{new}(u) = \begin{cases} F & \text{if } u = w \\ f(u) & \text{otherwise} \end{cases}$$

For $v$ to have color $P$, we need firstly that its neighbors are neither in $I$ nor in $F$, because otherwise $v$ cannot be a private neighbor of some vertex of the solution, and we also need that exactly one neighbor of $v$ is in $F^*$ in the bag $t'$, so that these two vertices become matched. If these conditions are satisfied, we give color $F$ to the only neighbor of $v$ in $F^*$ in the bag $t'$ and all other vertices keep the same color from the coloring $f$.

Note that, since we go through all possible colorings $f$ in the bag $t'$ and through all colors for the vertex $v$, the running time of any Introduce node is $O(6^{pw} \cdot pw)$.

Given this dynamic programming algorithm, we have the following result:

**Theorem 4.8.** *The* Upper Dominating Set *problem can be solved in time $O(6^{pw} \cdot pw)$, where $pw$ is the pathwidth of the input graph.*

Now that we have improved the best FPT algorithm parameterized by the pathwidth from $O^*(7^{pw})$ to $O^*(6^{pw})$, we will show in the following section that our algorithm is optimal under SETH.

## 4.3 Tight Intractability Result

We prove in this section that, under the SETH, for all $\varepsilon > 0$, there is no algorithm for Upper Dominating Set with complexity $O^*((6-\varepsilon)^{pw})$. To get this result, we will do a reduction from the $q$-CSP-6 problem (see [Lam20]) to the Upper Dominating Set problem, similarly to our reduction for Min Mixed Dominating Set in Section 2.4. Recall that, in the $q$-CSP-6 problem, we are given a CSP instance with $n$ variables and $m$ constraints. The variables take values over a set of size 6. Without loss of generality, let $\{0,1,2,3,4,5\}$ be this set. Each constraint involves at most $q$ variables, and is given as a list of acceptable assignments for these variables. Without loss of generality, we force the following condition: each constraint involves exactly $q$ variables, because if it has fewer, we can add to it new variables and augment the list of satisfying assignments so that the value of the new variables is irrelevant.

Recall also the following result of Lampis [Lam20] to be a natural consequence of the SETH, specified with the value $B = 6$:

**Lemma 4.9** (Lemma 2 from [Lam20])**.** *If the SETH is true, then, for all $\varepsilon > 0$, there exists a $q$ such that $n$-variables $q$-CSP-6 cannot be solved in time $O^*((6-\varepsilon)^n)$.*

We will produce a polynomial-time reduction, similarly to what we have done for Min Mixed Dominating Set in Section 2.4, from an instance of $q$-CSP-6 with $n$ variables to an equivalent instance of Upper Dominating Set whose pathwidth is bounded by $n + O(1)$. Thus, any algorithm for the latter problem running faster than $O^*((6-\varepsilon)^{pw})$ would give a $O^*((6-\varepsilon)^n)$ algorithm for the former problem, contradicting the SETH.

The constructed graph consists of a main part of $n$ paths of length $4m$, each divided into $m$ sections. The idea is that an optimal solution will verify, for each path, a specific pattern in the whole graph. For four consecutive vertices, there are six ways for taking exactly two vertices among the four and dominating the two others. These six ways for each path will represent all possible assignments for all variables (see Figure 4.2). Then, we will add some *verification* gadgets for each constraint and attach it to the corresponding section, in order to check that the selected assignment satisfies the constraint or not.

As for the Min Mixed Dominating Set problem, a first difficulty is to prove that an optimal solution of the Upper Dominating Set instance has the desired form, and more precisely that the pattern selected for a variable is kept constant throughout the graph. To answer this difficulty, we make a polynomial number of copies of this construction and we connect them

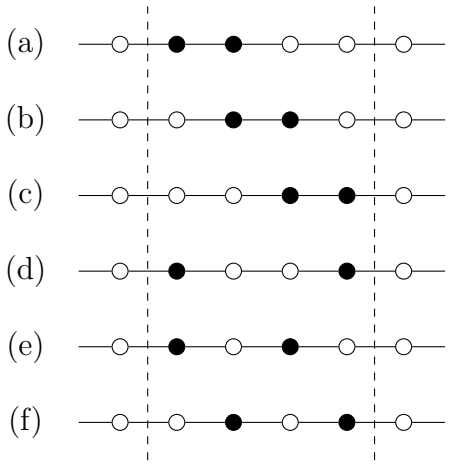


Figure 4.2: Main part of the construction with the six possible configurations. Filled vertices are in $D$.

together, enabling us to have a sufficiently large copy where the patterns are kept constant in this copy.

Moreover, we need to be careful in our verification gadget in order to have the following conditions: the vertices of the paths taken in the solution must not have any private neighbor in the corresponding verification gadget, because otherwise it would be impossible to keep the patterns constant in a sufficiently large copy of the graph; and the vertices of the paths not taken in the solution must not be dominated by the corresponding verification gadget, because otherwise there can be some vertices of the paths taken in the solution that have no private neighbor. Thus, we extensively use in our proofs that a vertex in an upper dominating set either is its own private vertex, or has at least one private neighbor.

We now present our reduction. We are given a $q$-CSP-6 instance $\varphi$ with $n$ variables $x_1, \ldots, x_n$ taking values over the set $\{0, 1, 2, 3, 4, 5\}$, and $m$ constraints $c_0, \ldots, c_{m-1}$, each containing exactly $q$ variables and $C_j$ possible assignments over these $q$ variables, for each $j \in \{0, \ldots, m-1\}$. We define the following numbers: $A = 4q + 2$ and $F = (2n + 1)(4n + 1)$. We set our budget to be $k = Fm(2n + A) + 2n$.

We construct our instance of Upper Dominating Set as follows:

1. For $i \in \{1, \ldots, n\}$, we construct a path $P_i$ of $4Fm + 6$ vertices: the vertices are labeled $u_{i,j}$ for $j \in \{-3, \ldots, 4Fm+2\}$; and for each $i, j$ the vertex $u_{i,j}$ is connected to $u_{i,j+1}$. We call these paths the *main* part of our graph.

2. For each section $j \in \{0, \ldots, Fm-1\}$, let $j' = j \mod m$. We construct

a verification gadget $H_j$ as follows (see Figure 4.3):

(a) A clique $K_j$ of size $AC_{j'}$ such that the $AC_{j'}$ vertices are partitioned into $C_{j'}$ cliques $K_j^1, \ldots, K_j^{C_{j'}}$, each corresponding to a satisfying assignment $\sigma_l$ in the list of $c_{j'}$, for $l \in \{1, \ldots, C_{j'}\}$, and each containing exactly $A$ vertices.

(b) A clique $L_j$ of size $AC_{j'}$ such that the $AC_{j'}$ vertices are partitioned into $C_{j'}$ cliques $L_j^1, \ldots, L_j^{C_{j'}}$, each containing exactly $A$ vertices.

(c) For each $i \in \{1, \ldots, n\}$ such that $x_i$ is involved in $c_{j'}$, and for each satisfying assignment $\sigma_l$ in the list of $c_{j'}$: if $\sigma_l$ sets $x_i$ value 0, connect the two vertices $u_{i,4j+2}$ and $u_{i,4j+3}$ to the $A$ vertices of the clique $K_j^l$; if $\sigma_l$ sets $x_i$ value 1, connect the two vertices $u_{i,4j+3}$ and $u_{i,4j}$ to the $A$ vertices of the clique $K_j^l$; if $\sigma_l$ sets $x_i$ value 2, connect the two vertices $u_{i,4j}$ and $u_{i,4j+1}$ to the $A$ vertices of the clique $K_j^l$; if $\sigma_l$ sets $x_i$ value 3, connect the two vertices $u_{i,4j+1}$ and $u_{i,4j+2}$ to the $A$ vertices of the clique $K_j^l$; if $\sigma_l$ sets $x_i$ value 4, connect the two vertices $u_{i,4j+1}$ and $u_{i,4j+3}$ to the $A$ vertices of the clique $K_j^l$; if $\sigma_l$ sets $x_i$ value 5, connect the two vertices $u_{i,4j}$ and $u_{i,4j+2}$ to the $A$ vertices of the clique $K_j^l$.

(d) For each satisfying assignment $\sigma_l$ in the list of $c_{j'}$, do the following: add a perfect matching between the vertices of $K_j^l$ and the vertices of $L_j^l$; for any $l' \in \{1, \ldots, C_{j'}\}$ with $l' \neq l$, add all the edges between the vertices of $K_j^l$ and the vertices of $L_j^{l'}$.

(e) Add a vertex $w$ connected to all the vertices of the clique $L_j$.

Now that we have presented our reduction, we argue that it is correct and that the obtained graph $G$ has the desired pathwidth. Recall that the target size of an optimal solution in $G$ is $k$ as defined above.

**Lemma 4.10.** *If $\varphi$ is satisfiable, then there exists an upper dominating set in $G$ of size at least $k$.*

*Proof.* Assume $\varphi$ admits some satisfying assignment $\rho : \{x_1, \ldots, x_n\} \to \{0, 1, 2, 3, 4, 5\}$. We construct a solution $D$ of the instance $G$ of Upper Dominating Set as follows:

1. For each $i \in \{1, \ldots, n\}$, let $\alpha$ and $\beta$ be the following numbers: if $\rho(x_i) = 0$, let $\alpha = 2$ and $\beta = 3$; if $\rho(x_i) = 1$, let $\alpha = 3$ and $\beta = 0$; if $\rho(x_i) = 2$, let $\alpha = 0$ and $\beta = 1$; if $\rho(x_i) = 3$, let $\alpha = 1$ and $\beta = 2$; if

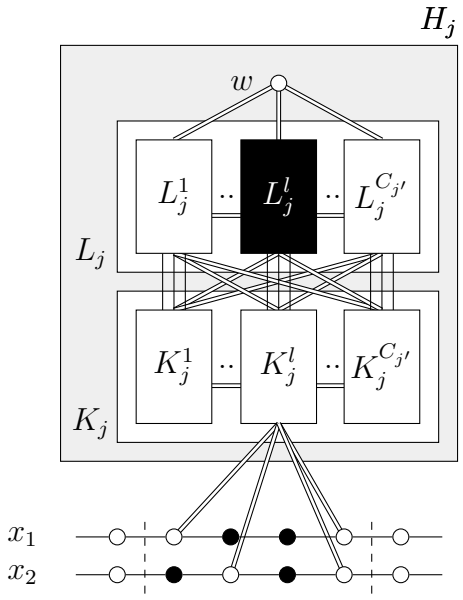


Figure 4.3: (Double edges between two sets of vertices represent all edges between the two sets, triple edges between two sets of vertices represent perfect matching between the two sets.) Checker gadget $H_j$ connected to the main part. Here we have considered an instance where the clause $c_{j'}$ has only two variables, $x_1$ and $x_2$. Moreover, only the clique $K_j^l$ is shown connected to the main part. The possible assignment $\sigma_l$ of $c_{j'}$ is $(x_1 = 1, x_2 = 3)$. We have supposed that this assignment is satisfiable, and we have marked the corresponding upper dominating set: filled vertices and filled rectangles are in $D$.

$\rho(x_i) = 4$, let $\alpha = 1$ and $\beta = 3$; if $\rho(x_i) = 5$, let $\alpha = 0$ and $\beta = 2$. Let $U = \bigcup_{j=0}^{Fm-1}\{u_{i,4j+\alpha}, u_{i,4j+\beta}\}$. We add to the solution all vertices of $(V(P_i) \setminus \{u_{i,-3}, u_{i,-2}, u_{i,-1}, u_{i,4Fm}, u_{i,4Fm+1}, u_{i,4Fm+2}\}) \setminus U$.

2. For each $j \in \{0, \ldots, Fm-1\}$, let $j' = j \mod m$. Consider the unique possible assignment $\sigma_{l^*}$ in the list of $c_{j'}$ satisfied by $\rho$ (such a unique possible assignment must exist since $\rho$ satisfies $\varphi$), and take the $A$ vertices of the clique $L_j^{l^*}$.

3. For each $i \in \{1, \ldots, n\}$, do the following: if $\rho(x_i) = 0$, then add $u_{i,-3}$, $u_{i,4Fm}$ and $u_{i,4Fm+1}$ to $D$; if $\rho(x_i) = 1$, then add $u_{i,-2}$ and $u_{i,4Fm+1}$ to $D$; if $\rho(x_i) = 2$, then add $u_{i,-2}$, $u_{i,-1}$ and $u_{i,4Fm+2}$ to $D$; if $\rho(x_i) = 3$, then add $u_{i,-3}$ and $u_{i,4Fm+2}$ to $D$; if $\rho(x_i) = 4$, then add $u_{i,-3}$ and $u_{i,4Fm+1}$ to $D$; if $\rho(x_i) = 5$, then add $u_{i,-2}$ and $u_{i,4Fm+2}$ to $D$.

Let us now argue why this solution has size at least $k$. In the first step, we have selected $2Fmn$ vertices. To see this, let $Q_{i,j}$ be the subpath of $P_i$ corresponding to the section $j$ ($j \in \{0, \ldots, Fm-1\}$), i.e. $Q_{i,j} = \{u_{i,4j}, u_{i,4j+1}, u_{i,4j+2}, u_{i,4j+3}\}$. Observe that we have put exactly two vertices of $Q_{i,j}$ in $U$, which leaves two vertices in the solution, for all $i$ and all $j$. Consider now any $j \in \{0, \ldots, Fm-1\}$ and the corresponding verification gadget $H_j$. In this gadget, we have selected all the vertices of the clique $L_j^{l^*}$, corresponding to the satisfied assignment $\sigma_{l^*}$. So we have selected $AFm$ vertices for all the verification gadgets. Finally, at least $2n$ vertices have been added to the solution at Step 3. So the total size is at least $2Fmn + AFm + 2n = k$.

Let us now argue why the solution is a valid upper dominating set.

Consider any $j \in \{0, \ldots, Fm-1\}$ and let $j' = j \mod m$. We have selected the $A$ vertices of the clique $L_j^{l^*}$ corresponding to the unique possible assignment $\sigma_{l^*}$ in the list of $c_{j'}$ satisfied by $\rho$ (such a unique possible assignment must exist since $\rho$ satisfies $\varphi$). Since $L_j$ is a clique, since the vertices of $L_j^{l^*}$ are connected to all vertices of $K_j^{l'}$, for any $l' \in \{1, \ldots, C_{j'}\}$ with $l' \neq l^*$, since there is a perfect matching between the vertices of $L_j^{l^*}$ and the vertices of $K_j^{l^*}$, and since the vertex $w$ is connected to all vertices of $L_j$, we have that all the vertices of $H_j$ are dominated by $D$.

Now, observe that, since $\sigma_{l^*}$ is satisfied by $\rho$, it means that the values given by $\rho$ to the variables appearing in the constraint $c_{j'}$ satisfy $\sigma_{l^*}$, so by the construction it follows that the neighbors of the vertices of $K_j^{l^*}$ in the paths all belong to $U$. Indeed, consider any variable $x_i$ appearing in $c_{j'}$: if $\sigma_{l^*}$ sets value 0 to $x_i$, then $\rho(x_i) = 0$, and then, for $\alpha = 2$ and $\beta = 3$, we have that $u_{i,4j+\alpha}$ and $u_{i,4j+\beta}$ are in $U$ and are the only vertices of $Q_{i,j}$ neighbors of the vertices of $K_j^{l^*}$; it remains true whether $\sigma_{l^*}$ sets value 1, 2, 3, 4 or 5 to $x_i$ with the convenient $\alpha$ and $\beta$. So all the neighbors of $K_j^{l^*}$ in the main part of the graph are not in $D$. Moreover, no vertex of $K_j$ is taken in the solution, and no vertex of $L_j \setminus L_j^{l^*}$ is taken in the solution. By these facts, and since the only edges between $L_j^{l^*}$ and $K_j^{l^*}$ form a perfect matching between the vertices of these two sets, it follows that each vertex of $L_j^{l^*}$ has a private neighbor, namely its unique neighbor in $K_j^{l^*}$.

Consider now any $i \in \{1, \ldots, n\}$. The set $U$ never takes three consecutive vertices in the path $P_i$, so $(V(P_i) \setminus \{u_{i,-3}, u_{i,-2}, u_{i,-1}, u_{i,4Fm}, u_{i,4Fm+1}, u_{i,4Fm+2}\}) \setminus U$ is a dominating set in the path $(V(P_i) \setminus \{u_{i,-3}, u_{i,-2}, u_{i,-1}, u_{i,4Fm}, u_{i,4Fm+1}, u_{i,4Fm+2}\})$. Observe now that, for any $j \in \{0, \ldots, Fm-1\}$, the vertices of the clique $K_j$ in the gadget $H_j$ are never taken by the solution, so the vertices of the path $P_i$ are only dominated by the vertices of $P_i$, whether the variable $x_i$ appears in $c_{j'}$ or not (for $j' = j \mod m$). Moreover, by the same argument, the neighbors in the verification gadgets of the vertices of the path $P_i$ taken in the solution are never taken in the solution.

If $\rho(x_i) \in \{0, 1, 2, 3\}$, then $U$ takes two consecutive vertices, leaves two consecutive vertices in $D$, takes again two consecutive vertices, and so on. In these cases, the two vertices of $d$ each have a private neighbor, namely their other neighbor in the path. If $\rho(x_i) \in \{4, 5\}$, then $U$ takes a vertex, leaves a vertex in $D$, takes a vertex, and so on. In these cases, the vertices of $D$ are their own private vertex. So all the vertices of the paths either have a private neighbor, or are their own private vertices.

Nonetheless, we have to be more careful for the first and last sections (for $j = 0$ and $j = Fm - 1$). By Step 3 of our construction of the solution $D$, and by some simple observations, we have that all vertices of the main part are dominated, and that the vertices of the main part which belong to the solution either have a private neighbor in the corresponding path, or are their own private vertices. □

Let us now prove the other direction of our reduction. The idea of this proof is the following: by partitioning the graph into different parts and upper bounding the cost of these parts, we prove that, if an upper dominating set in $G$ has not the same form as in Lemma 4.10 in a sufficiently large copy, then it has size strictly less than $k$, enabling us to produce a satisfiable assignment for $\varphi$ using the copy where the upper dominating set has the desired form.

**Lemma 4.11.** *If there exists an upper dominating set of size at least $k$ in $G$, then $\varphi$ is satisfiable.*

*Proof.* Suppose that we are given an upper dominating set $D$ of maximum size. Before we proceed any further, let us define, for each $S \subseteq V$, its *cost* as $\text{cost}(S) = |S \cap D|$. Clearly, $\text{cost}(V) \geq k$. Also, for two disjoint sets $S_1$ and $S_2$, we have $\text{cost}(S_1 \cup S_2) = \text{cost}(S_1) + \text{cost}(S_2)$. Our strategy will therefore be to partition $V$ into different parts and upper bound their cost.

Let $V_j = H_j \cup \bigcup_{i=1}^{n} Q_{i,j}$ with $Q_{i,j} = \{u_{i,4j}, u_{i,4j+1}, u_{i,4j+2}, u_{i,4j+3}\}$.

**Claim 4.12.** $\text{cost}(V_j) \leq 2n + A$.

*Proof.* Consider any $j \in \{0, \ldots, Fm - 1\}$, and let $j' = j \mod m$. We will prove that $\text{cost}(H_j) \leq A$. Note that the vertex $w$ has to be dominated, so either it is in $D$, or at least one vertex of $L_j$ is in $D$.

First, suppose that the vertex $w$ belongs to $D$. No vertex of $L_j$ can be in $D$, because otherwise $w$ has no private neighbor and is the neighbor of another vertex of $D$. Moreover, since $L_j$ is dominated, either only one vertex of $K_j$ belongs to $D$ and all the other vertices of $K_j$ can be its private neighbor, and in this case, the desired bound is obtained, or more that one vertex of $K_j$ belongs to $D$. In this last case, since $K_j$ is a clique, and since $L_j$ is dominated, the vertices in $D \cap K_j$ must have their private neighbor

in the main part of the graph. Note first that, for any $l \in \{1, \ldots, C_{j'}\}$, it cannot be the case that two vertices of $K_j^l$ are in $D$, since they share the same neighborhood. So the vertices of $K_j$ that belong to $D$ are in at least two distinct cliques $K_j^{l_1}$ and $K_j^{l_2}$ (for $l_1, l_2 \in \{1, \ldots, C_{j'}\}$ and $l_1 \neq l_2$). Note that, for any $i \in \{1, \ldots, n\}$ such that $x_i$ is involved in $c_{j'}$, any vertex of $K_j$ is connected to two vertices of $Q_{i,j}$ (for $Q_{i,j} = \{u_{i,4j}, u_{i,4j+1}, u_{i,4j+2}, u_{i,4j+3}\}$). So it cannot be the case that three vertices of $K_j$ are in $D$, because it would imply that one of them has no private neighbor. So if the vertex $w$ is in $D$, then we have $\text{cost}(H_j) \leq 3$.

Let us now consider the case where $w$ does not belong to $D$. Suppose now that there exist $l_1, l_2 \in \{1, \ldots, C_{j'}\}$ with $l_1 \neq l_2$ such that at least two vertices of $L_j^{l_1}$, let us say $v_1$ and $v_1'$, and at least one vertex of $L_j^{l_2}$, let say $v_2$, belong to $D$. Note that, since $L_j \cup \{w\}$ is a clique, the three vertices $v_1$, $v_1'$ and $v_2$ must have, each of them, a private neighbor in $K_j$. Now observe that all the vertices of $L_j^{l_1}$ are connected to all vertices of $K_j \setminus K_j^{l_1}$, so the private neighbors of $v_1$ and $v_1'$ must belong to $K_j^{l_1}$. But the vertex $v_2$ is connected to all vertices of $K_j^{l_1}$, since all vertices of $L_j^{l_2}$ are, which implies that $v_1$ and $v_1'$ have no private neighbor. So it cannot be the case that at least two vertices of $L_j^{l_1}$ and at least one vertex of $L_j^{l_2}$ are in $D$, for any $l_1, l_2$.

Suppose now that there exist $l_1, l_2, l_3 \in \{1, \ldots, C_{j'}\}$ with $l_1 \neq l_2 \neq l_3$ such that one vertex of $L_j^{l_1}$, let us say $v_1$, one vertex of $L_j^{l_2}$, let us say $v_2$, and one vertex of $L_j^{l_3}$, let us say $v_3$, are in $D$. By a similar argument, we have that the private neighbor of $v_1$ has to be in $K_j^{l_1}$: it cannot be in $K_j^{l_2}$ since all vertices of $K_j^{l_2}$ are connected to $v_1$ and $v_3$; it cannot be in $K_j^{l_3}$ since all vertices of $K_j^{l_3}$ are connected to $v_1$ and $v_2$; and it cannot be in any other $K_j^{l'}$ (for $l' \neq l_1, l_2, l_3$) since the vertices of $K_j^{l'}$ are connected to $v_1$, $v_2$ and $v_3$. But observe that all the vertices of $K_j^{l_1}$ are connected to $v_2$ and $v_3$, which implies that $v_1$ has no private neighbor. So it cannot be the case that one vertex of $L_j^{l_1}$, one vertex of $L_j^{l_2}$ and one vertex of $L_j^{l_3}$, are in $D$, for any $l_1, l_2, l_3$.

So, by these arguments, we have that at most $A$ vertices of $L_j \cup \{w\}$ belong to $D$, i.e., the $A$ vertices of a single clique $L_j^l$ (for $l \in \{1, \ldots, C_{j'}\}$). Now, suppose that there exists $l \in \{1, \ldots, C_{j'}\}$ such that $D \cap L_j = L_j^l$. The private neighbors of these vertices taken in $D$ must be in $K_j^l$, which implies that no vertex of $K_j$ can be in $D$. It follows that $\text{cost}(H_j) \leq A$, and this bound is attained if there exists an $l \in \{1, \ldots, C_{j'}\}$ such that $L_j^l \subseteq D$ and such that the vertices of $K_j^l$ are only dominated by the vertices of $L_j^l$.

Now, consider any $j \in \{0, \ldots, Fm - 1\}$ and any $i \in \{1, \ldots, n\}$ such that variable $x_i$ is involved in $c_{j'}$, for $j' = j \mod m$. Suppose that at least three vertices of $Q_{i,j}$ are in $D$, where we recall $Q_{i,j} = \{u_{i,4j}, u_{i,4j+1}, u_{i,4j+2}, u_{i,4j+3}\}$. Then all vertices of $K_j$ are dominated, since every vertex of $K_j$ is connected

to two vertices of $Q_{i,j}$. From this it follows that at most one vertex of $H_j$ is in $D$, since $L_j \cup \{w\}$ is a clique. Let $W_j = H_j \cup \bigcup_{x_i \text{active}} Q_{i,j}$. We have $\text{cost}(W_j) \leq 4q + 1$. We construct another solution by doing the following: consider a satisfying assignment $\sigma_l$ in the list of $c_{j'}$ and take all vertices of the clique $L_j^l$; plus take all the vertices of $Q_{i,j}$ not neighbors of the vertices of $K_j^l$, for any active variable $x_i$; and modify the solution to obtain an upper dominating set. Clearly, it gives us a valid solution. Moreover, this has increased the total cost. Indeed, we lose at most $4q + 1$ vertices: at most 2 vertices per $Q_{i,j}$ if the original solution had taken the four vertices; at most the two vertices $u_{i,4j-1}$ and $u_{i,4(j+1)}$, for each $x_i$ active, in order to keep the solution valid; and the vertex of $L_j \cup \{w\}$. On the other side, we have added $4q+2$ vertices: the $A = 4q+2$ vertices of $L_j^l$. Doing so should not be possible since $D$ is of maximum size, so for any active variable $x_i$, at most two vertices of $Q_{i,j}$ belong to $D$.

Now, consider any $j \in \{0, \ldots, Fm - 1\}$ and any $i \in \{1, \ldots, n\}$ such that variable $x_i$ is not involved in $c_{j'}$, for $j' = j \mod m$. Observe that, since the vertices of $Q_{i,j}$ are not connected to any verification gadget, it cannot be the case that three vertices of $Q_{i,j}$ belong to $D$, because otherwise at least one of them would be neighbor of another vertex of $D$ and would have no private neighbor.

We now have all the lower bounds we need: $\text{cost}(H_j) \leq A$; and $\text{cost}(Q_{i,j}) \leq 2$, whether $x_i$ is active or not. For each $j \in \{0, \ldots, Fm - 1\}$, recall that $V_j = H_j \cup \bigcup_{i=1}^{n} Q_{i,j}$. We have $\text{cost}(V_j) \leq 2n + A$. □

We will say that $j$ is *problematic* if $\text{cost}(V_j) < 2n + A$.

**Claim 4.13.** *There exists a contiguous interval $K \subseteq \{0, \ldots, Fm-1\}$ of size at least $m(4n + 1)$ in which all $j \in K$ are not problematic.*

*Proof.* Now, consider any $i \in \{1, \ldots, n\}$ and observe that among the three vertices $u_{i,-3}, u_{i,-2}$ and $u_{i,-1}$, at most two vertices can be in $D$, because otherwise the vertex $u_{i,-3}$ has no private neighbor. The same observation holds for the last three vertices $u_{i,4Fm}, u_{i,4Fm+1}$ and $u_{i,4Fm+2}$.

Let $L \subseteq \{0, \ldots, Fm - 1\}$ be the set of problematic indices. We claim that $|L| \leq 2n$. Indeed, we have $\text{cost}(V) \leq \sum_{j=0}^{Fm-1} \text{cost}(V_j) + 4n \leq Fm(2n + A) - |L| + 4n = k + 2n - |L|$. But, since the total cost is at least $k$, we have $|L| \leq 2n$. Now consider the longest contiguous interval $K \subseteq \{0, \ldots, Fm-1\}$ such that all $j \in K$ are not problematic. Since $F = (2n + 1)(4n + 1)$, we have $K \geq Fm/(|L| + 1) = m(4n + 1)$. □

Before we proceed further, note that, if $j$ is not problematic, then we have the following: $\text{cost}(H_j) = A$, which implies that there exists $l \in \{1, \ldots, C_{j'}\}$

such that $L_j^l \subseteq D$ and such that the vertices of $K_j^l$ are only dominated by $L_j^l$; for any $i \in \{1, \ldots, n\}$, $\text{cost}(Q_{i,j}) = 2$, so exactly two vertices in $Q_{i,j}$ are in $D$, and these two vertices are not connected to the vertices of $K_j^l$ (since this set is only dominated by $L_j^l$).

Consider now a non-problematic $j \in K$ and $i \in \{1, \ldots, n\}$. Since $\text{cost}(Q_{i,j}) = 2$, we claim that the solution must follow one of the six following configurations below:

(a) $u_{i,4j}, u_{i,4j+1} \in D$;

(b) $u_{i,4j+1}, u_{i,4j+2} \in D$;

(c) $u_{i,4j+2}, u_{i,4j+3} \in D$;

(d) $u_{i,4j+3}, u_{i,4j} \in D$;

(e) $u_{i,4j}, u_{i,4j+2} \in D$;

(f) $u_{i,4j+1}, u_{i,4j+3} \in D$.

Indeed, these six configurations cover all the cases where exactly two vertices of $Q_{i,j}$ are in $D$ (since $cost(Q_{i,j}) = 2$).

Now we make the following observations. For any $j \in K$ and any $i \in \{1, \ldots, n\}$, the vertices of $Q_{i,j}$ which are not in $D$ are only dominated by the vertices of the main part. Firstly, it is obvious if $x_i$ is not active in $c_{j'}$ (for $j' = j \mod m$) since in this case the vertices of $Q_{i,j}$ are not connected to any verification gadget. If $x_i$ is active in $c_{j'}$, it is also clear when we note that no vertex of $K_j$ is taken in the solution (since $\text{cost}(H_j) = A$). Moreover, the vertices of $Q_{i,j}$ which are in $D$ are not neighbors of vertices in $D$ outside the main part. It is again obvious if $x_i$ is not active in $c_{j'}$. If $x_i$ is active in $c_{j'}$, it is also clear since no vertex of $K_j$ is taken in $D$. Furthermore, the neighbors in the verification gadgets of the vertices of $Q_{i,j}$ not in the solution are all dominated by the vertices of $L_j^l$. From these observations, we obtain the following: the vertices of $Q_{i,j}$ which are in $D$ must have a private neighbor in the path $P_i$ or must be their own private vertex; and the vertices of $Q_{i,j}$ which are not in $D$ must be dominated by the vertices in the path $P_i$.

Now, we claim the following:

**Claim 4.14.** *There exists a contiguous interval $K' \subseteq \{0, \ldots, Fm-1\}$ of size at least $m$ in which all $j \in K'$ are not problematic and, for all $j_1, j_2 \in K'$, $Q_{i,j_1}$ and $Q_{i,j_2}$ are in the same configuration.*

*Proof.* Now, given the last observations, and the six configurations given before, we make the following statements, where a statement apply for any $i \in \{1, \ldots, n\}$ and $j$ such that $j$ and $j+1$ are in $K$:

- If $Q_{i,j}$ is in configuration (a), then $Q_{i,j+1}$ is in configuration (a), (d) or (e)
- If $Q_{i,j}$ is in configuration (b), then $Q_{i,j+1}$ is in configuration (b) or (f)
- If $Q_{i,j}$ is in configuration (c), then $Q_{i,j+1}$ is in configuration (c)
- If $Q_{i,j}$ is in configuration (d), then $Q_{i,j+1}$ is in configuration (c), (d) or (f)
- If $Q_{i,j}$ is in configuration (e), then $Q_{i,j+1}$ is in configuration (b), (d), (e) or (f)
- If $Q_{i,j}$ is in configuration (f), then $Q_{i,j+1}$ is in configuration (c) or (f)

For the first statement, we have the following: (b), (c) and (f) cannot follow (a) since it would left at least one vertex not dominated. For the second statement, we have the following: (a), (d) and (e) cannot follow (b) since at least one vertex will not have a private neighbor; (c) cannot follow (b) since it would left a vertex non dominated. For the third statement, we have the following: (a), (b), (d), (e) and (f) cannot follow (c) since at least one vertex will not have a private neighbor. For the fourth statement, we have the following: (a), (b) and (e) cannot follow (d) since at least one vertex will not have a private neighbor. For the fifth statement, we have the following: (a) cannot follow (e) since at least one vertex will not have a private neighbor; (c) cannot follow (e) since it would left a vertex non dominated. For the last statement, we have the following: (a), (b), (d) and (e) cannot follow (f) since at least one vertex will not have a private neighbor.

For some $i \in \{1, \ldots, n\}$ and $j \in K$, we will say that $j$ is *shifted* for variable $i$ if $j + 1 \in K$ but $Q_{i,j}$ and $Q_{i,j+1}$ are not in the same configuration. We observe that there cannot exist distinct $j_1, j_2, j_3, j_4, j_5 \in K$ such that they are all shifted for variable $i$. Indeed, if we draw a directed graph with a vertex for each configuration and an arc $(u, v)$ expressing the property that the configuration represented by $v$ can follow the configuration represented by $u$, then we observe that the graph obtained is a DAG of maximum path length 4.

Then, by the above, the number of shifted indices $j \in K$ is at most $4n$. Hence, the longest contiguous interval without shifted indices has length at least $|K|/(4n + 1) \geq m$, since $|K| \geq m(4n + 1)$. Let $K'$ be this interval. □

We have located an interval $K' \subseteq \{0, \ldots, Fm - 1\}$ of length at least $m$ where, for all $i \in \{1, \ldots, n\}$ and all $j_1, j_2 \in K'$, we have the same configuration in $Q_{i,j_1}$ and $Q_{i,j_2}$. We now extract a satisfying assignment for $\varphi$ from

this in the natural way. For some $j \in K'$: if $Q_{i,j}$ is in configuration (a), then we set $x_i = 0$; if $Q_{i,j}$ is in configuration (b), then we set $x_i = 1$; if $Q_{i,j}$ is in configuration (c), then we set $x_i = 2$; if $Q_{i,j}$ is in configuration (d), then we set $x_i = 3$; if $Q_{i,j}$ is in configuration (e), then we set $x_i = 4$; if $Q_{i,j}$ is in configuration (f), then we set $x_i = 5$. We claim this satisfies $\varphi$. Consider a constraint $c_{j'}$ of $\varphi$. There must exist $j \in K'$ such that $j' = j \mod m$ since $|K'| \geq m$ and $K'$ is contiguous. We therefore check the verification gadget $H_j$, where there exists $\sigma_l$ such that $L_j^l \subseteq D$ (this is because $j$ is not problematic, that is, $H_j$ attains its maximum cost). But because the vertices of $K_j^l$ are only dominated by the vertices $L_j^l$ and not by the vertices of the main part, it must be the case that the assignment we extracted agrees with $\sigma_l$, hence $c_{j'}$ is satisfied. This is true for all constraint $c_{j'}$ of $\varphi$. □

We now show that the pathwidth of $G$ is bounded by $n + O(1)$.

**Lemma 4.15.** *The pathwidth of $G$ is at most $n + O(1)$.*

*Proof.* We will show how to build a path decomposition of $G$. As in Lemma 4.11, for all $j \in \{0, \ldots, Fm - 1\}$, let $V_j = H_j \cup \bigcup_{i=0}^{n} Q_{i,j}$, where $Q_{i,j} = \{u_{i,4j}, u_{i,4j+1}, u_{i,4j+2}, u_{i,4j+3}\}$. We will show how to obtain a path decomposition of $G[V_j]$ with the following properties:

- The first bag of the decomposition contains the vertices $u_{i,4j}$, for all $i \in \{1, \ldots, n\}$
- The last bag of the decomposition contains the vertices $u_{i,4j+3}$, for all $i \in \{1, \ldots, n\}$
- The width of the decomposition is $n + O(q6^q)$

We now show how to obtain such a decomposition of $G[V_j]$, having partially fixed the contents of the first and last bag of the decomposition. The verification gadget $H_j$ contains at most $2(4q + 2)(6^q - 1) + 1$ vertices (since $6^q - 1$ is an upper bound on the number of assignments in the list of the corresponding constraint), so we place all its vertices in all bags. The remaining graph is a union of paths of length 4. We therefore have a sequence of $O(n)$ bags, where, for each $i \in \{1, \ldots, n\}$, we add to the current bag the vertices of $Q_{i,j}$ and then we add another bag with $Q_{i,j}$ removed except for $u_{i,4j+3}$.

Now that we have found a path decomposition of $G[V_j]$ with the desired properties, we present how to obtain a path decomposition of the whole graph. The sets $V_j$ partition all remaining vertices of the graph (except the first three vertices and the last three vertices of each path $P_i$), while the only edges not covered by the above decompositions of $G[V_j]$ are those between

the vertices $u_{i,4j+3}$ and $u_{i,4(j+1)}$. We therefore place the decompositions of $G[V_j]$ in order, and then, between the last bag of the decomposition of $G[V_j]$ and the first bag of the decomposition of $G[V_{j+1}]$, we have $2n$ "transition" bags, where in each transition step we add a vertex $u_{i,4(j+1)}$ in the bag, and then remove the corresponding vertex $u_{i,4j+3}$.

We have now a path decomposition of the whole graph except the first three and the last three vertices of each path $P_i$, for all $i \in \{1, \ldots, n\}$. So, before the first bag of the decomposition of $G[V_0]$, we have a sequence of $O(n)$ bags, where, for each $i \in \{1, \ldots, n\}$, we add to the current bag the four vertices $u_{i,-3}, u_{i,-2}, u_{i,-1}$ and $u_{i,0}$ and then we add another bag with only the vertex $u_{i,0}$. We use the same method for the last three vertices of the paths $P_i$, after the decomposition of $G[V_{Fm-1}]$.

Thus, we obtain a path decomposition of width $n + O(1)$. □

We are now ready to present the main result of this Section:

**Theorem 4.16.** *Under the SETH, for all $\varepsilon > 0$, no algorithm solves* Upper Dominating Set *in time $O^*((6-\varepsilon)^{pw})$, where $pw$ is the pathwidth of the input graph.*

*Proof.* Fix $\varepsilon > 0$ and let $q$ be sufficiently large so that Lemma 4.9 is true. Consider an instance $\varphi$ of $q$-CSP-6. Apply our reduction to obtain an instance $(G, k)$ of Upper Dominating Set. Thanks to Lemmas 4.10 and 4.11, we know that $\varphi$ is satisfiable if and only if there exists an upper dominating set of size at least $k$ in $G$.

Now suppose that there exists an algorithm that solves Upper Dominating Set in time $O^*((6-\varepsilon)^{pw})$. With this algorithm and our reduction, we can determine if $\varphi$ is satisfiable in time $O^*((6-\varepsilon)^{pw})$, where $pw = n + O(1)$ (Lemma 4.15), so the total running time of this procedure is at most $O^*((6-\varepsilon)^n)$, contradicting the SETH. □

With this theorem, we proved that our algorithm parameterized by the pathwidth for Upper Dominating Set running in time $O^*(6^{pw})$ is optimal under the SETH. In the next two sections, we focus on super-polynomial approximation.

## 4.4 Super-Polynomial Approximation

In this section, we present a super-polynomial approximation algorithm for the Upper Dominating Set problem. We prove the following: for any $r \leq n$, there exists an $r$-approximation algorithm for the Upper Dominating Set problem running in time $n^{O(n/r)}$.

Let us first give an overview of our algorithm. We use a common tool to design sub-exponential algorithms: the divide-and-conquer method where we partition the set of vertices $V(G)$ of the input graph into a convenient number of subsets of the same size. On each subset, we create a number of solutions: all maximal independent sets $I$ in the subgraph induced by the considered set of vertices, similarly to the super-polynomial approximation algorithm for Max Min Vertex Cover we have presented in Section 1.5; and all subsets $S$ of the considered subset of the partition. For each maximal independent set $I$, we extend it to the whole graph. For each subset $S$ as defined above, we first go through all subsets of neighbors of vertices of $S$ in order to find the correct set of private neighbors, and then we extend the solution to the whole graph. At the end, we output the best solution encountered. By computing all maximal independent sets $I$ and by going through all subsets $S$, we prove that there exists at least one valid upper dominating set which has the desired size. Note that, given a subset of an upper dominating set whose vertices have private neighbors, it may be impossible to extend the partial solution if we do not know their private vertices. This is why we need to find the private vertices of the subset $S$ we consider, since in our proof the solution which has the desired size may come from such a subset $S$. Interestingly, the two steps of our algorithm correspond to the two possible private vertices of the mandatory private structure of Upper Dominating Set: either a vertex $u$ taken in the solution is its own private vertex, corresponding to the independent sets $I$ we extend in the first step; or a vertex $u$ taken in the solution has a private neighbor, corresponding to the subsets $S$ for which we "guess" their private neighbors. We prove the following:

**Theorem 4.17.** *For any $r \leq n$, Upper Dominating Set is $r$-approximable in time $n^{O(n/r)}$.*

*Proof.* Let $D^* = S^* \cup I^*$ be any maximum upper dominating set of $G$, where $S^*$ is the set of vertices of $D^*$ which have some private neighbors, and $I^*$ is the set of vertices of $D^*$ which forms an independent set.

We begin our algorithm by partitioning the set of vertices $V(G)$ into $l$ subsets $V_1, \ldots, V_l$, where $l = \lfloor \frac{r}{2} \rfloor$.

Now, for each $i \in \{1, \ldots, l\}$, we do the following:

1. Enumerate all maximal independent sets of $G[V_i]$. Let $\mathcal{I}_i$ be this family of independent sets.

2. For each maximal independent set $I \in \mathcal{I}_i$, do the following:

   (a) Extend $I$ greedily to obtain a maximal independent set $I'$ of the whole graph $G$, in the natural way: while there exists a vertex $u \in V \setminus N[I']$, add $u$ to $I'$.

3. Consider all subsets of vertices $S$ of $V_i$.

4. For each such subset $S \subseteq V_i$, do the following:

   (a) For each vertex $u \in S$, go trough all vertices $v \in N(u) \setminus N[S]$ so that the vertex $v$ is the private neighbor of $u$.

   (b) Let $P$ be the set of private neighbors of the vertices of $S$ found in the previous step, if such a set exists.

   (c) Let $N_{SP} = N(S) \cap N(P)$, $N_S = N(S) \setminus N_{SP}$, $N_P = N(P) \setminus N_{SP}$, $V_{SP} = V \setminus (N[S] \cup N[P])$, and $Q_P = N_P \setminus N(V_{SP})$.

   (d) We extend the partial solution $S$ as follows:

      i. Let $T_1 = N(Q_P) \cap N_S$.

      ii. Greedily remove vertices of $T_1$ which have not a private neighbor in $N_P$, that is vertices $u \in T_1$ such that $(N(u) \cap N_P) \subseteq N(T_1 \setminus \{u\})$.

      iii. Let $T_2 = N(N_P \setminus N(T_1)) \cap V_{SP}$.

      iv. Greedily remove vertices of $T_2$ which have not a private neighbor in $N_P \setminus N(T_1)$, that is vertices $u \in T_2$ such that $(N(u) \cap (N_P \setminus N(T_1))) \subseteq N(T_2 \setminus \{u\})$.

      v. Extend $S \cup T_1 \cup T_2$ greedily to obtain an upper dominating set $S'$ of the whole graph $G$, in the natural way: while there exists a vertex $u \in V_{SP} \setminus N[T_2]$, add $u$ to $S'$.

      vi. Discard $S'$ if it is not an upper dominating set of $G$.

5. Output the solution of maximum size encountered.

We first prove that our algorithm has the desired running-time. For each $i \in \{1, \ldots, l\}$, the set $V_i$ is of size roughly $\frac{n}{l} = 2n/r$, so we have that enumerating all maximal independent sets of $G[V_i]$ takes time $O^*(3^{2n/3r})$, by the well-known result of Moon and Moser [MM65] which states that computing all maximal independent sets of a graph of order $n$ can be done in time $O^*(3^{n/3})$. Moreover, by the same upper-bound on the size of the set $V_i$, we have that considering all subsets $S \subseteq V_i$ takes time $2^{2n/r}$, and there are that many subsets $S$. Now, observe that at Step 4.(a), for a vertex $u \in S$, we go through all vertices $v \in N(u) \setminus N[S]$, so through at most $n$ vertices, and that there are at most $2n/r$ such vertices $u \in S$. So, for a subset $S \subseteq V_i$, we consider at most $n^{2n/r}$ sets $P$ of private neighbors of the vertices of $S$. Note that the other steps of our algorithm can be done in polynomial time. So the total running-time of our algorithm is:

$$l \cdot (O^*(3^{2n/3r}) + 2^{2n/r} \cdot n^{2n/r}) = n^{O(n/r)}$$

Now, we will prove that our algorithm outputs an upper dominating set. Consider any $i \in \{1, \ldots, l\}$ and any maximal independent set $I \in \mathcal{I}_i$ of $G[V_i]$. Note that, since $I$ is a maximal independent set of $G[V_i]$, it can be easily extended to obtain a maximal independent set $I'$ of the whole graph $G$. Indeed, by greedily adding vertices of $V \setminus N[I']$, we obtain at the end of Step 2.(a) a maximal independent set of $G$, since every vertex of $V(G)$ is either in $I'$ or has a neighbor in $I'$. Note that, since $I'$ is maximal, it is also an upper dominating set: all vertices of $V(G)$ are dominated and the vertices of $I'$ form an independent set. So all the independent set $I'$ for all $i$ are valid upper dominating sets of the graph $G$.

Consider any $i \in \{1, \ldots, l\}$. For the sets $S'$ constructed at Step 4 of our algorithm, we will show that at least one of them is an upper dominating set of $G$. Since we consider all subsets $S$ of $V_i$, we consider the set $S_i^* = S^* \cap V_i$. Then, for each vertex $u$ in this set $S_i^*$, we consider all its neighbors in $N(u) \setminus N[S_i^*]$ to be its private neighbor. So we consider the set $P_i^*$ which contains the private neighbor $v$ for each vertex $u \in S_i^*$ associated to the optimal solution $D^*$. Observe that the sets $N_{S_i^* P_i^*}$ and $N_{S_i^*}$ are dominated by $S_i^*$. Now consider the vertices of the set $Q_{P_i^*}$: they are not neighbors of $V_{S_i^* P_i^*}$ by definition; and they cannot be dominated by $N(P_i^*)$ since this set contains only neighbors of the vertices of $P_i^*$ and the vertices of $P_i^*$ are the private neighbors of the vertices of $S_i^*$. So the vertices of $Q_{P_i^*}$ can only be dominated by vertices of $N_{S_i^*}$. By our construction, the set $T_1$ is a set of vertices of $N_{S_i^*}$ which dominates $Q_{P_i^*}$ and such that each vertex $u \in T_1$ has a private neighbor. So the set $Q_{P_i^*}$ is dominated by $T_1$ and the vertices of $T_1$ each have at least one private neighbor (in $Q_{P_i^*}$ or in $N_{P_i^*} \setminus Q_{P_i^*}$). Now consider the vertices of the set $N_{P_i^*} \setminus N(T_1)$: they cannot be in the solution since they are neighbors of $P_i^*$; and they all have at least one neighbor in $V_{S_i^* P_i^*}$ (since they were not in $Q_{P_i^*}$). By our construction, the set $T_2$ is a set of vertices of $V_{S_i^* P_i^*}$ which dominates $N_{P_i^*} \setminus N(T_1)$ and such that each vertex $u \in T_2$ has a private neighbor in $N_{P_i^*} \setminus N(T_1)$: if a vertex of $T_2$ has no private neighbor in $N_{P_i^*} \setminus N(T_1)$, then it is removed from $T_2$ and $N_{P_i^*} \setminus N(T_1)$ stay dominated. Now observe that all vertices of $T_2$ have their private neighbor in $N_{P_i^*} \setminus N(T_1)$. So we can greedily extend $S_i^* \cup T_1 \cup T_2$ in a maximal independent set fashion by adding vertices of $V_{S_i^* P_i^*} \setminus N[T_2]$ until the whole graph $G$ becomes dominated. So the set $S_i^{*\prime}$ obtained is an upper dominating set of $G$. So for any $i \in \{1, \ldots, l\}$, there exists at least one set $S'$ which is an upper dominating set of $G$, and the non-valid solutions are discarded at the end of Step 4.(d).

Thus, the algorithm always outputs an upper dominating set.

Now, we will prove the approximation ratio. Note first that, since we have partitioned $V(G)$ into $l = \lfloor \frac{r}{2} \rfloor$ almost equal-sized subsets $V_1, \ldots, V_l$,

there exists $i^* \in \{1, \ldots, l\}$ such that $|D^* \cap V_{i^*}| \geq |D^*|/l \geq 2|D^*|/r$. Consider the corresponding subset $V_{i^*}$. Note now that, since $D^* = S^* \cup I^*$, we have the following: either at least $|D^* \cap V_{i^*}|/2$ vertices of $D^* \cap V_{i^*}$ are in $I^*$, or at least $|D^* \cap V_{i^*}|/2$ vertices of $D^* \cap V_{i^*}$ are in $S^*$.

Suppose first that at least $|D^* \cap V_{i^*}|/2$ vertices of $D^* \cap V_{i^*}$ are in $I^*$. Since we have enumerated all maximal independent sets $I \in \mathcal{I}_{i^*}$ of $G[V_{i^*}]$, and since $I^* \cap V_{i^*}$ is an independent set of $V_{i^*}$, we have found at least one maximal independent set $I_{i^*}$ of $G[V_{i^*}]$ such that $I^* \cap V_{i^*} \subseteq I_{i^*}$. Then, we have extended $I_{i^*}$ to obtain a maximal independent set $I'_{i^*}$ of $G$. Thus, we have the following:

$$|I'_{i^*}| \geq |I_{i^*}| \geq |I^* \cap V_{i^*}| \geq |D^* \cap V_{i^*}|/2 \geq 2|D^*|/2r = |D^*|/r$$

But since our algorithm outputs the maximum-sized solution encountered, we have the desired approximation ratio in this case.

Suppose now that at least $|D^* \cap V_{i^*}|/2$ vertices of $D^* \cap V_{i^*}$ are in $S^*$. Since we have considered all subsets $S$ of $V_{i^*}$, we have considered the subset $S^*_i = S^* \cap V_{i^*}$. For this set, we have considered all possible sets of private neighbors of vertices of $S^*_i$, and we have extended the set to an upper dominating set $S^{*\prime}_i$ of $G$ (note that the set $S^*_i$ has been successfully extended since it is the set we have considered when we have proved that at least one set $S'$ constructed at Step 4 is a valid upper dominating set of $G$). Thus, we have the following:

$$|S^{*\prime}_i| \geq |S^*_i| = |S^* \cap V_{i^*}| \geq |D^* \cap V_{i^*}|/2 \geq 2|D^*|/2r = |D^*|/r$$

Again, since our algorithm outputs the maximum-sized solution encountered, we have the desired approximation ratio in this case also. □

Now that we have presented our super-polynomial approximation algorithm for Upper Dominating Set, we will in the following section prove that it is asymptotically optimal under the randomized ETH.

## 4.5 Matching Lower Bound

In this section, we give a lower bound on the complexity of any $r$-approximation algorithm for Upper Dominating Set, matching our algorithm of the previous section. We get the following result: for any $r < n$ and any $\varepsilon > 0$, there is no algorithm that outputs an $r$-approximation for the Upper Dominating Set problem running in time $n^{(n/r)^{1-\varepsilon}}$ under the randomized ETH.

To obtain this result, we will first prove the desired lower bound for the Max Min Hitting Set problem, similarly of what Bazgan et al. did

in [BBC+18a] to obtain the $n^{1-\varepsilon}$-inapproximability for Upper Dominating Set. To obtain this lower bound for the Max Min Hitting Set problem, we will do a reduction from the Max Independent Set problem. Then, we will make a reduction from the Max Min Hitting Set problem to the Upper Dominating Set problem to transfer this lower bound to our problem.

Recall that we have the following lower bound by Chalermsook et al. [CLN13] for the Max Independent Set problem, which is the starting point of our reduction, and which we have already used to derive the super-polynomial inapproximability of Max Min Feedback Vertex Set in Section 3.6:

**Theorem 4.18** (Theorem 1.2 from [CLN13])**.** *For any $\varepsilon > 0$ and any sufficiently large $r > 1$, if there exists an $r$-approximation algorithm for Maximum Independent Set running in time $2^{(n/r)^{1-\varepsilon}}$, then the randomized ETH is false.*

Our first reduction is similar to the reduction of Bazgan et al. [BBC+18a] from Max Independent Set to Max Min Hitting Set, and will allow us to get the desired hardness result for the latter problem. Our second reduction, from Max Min Hitting Set to Upper Dominating Set is the approximation-preserving reduction designed by Bazgan et al. [BBC+18a].

Nonetheless, note that our reduction from Max Independent Set to Max Min Hitting Set creates a quadratic blow-up of the size of the instance of the latter problem. Such a blow-up does not allow us to derive the desired running-time. To answer this difficulty, we make another step in the reduction where we "sparsify" the instance of Max Min Hitting Set in order to keep the blow-up under control. To prove that the inapproximability gap stays the same, we use a probabilistic analysis with Chernoff bounds, method we have already used for Max Min Feedback Vertex Set in Section 3.6.

We will first prove the following hardness result:

**Theorem 4.19.** *For any $\varepsilon > 0$ and any sufficiently large $r > 1$, if there exists an $r$-approximation algorithm for Max Min Hitting Set running in time $n^{(n/r)^{1-\varepsilon}}$, then the randomized ETH is false.*

*Proof.* First, we recall some details about Theorem 4.18. To get this result, Chalermsook et al. [CLN13] designed a reduction from an instance $\phi$ of 3-Sat with $n$ variables, and for any $\varepsilon > 0$ and $r$ sufficiently large, they construct a graph $G$ with $|V(G)| = n^{1+\varepsilon}r^{1+\varepsilon}$ vertices which, with high probability, satisfies the following properties:

- Yes-instance: if $\phi$ is satisfiable, then $\alpha(G) \geq n^{1+\varepsilon}r$
- No-instance: if $\phi$ is not satisfiable, then $\alpha(G) \leq n^{1+\varepsilon}r^{2\varepsilon}$.

With these properties, any approximation algorithm with ratio $r^{1-2\varepsilon}$ for Max Independent Set would distinguish whether $\phi$ is satisfiable or not, and so would solve the 3-Sat instance. If this algorithm runs in time $2^{(n/r)^{1-\varepsilon}}$, then we obtain a sub-exponential algorithm for 3-Sat, which contradicts the randomized ETH.

Suppose that we are given $\varepsilon > 0$ and $r$ sufficiently large. Let $d = \frac{1}{\varepsilon^{1/2}}$ We will also design a reduction from the instance $\phi$ of 3-Sat to an instance of Max Min Hitting Set going through an instance of Max Independent Set to show that an algorithm for the Max Min Hitting Set that achieves this ratio $r$ too rapidly would give a sub-exponential algorithm for 3-Sat. So we start with the reduction of [CLN13], from an instance $\phi$ of 3-Sat on $n$ variables, and we adjust the parameter $r$ so that we obtain with high probability a graph $G$ with the following properties:

- $|V(G)| = n^{1+\varepsilon}r^{1/d+\varepsilon/d}$
- Yes-instance: if $\phi$ is satisfiable, then $\alpha(G) \geq n^{1+\varepsilon}r^{1/d}$.
- No-instance: if $\phi$ is not satisfiable, then $\alpha(G) \leq n^{1+\varepsilon}r^{2\varepsilon/d}$.

We now construct an hypergraph $G'$ for the Max Min Hitting Set problem in the following way: we keep the graph $G$; for every subset $S \subseteq V(G)$ with $|S| = d$, we construct an independent set $Z_S$ of size $t = r^{1/d}$; and, for every vertex $u \in Z_S$, we add the hyper-edge $S \cup \{u\}$. Now, we claim that the graph $G'$ has the following properties:

- $|V(G')| = \Theta(n^{d+d\varepsilon}r^{1+1/d+\varepsilon})$
- Yes-instance: if $\phi$ is satisfiable, then $\text{mmhs}(G') = \Omega(n^{d+d\varepsilon}r^{1+1/d})$.
- No-instance: if $\phi$ is not satisfiable, then $\text{mmhs}(G') = O(n^{d+d\varepsilon}r^{1/d+2\varepsilon})$.

Here, $\text{mmhs}(G')$ is the maximum size of a minimal hitting set in $G'$.

Let us prove why the hypergraph $G'$ has these properties.

For the first property, note that there are $\binom{|V(G)|}{d}$ subsets $S$ of $V(G)$ of size $d$, and that for each of them we have added $t = r^{1/d}$ vertices in the corresponding independent set $Z_S$. So we have the following:

$$|V(G')| = t \cdot \binom{|V(G)|}{d} + |V(G)| = \Theta(n^{d+d\varepsilon}r^{1+1/d+\varepsilon})$$

For the second property, suppose that $\phi$ is satisfiable. It follows that $\alpha(G) \geq n^{1+\varepsilon} r^{1/d}$. We construct a minimal hitting set of $G'$ as follows: we take a minimum vertex cover $C$ of $G$; and for every subset $S$ of $I = V(G) \setminus C$ of size $d$, we take the $t$ vertices of the corresponding independent set $Z_S$. We observe that this solution is a minimal hitting set of $G'$. Indeed, $C$ is a minimum vertex cover of $G$, so all edges of $G$ are dominated by the solution, and every vertex of $C$ has at least one private edge since $C$ is a minimum vertex cover. Now observe that all the hyper-edges added in the construction of $G'$ which still have to be covered are hyper-edges between some vertices of the independent set $I = V(G) \setminus C$ and the corresponding independent sets $Z_S$, since all hyper-edges connected at least one vertex of $C$ are covered. But we took the $t$ vertices of the independent set $Z_S$ for every subset $S \subseteq I$ of size $d$, so it follows that all the remaining hyper-edges are covered by our solution. Moreover, for any subset $S$ of $I$ of size $d$, note that $S$ is an independent set, so every vertex $u$ of $Z_S$ taken has a private hyper-edge, namely the hyper-edge $S \cup \{u\}$. So our solution is a minimal hitting set. Now, let us determine its size. The number of independent sets $Z_S$ with $S \subseteq I$ of size $d$ is $\binom{\alpha(G)}{d}$. So the size of our solution is at least:

$$t \cdot \binom{\alpha(G)}{d} = \Omega(n^{d+d\varepsilon} r^{1+1/d})$$

For the third property, take any minimal hitting set in $G'$ and let $I$ be the corresponding independent set of $G$ ($I = V(G) \setminus C$ where $C$ is a vertex cover in $G$ which belongs to the minimal hitting set). We have that for any subset $S$ of $I$ of size $d$, the minimal hitting set takes at most the $t$ vertices of the independent set $Z_S$. And there are at most $\binom{\alpha(G)}{d}$ such subsets $S$. So the size of any minimal hitting set is bounded by:

$$t \cdot \binom{\alpha(G)}{d} + |V(G)| = O(n^{d+d\varepsilon} r^{1/d+2\varepsilon})$$

We have now constructed a graph $G'$ of the Max Min Hitting Set problem where the gap between the values of $\text{mmhs}(G')$, corresponds to whether $\phi$ is satisfiable or not, is smaller than $r$ (it is $r^{1-2\varepsilon}$). Nonetheless, we cannot derive the desired hardness result since the order of $G'$ is quadratic on $n$. This blow-up makes it impossible to derive a sub-exponential algorithm for 3-SAT. So we need to sparsify the gaph $G'$.

Thus, we construct a graph $G''$ in the following way: we keep the graph $G'$; and we delete every vertex of $V(G') \setminus V(G)$ with probability $\frac{n^d-1}{n^d}$. That is, for every vertex $u$ in an independent set $Z_S$, the vertex $u$ stays in $G''$ with probability $\frac{1}{n^d}$. We claim that the graph $G''$ has the following properties:

- $|V(G'')| = \Theta(n^{1+d\varepsilon} r^{1+1/d+\varepsilon})$
- Yes-instance: if $\phi$ is satisfiable, then $\text{mmhs}(G'') = \Omega(n^{1+d\varepsilon} r^{1+1/d})$.
- No-instance: if $\phi$ is not satisfiable, then $\text{mmhs}(G'') = O(n^{1+d\varepsilon} r^{1/d+2\varepsilon})$.

To establish these three properties, we will use the following Chernoff bound: suppose $X = \sum_{i=1}^{p} X_i$ is the sum of $p$ independent random 0/1 variables $X_i$ and that $E[X] = \sum_{i=1}^{p} E[X_i] = \mu$. We have the following: for all $0 \leq \delta \leq 1$, $Pr[|X - \mu| \geq \delta\mu] \leq 2e^{-\mu\delta^2/3}$.

For the first property, we begin by defining a random variable $X_i$ for each vertex of each independent set $Z_S$ of $G'$: $X_i = 1$ if the corresponding vertex stays in $G''$; and $X_i = 0$ otherwise. Let $X$ be the sum of these $X_i$ variables, which is equal to the number of such vertices staying in $G''$. Suppose now that the number of vertices in the sets $Z_S$ in $G'$ is $cn^{d+d\varepsilon} r^{1+1/d+\varepsilon}$, where $c$ is a constant (it follows from the size of $V(G')$). Then $E[X] = cn^{1+d\varepsilon} r^{1+1/d+\varepsilon}$. We obtain $Pr[|X - E[X]| \geq \frac{E[X]}{2}] \leq 2e^{-E[X]/12} = o(1)$. So we conclude with high probability that $|V(G'')| = \Theta(n^{1+d\varepsilon} r^{1+1/d+\varepsilon})$.

For the second property, we consider a minimal hitting set $F$ of $G'$, of size $cn^{d+d\varepsilon} r^{1+1/d}$. We define a variable for each vertex of $F$ in the independent sets $Z_S$. As in the previous paragraph, we have that the expected number of such vertices which stay in $G''$ is $cn^{1+d\varepsilon} r^{1+1/d}$. Again, as in the previous paragraph, the actual number of such vertices will be close to this bound. We just need to prove that almost the same set is a minimal hitting set of $G''$. So we begin with the surviving vertices of $F$, which is a hitting set of $G''$ (since the removal of a vertex of $F$ implies the removal of its incident hyper-edges). Now, we delete vertices from $F$ until we obtain a minimal hitting set of $G''$. We will prove that the number of vertices deleted as redundant is at most $|V(G)| = n^{1+\varepsilon} r^{1/d+\varepsilon/d}$. Consider first an independent set $Z_S$ such that $Z_S \cap F \neq \emptyset$. Since $Z_S \cap F \neq \emptyset$, it follows that the vertices of the set $S$ are not in the solution $F$, because otherwise the vertices of $Z_S \cap F$ would not have a private hyper-edge. But because $S \cap F = \emptyset$, the vertices of $Z_S \cap F$ cannot be considered as redundant, since for every vertex $u \in Z_S \cap F$, it covers the hyper-edge $S \cup \{u\}$. Thus, no vertex of the independent sets $Z_S$ can be removed as redundant. So the only vertices which can be removed as redundant are the vertices initially in $V(G)$. So at most $|V(G)| = n^{1+\varepsilon} r^{1/d+\varepsilon/d}$ vertices can be removed as redundant. Since $|V(G)| < c(n^{1+d\varepsilon} r^{1+1/d})$ (for $n$ and $r$ sufficiently large and a constant $c$), it follows that removing these redundant vertices will not change the order of magnitude of the solution in $G''$.

For the third property, we need to consider every possible minimal hitting set of $G''$ and prove that none of them is too large. So consider any subset $I \subseteq$

$V(G)$ being an independent set of $G$. Our goal is to prove that any minimal hitting set $F$ of $G''$ that satisfies $V(G) \setminus I \subseteq F$ has a probability of being too big smaller than $2^{-|V(G)|}$. Indeed, if we prove this, we can take the union bound over all sets $I$ and conclude that with high probability no minimal hitting set of $G''$ is too big. So suppose now that we have fixed an independent set $I \subseteq V(G)$. We have $|I| \leq \alpha(G) \leq n^{1+\varepsilon} r^{2\varepsilon/d}$. We now make the following observation: any minimal hitting set $F$ which satisfies $V(G) \setminus I \subseteq F$ cannot contain any vertex of a set $Z_S$ if $S \cap F \neq \emptyset$; but may contain the $t$ vertices of an independent set $Z_S$ if $S \cap F = \emptyset$. The total number of such vertices in $G'$ is $O(n^{d+d\varepsilon} r^{1/d+2\varepsilon})$, since it is an upper bound on $\text{mmhs}(G')$. So, by the same argument as in the previous paragraph, the expected number of such vertices which stay in $G''$ is at most $\mu = cn^{1+d\varepsilon} r^{1/d+2\varepsilon}$, for a constant $c$. By using the Chernoff bound, we have $Pr[|X - \mu| \geq \frac{\mu}{2}] \leq 2e^{-\mu/12}$. We claim that $2e^{-\mu/12} = o(2^{-|V(G)|})$. Indeed, it follows since $|V(G)| = n^{1+\varepsilon} r^{1/d+\varepsilon/d} = o(\mu)$. Thus, the probability that a minimal hitting set being too large exists for a fixed independent set $I \subseteq V(G)$ is low enough so that taking the union bound over all possible independent sets $I$ give a probability that at least one minimal hitting set is too big of value $o(1)$. So we have with high probability that no minimal hitting set of size greater than $3\mu/2$ exists. So we obtain the third property.

Now that we have proved that $G''$ satisfies these three properties, we will show how to obtain the theorem. Suppose that, for sufficiently large $r$ and any $\varepsilon > 0$, there exists an approximation algorithm for Max Min Hitting Set with ratio $r^{1-3\varepsilon}$ running in time $N^{(N/r)^{1-4\varepsilon}}$ for graphs of order $N$. The ratio of this algorithm is sufficiently small to distinguish between the two cases in our graph $G''$, as the ratio between $\text{mmhs}(G'')$ when $\phi$ is satisfiable or not is $\Omega(r^{1-2\varepsilon})$ (for $r$ sufficiently large). So we can use this approximation algorithm to solve 3-Sat. Furthermore, we have the following:

$$N/r = \Theta((n^{1+d\varepsilon} r^{1+1/d+\varepsilon})/r) = O(n^{1+(d+1)\varepsilon+1/d}) = O(n^{1+\varepsilon+2\varepsilon^{1/2}})$$

Therefore, $(N/r)^{1-4\varepsilon} = o(n)$. We obtain an algorithm for 3-Sat in time $N^{(N/r)^{1-\varepsilon}} = 2^{n^{1-\varepsilon'}}$ for $\varepsilon' < \varepsilon$ chosen appropriately. This contradicts the randomized ETH. So by adjusting $r$ and $\varepsilon$, we get that no $r$-approximation algorithm for Max Min Hitting Set can run in time $N^{(N/r)^{1-\varepsilon}}$ for graphs of order $N$. Thus we get the statement of the theorem. □

With this hardness result for Max Min Hitting Set, and by using the reduction of Bazgan et al. [BBC+18a], we get the following hardness result for Upper Dominating Set:

**Theorem 4.20.** *For any $\varepsilon > 0$ and any sufficiently large $r > 1$, if there exists an $r$-approximation for* Upper Dominating Set *running in time $n^{(n/r)^{1-\varepsilon}}$, then the randomized ETH is false.*

*Proof.* We start with an instance $G''$ of Max Min Hitting Set obtained from Theorem 4.19. From this instance, we construct an instance $G^*$ of Upper Dominating Set by the Theorem 12 of Bazgan et al. [BBC+18a], and we know that this reduction is approximation-preserving. So the gap from the hardness result of Max Min Hitting Set stays the same for Upper Dominating Set. Now observe that in Theorem 4.19, the number of hyper-edges in $G''$ has the same order of magnitude than the number of vertices in $G''$. Thus, the number of vertices in $G^*$ is linearly dependent on the number of vertices in $G''$. So an $r$-approximation algorithm for Upper Dominating Set running in time $n^{(n/r)^{1-\varepsilon}}$ would give an $r$-approximation algorithm for Max Min Hitting Set with the same running time, which would contradict Theorem 4.19 and the randomized ETH. So we obtain the desired hardness result for Upper Dominating Set. □

With the two Theorems 4.17 and 4.20, we have completely settled the super-polynomial approximation of Upper Dominating Set.

# Chapter 5

# Conclusion

We have begun by introducing the basic notions about three main frameworks to tackle **NP**-hard problems: the polynomial-time approximation, exact algorithms, and the super-polynomial approximation. Indeed, we have presented in the Introduction how to obtain algorithms in these frameworks, as well as intractability results under some complexity assumptions.

We have then studied three domination **NP**-hard problems, under the scope of private structure: Min Mixed Dominating Set, Max Min Feedback Vertex Set and Upper Dominating Set. For each of these problems, we have improved the state of the art in the different frameworks, either by giving faster algorithms, or by designing better algorithms for which we give a matching lower bound under some complexity assumptions showing that our algorithms are the best algorithms we can hope under these assumptions.

From these results, there are some questions that could be interesting to investigate:

- For Min Mixed Dominating Set:
  - Can we design FPT algorithms for this problem, with other parameters? Designing such an algorithm parameterized by the vertex cover $vc$ of the graph could be interesting due to the strong relationship between these two problems (see Lemma 2.4).
  - Can we improve the exact and FPT (parameterized by $k$) algorithms for this problem, maybe using the *measure-and-conquer* method?
  - Can we obtain a super-polynomial approximation algorithm for this problem matching the 2-approximation algorithm of Hatami [Hat07] and our exact or FPT algorithm?

- For Max Min Feedback Vertex Set:
  - Can we design FPT algorithms for this problem? For example parameterized by the treewidth $tw$ and using the *cut-and-count* method introduced by Cygan et al. [CNP+11] which gave a good algorithm for the Min version Min Feedback Vertex Set.
  - Can we improve the super-polynomial approximation of this problem from $n^{O(n/r^{3/2})}$ to $c^{O(n/r^{3/2})}$ for a constant $c$? Using an FPT algorithm parameterized by the treewidth could improve such an algorithm compared to our algorithm of Lemma 3.21 which outputs a constant factor approximation in exponential time of a given feedback vertex set.
  - Can we prove that the cubic kernel for this problem is the best we can hope for?
- For Upper Dominating Set:
  - Can we design a $O^*(6^{tw})$ FPT algorithm parameterized by the treewidth $tw$ for this problem? The current best algorithm of Bazgan et al. [BBC+18a] works in time $O^*(10^{tw})$ and could be improved, maybe using the *fast subset convolution* technique. Such a result would automatically be optimal due to our $O^*((6-\varepsilon)^{pw})$ intractability result under the SETH.
  - Can we improve the super-polynomial approximation of this problem from $n^{O(n/r)}$ to $c^{O(n/r)}$ for a constant $c$? Designing a subroutine algorithm which would be able to determine the private neighbors of a given set of vertices effectively would allow to obtain such a running time.

Finally, some more optimistic direction would be to study more general problems. For example, we could ask if we can design FPT algorithms parameterized by the treewidth $tw$ for a large class of problems as the Max-Min versions of domination-like problems? Recall that van Rooij et al. [vRBR09] have obtained such general algorithms for the $[\rho,\sigma]$-Dominating Set problems using the fast subset convolution technique. An interesting question would be determine if we can transfer such algorithms for the Max-Min and Min-Max versions of $[\rho,\sigma]$-Dominating Set. Nonetheless, in contrary to the connectivity versions of some **NP**-hard problems for which there exist FPT algorithms parameterized by the treewidth running in time $O^*((c+1)^{tw})$ compared to the $O^*(c^{tw})$ for the not connected version (see [CNP+11] for example), such an additive "plus one" in the base seems not to work for the

Max-Min and Min-Max versions of the $[\rho, \sigma]$-Dominating Set problems. Indeed, Max Independent Set can be solved in time $O^*(2^{tw})$ and its Min-Max version in time $O^*(3^{tw})$, while Min Dominating Set can be solved in time $O^*(3^{tw})$ and its Max-Min version in at least $O^*(6^{tw})$ (due to our intractability result under the SETH). Nonetheless, it could be interesting to focus on the Max-Min versions of the Min versions of $[\rho, \sigma]$-Dominating Set.

# Chapter 6

# Appendix

---

3-Sat:
INSTANCE: A 3-CNF formula $\phi$ (that is a CNF formula where each clause has at most 3 literals) over the variable set $X$.
QUESTION: Is there a truth assignment of the variables of $X$ that satisfies all clauses of $\phi$?

---

Clique:
INSTANCE: An undirected graph $G = (V, E)$, with $|V| = n$ and $|E| = m$, and an integer $k \leq n$.
QUESTION: Is there a clique, that is a set $K$ of vertices such that any two vertices of $K$ are neighbors, of size at least $k$ in $G$?

---

Dominating Set:
INSTANCE: An undirected graph $G = (V, E)$, with $|V| = n$ and $|E| = m$, and an integer $k \leq n$.
QUESTION: Is there a dominating set, that is a set $D$ of vertices such that for all vertices $u \in V$, there exists a vertex $v \in N[u] \cap D$, of size at most $k$?

---

Independent Set:
INSTANCE: An undirected graph $G = (V, E)$, with $|V| = n$ and $|E| = m$, and an integer $k \leq n$.
QUESTION: Is there an independent set of size at least $k$ in $G$?

---

$k$-SAT:
INSTANCE: A $k$-CNF formula $\phi$ (that is a CNF formula where each clause has at most $k$ literals) over the variable set $X$.
QUESTION: Is there a truth assignment of the variables of $X$ that satisfies all clauses of $\phi$ ?

---

MAX-3-SAT:
INSTANCE: A 3-CNF formula $\phi$ (that is a CNF formula where each clause has at most 3 literals) over the variable set $X$.
QUESTION: Determine the maximum number of clauses a truth assignment can satisfy in $\phi$.

---

MAX INDEPENDENT SET:
INSTANCE: An undirected graph $G = (V, E)$, with $|V| = n$ and $|E| = m$.
QUESTION: Determine the maximum size of an independent set in $G$, i.e., of a subset $I \subseteq V$ of vertices of maximum size such that for any two vertices $u, u' \in I$ there is no edge $(u, u') \in E$.

---

MAX MIN FEEDBACK VERTEX SET:
INSTANCE: A graph $G = (V, E)$, with $|V| = n$ and $|E| = m$.
QUESTION: determine the maximum size of a minimal feedback vertex set in $G$, i.e., of a subset $S \subseteq V$ of vertices of maximum size which forms a minimal feedback vertex set, that is a set $S$ such that for every cycle of $G$ at least one vertex from this cycle is in $S$, and which is inclusion-wise minimal that is if we remove a vertex from $S$ it is not a feedback vertex set anymore.

---

MAX MIN HITTING SET:
INSTANCE: An hypergraph $G = (V, A)$, with $|V| = n$ and $|E| = m$.
QUESTION: Determine the maximum size of a minimal hitting set in $G$, i.e., of subset $C \subseteq V$ of vertices of maximum size which forms a minimal hitting set, that is a set of vertices that cover all hyper-edges of $G$, and which is inclusion-wise minimal that is if we remove a vertex from $C$ it is not an hitting set anymore.

---

MAX MIN VERTEX COVER:
INSTANCE: A graph $G = (V, E)$, with $|V| = n$ and $|E| = m$.
QUESTION: Determine the maximum size of a minimal vertex cover in $G$, i.e., of a subset $C \subseteq V$ of vertices of maximum size which forms a minimal vertex cover, that is a subset $C$ of vertices such that all edges of $G$ are covered by the vertices of $C$, and which is inclusion-wise minimal that is if we remove a vertex from $C$ it is not a vertex cover anymore.

---

MIN DISTANCE-2-DOMINATING SET:
INSTANCE: A graph $G = (V, E)$, with $|V| = n$ and $|E| = m$.
QUESTION: Determine the minimum size of a distance-2-dominating set in $G$, i.e., of a set $D \subseteq V$ of vertices of minimum size which forms a distance-2-dominating set, that is a set of vertices $D$ such that every vertex $u \in V \setminus D$ is at distance at most 2 from a vertex of $D$.

---

MIN DOMINATING SET:
INSTANCE: An undirected graph $G = (V, E)$, with $|V| = n$ and $|E| = m$.
QUESTION: Determine the minimum size of a dominating set in $G$, i.e., of a set $D \subseteq V$ such that for all vertices $u \in V$, there exits a vertex $v \in N[u] \cap D$, of minimum size.

---

MIN EDGE COVER:
INSTANCE: A graph $G = (V, E)$, with $|V| = n$ and $|E| = m$.
QUESTION: Determine the minimum sie of an edge cover in $G$, i.e., of a subset $M \subseteq E$ of edges that dominates all vertices of $G$.

---

MIN EDGE DOMINATING SET:
INSTANCE: A graph $G = (V, E)$, with $|V| = n$ and $|E| = m$.
QUESTION: Determine the minimum size of an edge dominating set in $G$, i.e., of a subset $M \subseteq E$ of edges that dominates all edges of $G$.

---

MIN INDEPENDENT DOMINATING SET:
INSTANCE: An undirected graph $G = (V, E)$, with $|V| = n$ and $|E| = m$.
QUESTION: Determine the minimum size of an independent dominating set in $G$, i.e., of a subset $I \subseteq V$ of vertices of minimum size such that any two vertices of $I$ are not adjacent, and all vertices of $V$ have at least one neighbor in $S$.

---

MIN FEEDBACK VERTEX SET:
INSTANCE: A graph $G = (V, E)$, with $|V| = n$ and $|E| = m$.
QUESTION: Determine the minimum size of a feedback vertex set in $G$, i.e. of a subset $S \subseteq V$ of vertices of minimum size which forms a minimal feedback vertex set, that is a set $S$ such that for every cycle of $G$ at least one vertex from this cycle is in $S$.

---

MIN MAX MATCHING:
INSTANCE: An undirected graph $G = (V, E)$, with $|V| = n$ and $|E| = m$.
QUESTION: Determine the minimum size of a maximal matching in $G$, i.e., of a maximal matching, i.e. a set of edges $M$ such that each edge $e \in E$ is adjacent to an edge of $M$, of minimum size.

---

MIN MIXED DOMINATING SET:
INSTANCE: A graph $G = (V, E)$, with $|V| = n$ and $|E| = m$.
QUESTION: Determine the minimum size of a mixed dominating set in $G$, i.e. of a set $D \cup M$, that is a set of vertices $D \subseteq V$ and a set of edges $S \subseteq E$ such that $V \cup E$ is dominated, of minimum size.

---

MIN VERTEX COVER:
INSTANCE: An undirected graph $G = (V, E)$, with $|V| = n$ and $|E| = m$.
QUESTION: Determine the minimum size of a vertex cover in $G$, i.e., of a subset $C \subseteq V$ of vertices of minimum size such that for every edge $(u, v) \in E$, at least one of $u$ and $v$ is in $C$.

---

MIXED DOMINATING SET:
INSTANCE: A graph $G = (V, E)$, with $|V| = n$ and $|E| = m$, and an integer $k \leq n + m$.
QUESTION: Is there a mixed dominating set $D \cup M$, that is a set of vertices $D \subseteq V$ and a set of edges $M \subseteq E$ such that $V \cup E$ is dominated, of size at most $k$ ?s

---

$q$-CSP-$B$:
INSTANCE: A CSP instance with $n$ variables and $m$ constraints, where the $n$ variables take values over a set of size $B$, and each constraint involves at most $q$ variables, and is given as a list of acceptable assignments for these variables.
QUESTION: Is there an evaluation that satisfies all constraints ?

---

SAT:
INSTANCE: A CNF formula $\phi$ over the variable set $X$.
QUESTION: Is there a truth assignment of the variables of $X$ that satisfies all clauses of $\phi$ ?

---

UPPER DOMINATING SET:
INSTANCE: A graph $G = (V, E)$, with $|V| = n$ and $|E| = m$.
QUESTION: Determine the maximum size of an upper dominating set in $G$, i.e., of a subset $D \subseteq V$ of vertices of maximum size and which forms a minimal dominating set, that is a set $D$ such that every vertex $v \in V \setminus D$ is adjacent to a vertex of $D$, and which is inclusion-wise minimal, that is if we remove a vertex from $D$ it is not a dominating set anymore.

---

VERTEX COVER:
INSTANCE: An undirected graph $G = (V, E)$, with $|V| = n$ and $|E| = m$, and an integer $k \leq n$.
QUESTION: Is there a vertex cover of size at most $k$ in $G$ ?

# Bibliography


[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[ABCS21] Júlio Araújo, Marin Bougeret, Victor A. Campos, and Ignasi Sau. Parameterized complexity of computing maximum minimal blocking and hitting sets. *CoRR*, abs/2102.03404, 2021.

[ABKS20] Pierre Aboulker, Édouard Bonnet, Eun Jung Kim, and Florian Sikora. Grundy coloring & friends, half-graphs, bicliques. In *STACS*, volume 154 of *LIPIcs*, pages 58:1–58:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[ABLN77] Yousef Alavi, M. Behzad, Linda M. Lesniak-Foster, and E. A. Nordhaus. Total matchings and total coverings of graphs. *Journal of Graph Theory*, 1(2):135–140, 1977.

[ABMS03] Esther M. Arkin, Michael A. Bender, Joseph S. B. Mitchell, and Steven Skiena. The lazy bureaucrat scheduling problem. *Inf. Comput.*, 184(1):129–146, 2003.

[ACP87] Stefan Arnborg, Derek G Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in ak-tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, 1987.

[AHL+18] Hassan AbouEisha, Shahid Hussain, Vadim V. Lozin, Jérôme Monnot, Bernard Ries, and Viktor Zamaraev. Upper domination: Towards a dichotomy through boundary properties. *Algorithmica*, 80(10):2799–2817, 2018.

[ALWZ92] Yousef Alavi, Jiuqiang Liu, Jianfang Wang, and Zhongfu Zhang. On total covers of graphs. *Discret. Math.*, 100(1-3):229–233, 1992.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, 1998.

[BBC+18a] Cristina Bazgan, Ljiljana Brankovic, Katrin Casel, Henning Fernau, Klaus Jansen, Kim-Manuel Klein, Michael Lampis, Mathieu Liedloff, Jérôme Monnot, and Vangelis Th. Paschos. The many facets of upper domination. *Theor. Comput. Sci.*, 717:2–25, 2018.

[BBC+18b] Cristina Bazgan, Ljiljana Brankovic, Katrin Casel, Henning Fernau, Klaus Jansen, Kim-Manuel Klein, Michael Lampis, Mathieu Liedloff, Jérôme Monnot, and Vangelis Th. Paschos. The many facets of upper domination. *Theoretical Computer Science*, 717:2–25, 2018.

[BBCF19] Cristina Bazgan, Ljiljana Brankovic, Katrin Casel, and Henning Fernau. Domination chain: Characterisation, classical complexity, parameterised complexity and approximability. *Discrete Applied Mathematics*, 2019.

[BCL+19] Nikhil Bansal, Parinya Chalermsook, Bundit Laekhanukit, Danupon Nanongkai, and Jesper Nederlof. New tools and connections for exponential-time approximation. *Algorithmica*, 81(10):3993–4009, 2019.

[BCP13] Nicolas Boria, Federico Della Croce, and Vangelis Th. Paschos. On the max min vertex cover problem. In Christos Kaklamanis and Kirk Pruhs, editors, *Approximation and Online Algorithms - 11th International Workshop, WAOA 2013, Sophia Antipolis, France, September 5-6, 2013, Revised Selected Papers*, volume 8447 of *Lecture Notes in Computer Science*, pages 37–48. Springer, 2013.

[BCP15] Nicolas Boria, Federico Della Croce, and Vangelis Th. Paschos. On the max min vertex cover problem. *Discrete Applied Mathematics*, 196:62–71, 2015.

[BEKP13] Edouard Bonnet, Bruno Escoffier, Eun Jung Kim, and Vangelis Th. Paschos. On subexponential and fpt-time inapproximability. In Gregory Z. Gutin and Stefan Szeider, editors, *Parameterized and Exact Computation - 8th International Symposium, IPEC 2013, Sophia Antipolis, France, September 4-6,*

*2013, Revised Selected Papers*, volume 8246 of *Lecture Notes in Computer Science*, pages 54–65. Springer, 2013.

[BEP09] Nicolas Bourgeois, Bruno Escoffier, and Vangelis Th. Paschos. Approximation of min coloring by moderately exponential algorithms. *Inf. Process. Lett.*, 109(16):950–954, 2009.

[BEP11] Nicolas Bourgeois, Bruno Escoffier, and Vangelis Th. Paschos. Approximation of max independent set, min vertex cover and related problems by moderately exponential algorithms. *Discret. Appl. Math.*, 159(17):1954–1970, 2011.

[Ber73] Claude Berge. Graphs and hypergraphs. 1973.

[BKL+20] Rémy Belmonte, Eun Jung Kim, Michael Lampis, Valia Mitsou, and Yota Otachi. Grundy Distinguishes Treewidth from Pathwidth. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *28th Annual European Symposium on Algorithms (ESA 2020)*, volume 173 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:19, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[BL16] Glencora Borradaile and Hung Le. Optimal dynamic program for r-domination problems over tree decompositions. In Jiong Guo and Danny Hermelin, editors, *11th International Symposium on Parameterized and Exact Computation, IPEC 2016, August 24-26, 2016, Aarhus, Denmark*, volume 63 of *LIPIcs*, pages 8:1–8:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[BLP16] Édouard Bonnet, Michael Lampis, and Vangelis Th. Paschos. Time-approximation trade-offs for inapproximable problems. In Nicolas Ollinger and Heribert Vollmer, editors, *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016, February 17-20, 2016, Orléans, France*, volume 47 of *LIPIcs*, pages 22:1–22:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[BM17] Arman Boyaci and Jérôme Monnot. Weighted upper domination number. *Electron. Notes Discret. Math.*, 62:171–176, 2017.

[Bod96] Hans L Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on computing*, 25(6):1305–1317, 1996.

[Bon21] Édouard Bonnet. 4 vs 7 sparse undirected unweighted diameter is seth-hard at time $n^{4/3}$. *CoRR*, abs/2101.02312, 2021.

[CCK+17] Parinya Chalermsook, Marek Cygan, Guy Kortsarz, Bundit Laekhanukit, Pasin Manurangsi, Danupon Nanongkai, and Luca Trevisan. From gap-eth to fpt-inapproximability: Clique, dominating set, and more. *CoRR*, abs/1708.04218, 2017.

[CDL+12] Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. On problems as hard as CNF-SAT. In *Proceedings of the 27th Conference on Computational Complexity, CCC 2012, Porto, Portugal, June 26-29, 2012*, pages 74–84. IEEE Computer Society, 2012.

[Ces01] Marco Cesati. The turing way to the parameterized intractability. 2001.

[CFHJ90] Grant A. Cheston, Gerd Fricke, Stephen T. Hedetniemi, and David Pokrass Jacobs. On the computational complexity of upper fractional domination. *Discret. Appl. Math.*, 27(3):195–207, 1990.

[CFK+15] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer International Publishing, 2015.

[CHKX06] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *J. Comput. Syst. Sci.*, 72(8):1346–1367, 2006.

[CL15] Yijia Chen and Bingkai Lin. The constant inapproximability of the parameterized dominating set problem. *CoRR*, abs/1511.00075, 2015.

[CLN13] Parinya Chalermsook, Bundit Laekhanukit, and Danupon Nanongkai. Independent set, induced matching, and pricing: Connections and tight (subexponential time) approximation

hardnesses. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 370–379, 2013.

[CNP+11] Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 150–159. IEEE Computer Society, 2011.

[Cou90] Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.

[Dem99] Marc Demange. A note on the approximation of a minimum-weight maximal independent set. *Computational Optimization and Applications*, 14(1):157–169, 1999.

[DF99] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999.

[DHG+20] Louis Dublois, Tesshu Hanaka, Mehdi Khosravian Ghadikolaei, Michael Lampis, and Nikolaos Melissinos. (in)approximability of maximum minimal FVS. In Yixin Cao, Siu-Wing Cheng, and Minming Li, editors, *31st International Symposium on Algorithms and Computation, ISAAC 2020, December 14-18, 2020, Hong Kong, China (Virtual Conference)*, volume 181 of *LIPIcs*, pages 3:1–3:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[DLM19] Szymon Dudycz, Mateusz Lewandowski, and Jan Marcinkowski. Tight approximation ratio for minimum maximal matching. In Andrea Lodi and Viswanath Nagarajan, editors, *Integer Programming and Combinatorial Optimization - 20th International Conference, IPCO 2019, Ann Arbor, MI, USA, May 22-24, 2019, Proceedings*, volume 11480 of *Lecture Notes in Computer Science*, pages 181–193. Springer, 2019.

[DLP20] Louis Dublois, Michael Lampis, and Vangelis Th. Paschos. New algorithms for mixed dominating set. In Yixin Cao and Marcin

Pilipczuk, editors, *15th International Symposium on Parameterized and Exact Computation, IPEC 2020, December 14-18, 2020, Hong Kong, China (Virtual Conference)*, volume 180 of *LIPIcs*, pages 9:1–9:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[DLP21a] Louis Dublois, Michael Lampis, and Vangelis Th. Paschos. New algorithms for mixed dominating set. *Discrete Mathematics & Theoretical Computer Science*, 2021.

[DLP21b] Louis Dublois, Michael Lampis, and Vangelis Th. Paschos. Upper dominating set: Tight algorithms for pathwidth and sub-exponential approximation. *CoRR*, 2021.

[DS02] Irit Dinur and Shmuel Safra. The importance of being biased. In John H. Reif, editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 33–42. ACM, 2002.

[EHKK19] Hiroshi Eto, Tesshu Hanaka, Yasuaki Kobayashi, and Yusuke Kobayashi. Parameterized algorithms for maximum cut with connectivity constraints. In Bart M. P. Jansen and Jan Arne Telle, editors, *14th International Symposium on Parameterized and Exact Computation, IPEC 2019, September 11-13, 2019, Munich, Germany*, volume 148 of *LIPIcs*, pages 13:1–13:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[EM77] Paul Erdös and Amram Meir. On total matching numbers and total covering numbers of complementary graphs. *Discret. Math.*, 19(3):229–233, 1977.

[FGLS10] Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, and Saket Saurabh. Intractability of clique-width parameterizations. *SIAM J. Comput.*, 39(5):1941–1956, 2010.

[FLS17] Fabio Furini, Ivana Ljubic, and Markus Sinnl. An effective dynamic programming algorithm for the minimum-cost maximal knapsack packing problem. *European Journal of Operational Research*, 262(2):438–448, 2017.

[GMP13] Laurent Gourvès, Jérôme Monnot, and Aris Pagourtzis. The lazy bureaucrat problem with common arrivals and deadlines: Approximation and mechanism design. In Leszek Gasieniec and

Frank Wolter, editors, *Fundamentals of Computation Theory - 19th International Symposium, FCT 2013, Liverpool, UK, August 19-21, 2013. Proceedings*, volume 8070 of *Lecture Notes in Computer Science*, pages 171–182. Springer, 2013.

[Hås96] Johan Håstad. Clique is hard to approximate within n$^{1\text{-epsilon}}$. In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 627–636. IEEE Computer Society, 1996.

[Hås99] Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Math*, 182:105–142, 1999.

[Hat07] Pooya Hatami. An approximation algorithm for the total covering problem. *Discussiones Mathematicae Graph Theory*, 27(3):553–558, 2007.

[Hat10] Pooya Hatami. An approximation algorithm for the total cover problem. *arXiv preprint arXiv:1008.3216*, 2010.

[HBvdZO19] Tesshu Hanaka, Hans L. Bodlaender, Tom C. van der Zanden, and Hirotaka Ono. On the maximum weight minimal separator. *Theoretical Computer Science*, 796:294 – 308, 2019.

[HGM+20] Ararat Harutyunyan, Mehdi Khosravian Ghadikolaei, Nikolaos Melissinos, Jérôme Monnot, and Aris Pagourtzis. On the complexity of the upper r-tolerant edge cover problem. In Luís Soares Barbosa and Mohammad Ali Abam, editors, *Topics in Theoretical Computer Science - Third IFIP WG 1.8 International Conference, TTCS 2020, Tehran, Iran, July 1-2, 2020, Proceedings*, volume 12281 of *Lecture Notes in Computer Science*, pages 32–47. Springer, 2020.

[HHS98] Teresa W. Haynes, Stephen T. Hedetniemi, and Peter J. Slater. *Fundamentals of domination in graphs*, volume 208 of *Pure and applied mathematics*. Dekker, 1998.

[HKL+18] Tesshu Hanaka, Ioannis Katsikarelis, Michael Lampis, Yota Otachi, and Florian Sikora. Parameterized orientable deletion. In David Eppstein, editor, *16th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2018, June 18-20, 2018, Malmö, Sweden*, volume 101 of *LIPIcs*, pages 24:1–24:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[HKZZ19] Thomas Dueholm Hansen, Haim Kaplan, Or Zamir, and Uri Zwick. Faster $k$-sat algorithms using biased-ppsz. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 578–589. ACM, 2019.

[HP20] Michael A. Henning and Dinabandhu Pradhan. Algorithmic aspects of upper paired-domination in graphs. *Theor. Comput. Sci.*, 804:98–114, 2020.

[HY18] Michael A. Henning and Anders Yeo. On upper transversals in 3-uniform hypergraphs. *Electron. J. Comb.*, 25(4):P4.27, 2018.

[HY19] Michael A. Henning and Anders Yeo. Upper transversals in hypergraphs. *Eur. J. Comb.*, 78:1–12, 2019.

[HY20] Michael A. Henning and Anders Yeo. Bounds on upper transversals in hypergraphs. *J. Comb. Optim.*, 39(1):77–89, 2020.

[IN16] Ken Iwaide and Hiroshi Nagamochi. An improved algorithm for parameterized edge dominating set problem. *J. Graph Algorithms Appl.*, 20(1):23–58, 2016.

[IP01] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

[JJ17] Lars Jaffke and Bart M. P. Jansen. Fine-grained parameterized complexity analysis of graph coloring problems. In Dimitris Fotakis, Aris Pagourtzis, and Vangelis Th. Paschos, editors, *Algorithms and Complexity - 10th International Conference, CIAC 2017, Athens, Greece, May 24-26, 2017, Proceedings*, volume 10236 of *Lecture Notes in Computer Science*, pages 345–356, 2017.

[JJPS17] Pallavi Jain, M. Jayakrishnan, Fahad Panolan, and Abhishek Sahu. Mixed dominating set: A parameterized perspective. In Hans L. Bodlaender and Gerhard J. Woeginger, editors, *Graph-Theoretic Concepts in Computer Science - 43rd International Workshop, WG 2017, Eindhoven, The Netherlands, June 21-23, 2017, Revised Selected Papers*, volume 10520 of *Lecture Notes in Computer Science*, pages 330–343. Springer, 2017.

[JP90] Michael S. Jacobson and Kenneth Peters. Chordal graphs and upper irredundance, upper domination and independence. *Discret. Math.*, 86(1-3):59–69, 1990.

[KGMS20] Kaveh Khoshkhah, Mehdi Khosravian Ghadikolaei, Jérôme Monnot, and Florian Sikora. Weighted upper edge cover: Complexity and approximability. *J. Graph Algorithms Appl.*, 24(2):65–88, 2020.

[Kho02] Subhash Khot. On the power of unique 2-prover 1-round games. In John H. Reif, editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 767–775. ACM, 2002.

[KLP18] Ioannis Katsikarelis, Michael Lampis, and Vangelis Th. Paschos. Structurally parameterized d-scattered set. In Andreas Brandstädt, Ekkehard Köhler, and Klaus Meer, editors, *Graph-Theoretic Concepts in Computer Science - 44th International Workshop, WG 2018, Cottbus, Germany, June 27-29, 2018, Proceedings*, volume 11159 of *Lecture Notes in Computer Science*, pages 292–305. Springer, 2018.

[KLP19a] Ioannis Katsikarelis, Michael Lampis, and Vangelis Th. Paschos. Improved (in-)approximability bounds for d-scattered set. In *WAOA*, volume 11926 of *Lecture Notes in Computer Science*, pages 202–216. Springer, 2019.

[KLP19b] Ioannis Katsikarelis, Michael Lampis, and Vangelis Th. Paschos. Structural parameters, tight bounds, and approximation for (k, r)-center. *Discr. Applied Math.*, 264:90–117, 2019.

[KMS18] Subhash Khot, Dor Minzer, and Muli Safra. Pseudorandom sets in grassmann graph have near-perfect expansion. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 592–601. IEEE Computer Society, 2018.

[KR08] Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2-epsilon. *J. Comput. Syst. Sci.*, 74(3):335–349, 2008.

[Lam18] Michael Lampis. Finer tight bounds for coloring on clique-width. In Ioannis Chatzigiannakis, Christos Kaklamanis,

Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 86:1–86:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[Lam20] Michael Lampis. Finer tight bounds for coloring on clique-width. *SIAM J. Discret. Math.*, 34(3):1538–1558, 2020.

[LC13] James K. Lan and Gerard Jennhwa Chang. On the mixed domination problem in graphs. *TCS*, 476:84–93, 2013.

[Lin21] Bingkai Lin. Constant approximating k-clique is w[1]-hard. *CoRR*, abs/2102.04769, 2021.

[LMS18] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs of bounded treewidth are probably optimal. *ACM Trans. Algorithms*, 14(2):13:1–13:30, 2018.

[Maj93] Aniket Majumdar. Neighborhood hypergraphs: A framework for covering and packing parameters in graphs. 1993.

[Man99] David Manlove. On the algorithmic complexity of twelve covering and independence parameters of graphs. *Discret. Appl. Math.*, 91(1-3):155–175, 1999.

[Mei78] Amram Meir. On total covering and matching of graphs. *J. Comb. Theory, Ser. B*, 24(2):164–168, 1978.

[MM65] John W Moon and Leo Moser. On cliques in graphs. *Israel journal of Mathematics*, 3(1):23–28, 1965.

[MPSS19] Jayakrishnan Madathil, Fahad Panolan, Abhishek Sahu, and Saket Saurabh. On the complexity of mixed dominating set. In René van Bevern and Gregory Kucherov, editors, *Computer Science - Theory and Applications - 14th International Computer Science Symposium in Russia, CSR 2019, Novosibirsk, Russia, July 1-5, 2019, Proceedings*, volume 11532 of *Lecture Notes in Computer Science*, pages 262–274. Springer, 2019.

[MRRS12] Pranabendu Misra, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Parameterized algorithms for even cycle transversal. In Martin Charles Golumbic, Michal Stern, Avivit Levy, and Gila Morgenstern, editors, *Graph-Theoretic*

*Concepts in Computer Science - 38th International Workshop, WG 2012, Jerusalem, Israel, June 26-28, 2012, Revised Selcted Papers*, volume 7551 of *Lecture Notes in Computer Science*, pages 172–183. Springer, 2012.

[MS00] Sounaka Mishra and Kripasindhu Sikdar. On the hardness of approximating some np-optimization problems related to minimum linear ordering problem. In Jan van Leeuwen, Osamu Watanabe, Masami Hagiya, Peter D. Mosses, and Takayasu Ito, editors, *Theoretical Computer Science, Exploring New Frontiers of Theoretical Informatics, International Conference IFIP TCS 2000, Sendai, Japan, August 17-19, 2000, Proceedings*, volume 1872 of *Lecture Notes in Computer Science*, pages 186–199. Springer, 2000.

[MS01] Sounaka Mishra and Kripasindhu Sikdar. On the hardness of approximating some NP-optimization problems related to minimum linear ordering problem. *RAIRO Theor. Informatics Appl.*, 35(3):287–309, 2001.

[PS94] Uri N. Peled and Feng Sun. Total matchings and total coverings of threshold graphs. *Discret. Appl. Math.*, 49(1-3):325–330, 1994.

[PW10] Mihai Patrascu and Ryan Williams. On the possibility of faster SAT algorithms. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1065–1075. SIAM, 2010.

[RHDS18] M. Rajaati, Mohammad Reza Hooshmandasl, Michael J. Dinneen, and Ali Shakiba. On fixed-parameter tractability of the mixed domination problem for graphs with bounded tree-width. *Discret. Math. Theor. Comput. Sci.*, 20(2), 2018.

[RSS+17] M. Rajaati, P. Sharifani, Ali Shakiba, Mohammad Reza Hooshmandasl, and Michael J. Dinneen. An efficient algorithm for mixed domination on generalized series-parallel graphs. *CoRR*, abs/1708.00240, 2017.

[vRBR09] Johan M. M. van Rooij, Hans L. Bodlaender, and Peter Rossmanith. Dynamic programming on tree decompositions using generalised fast subset convolution. In Amos Fiat and Peter

Sanders, editors, *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, volume 5757 of *Lecture Notes in Computer Science*, pages 566–577. Springer, 2009.

[Wil15] Virginia Vassilevska Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In Thore Husfeldt and Iyad A. Kanj, editors, *10th International Symposium on Parameterized and Exact Computation, IPEC 2015, September 16-18, 2015, Patras, Greece*, volume 43 of *LIPIcs*, pages 17–29. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

[XN17] Mingyu Xiao and Hiroshi Nagamochi. Exact algorithms for maximum independent set. *Inf. Comput.*, 255:126–146, 2017.

[XS19] Mingyu Xiao and Zimo Sheng. Improved parameterized algorithms for mixed domination. In *AAIM*, volume 11640 of *LNCS*, pages 304–315. Springer, 2019.

[ZKS11] Yancai Zhao, Liying Kang, and Moo Young Sohn. The algorithmic complexity of mixed domination in graphs. *Theor. Comput. Sci.*, 412(22):2387–2392, 2011.

[Zuc05] David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Electron. Colloquium Comput. Complex.*, (100), 2005.

[ZZ95] Igor E. Zvervich and Vadim E. Zverovich. An induced subgraph characterization of domination perfect graphs. *Journal of Graph Theory*, 20(3):375–395, 1995.

## RÉSUMÉ

Pour résoudre des problèmes NP-difficiles, plusieurs paradigmes ont été développés durant les dernières décennies : l'approximation polynomiale, la résolution exacte, ou encore l'approximation super-polynomiale. Aussi, il a été prouvé que sous certaines hypothèses de complexité, il est impossible d'obtenir certains algorithmes. Dans cette thèse, nous présentons certaines méthodes permettant d'obtenir des algorithmes dans ces différents paradigmes, ainsi que des méthodes pour obtenir des résultats d'impossibilité. Nous illustrons ces méthodes en les mettant en œuvre sur trois problèmes de domination NP-difficiles qui possèdent une structure privée: Min Mixed Dominating Set, où l'on cherche un ensemble minimum d'arêtes et de sommets qui dominent toutes les arêtes et sommets du graphe ; Max Min Feedback Vertex Set, où l'on cherche un feedback vertex set minimal de taille maximum ; et Upper Dominating Set, où l'on cherche un dominating set minimal de taille maximum.

## MOTS CLÉS

Problèmes NP-difficiles, Approximation, Complexité Paramétrée

## ABSTRACT

To tackle NP-hard problems, several paradigms have been developed in the last decades: the polynomial-time approximation, the exact resolution, or the super-polynomial approximation. Moreover, under some complexity assumptions, it has been proven that it is impossible to obtain certain algorithms. In this thesis, we present some methods which allow to obtain algorithms in these paradigms, as well as some methods to obtain intractability results. We illustrate these methods on three NP-hard domination problems which possess some private structure: Min Mixed Dominating Set, where we seek a minimum size set of edges and vertices which dominate all edges and vertices of the graph; Max Min Feedback Vertex Set, where we seek a minimal feedback vertex set of maximum size; and Upper Dominating Set, where we seek a minimal dominating set of maximum size.

## KEYWORDS

NP-hard Problems, Approximation, Parameterized Complexity