The Parameterized Complexity Landscape of Some Graph Problems

विद्या वाचस्पति की
उपाधि की अपेक्षाओं की आंशिक पूर्ति में प्रस्तुत शोध प्रबंध

A thesis submitted in partial fulfillment of the requirements of the degree of Doctor of Philosophy

द्वारा / By

छात्र का नाम / Gaikwad Ajinkya Ramdas गायकवाड अजिंक्य रामदास

पंजीकरण सं. / Registration No.: 20193687

शोध प्रबंध पर्यवेक्षक / Thesis Supervisor:
Prof. Soumen Maity

IISER PUNE

भारतीय विज्ञान शिक्षा एवं अनुसंधान संस्थान पुणे

INDIAN INSTITUTE OF SCIENCE EDUCATION AND RESEARCH PUNE

2024

Certificate

This is to certify that this dissertation entitled The Parameterized Complexity Landscape of Graph Problemstowards the partial fulfilment of the BS-MS dual degree programme at the Indian Institute of Science Education and Research, Pune represents study/work carried out by Gaikwad Ajinkya Ramdas at Indian Institute of Science Education and Research under the supervision of Prof. Soumen Maity, IISER Pune, Professor at IISER Pune, Department of Mathematics , during the academic year 2019-2024.

[illegible]

Prof. Soumen Maity, IISER Pune

Committee:

Prof. Soumen Maity, IISER Pune

Prof. Saket Saurabh, IMSc Chennai

Dr. Vivek Mohan Mallick, IISER Pune

Declaration

I hereby declare that the matter embodied in the report entitled The Parameterized Complexity Landscape of Graph Problems are the results of the work carried out by me at the Department of Mathematics, Indian Institute of Science Education and Research, Pune, under the supervision of Prof. Soumen Maity, IISER Pune and the same has not been submitted elsewhere for any other degree.

[illegible]

Gaikwad Ajinkya Ramdas

Acknowledgments

I would like to express my heartfelt gratitude to everyone who supported me during my journey to complete this thesis.

First and foremost, I extend my appreciation to my advisor, Soumen Maity. His guidance, expertise, and support have been invaluable in shaping my research. I am particularly thankful for his constant availability for discussions, whether they pertained to my work or personal matters, including health challenges. Soumen's understanding and encouragement created a supportive environment that motivated me to continue striving for my goals. I truly appreciate the time and effort he dedicated to my development as a researcher and as an individual.

I would also like to thank Saket Saurabh, my RAC member and collaborator. His encouragement pushed me to take on challenging problems and broaden my perspective. The energy and enthusiasm he brought to our discussions greatly enriched my research experience. I value his insights into the ethics of research and his ability to inspire me to aim for excellence. Working alongside Saket has been a rewarding experience, and I am grateful for the contributions he made to my academic journey. I am thankful to Dr. Vivek Mallick for being in my research advisory committee and for his valuable suggestions.

I would also like to acknowledge my time at IMSc Chennai, where I spent a week working on a problem with Roohani Sharma. This experience initiated our collaboration and led to numerous productive meetings where we exchanged ideas. I am grateful for the fruitful discussions that resulted in some promising outcomes.

This journey would not have been possible without the financial support that enabled my academic pursuits. I am profoundly grateful to the MHRD for granting me the Prime

Minister's Research Fellowship (PMRF) throughout my doctoral studies. I would also like to express my sincere appreciation to the many reviewers whose invaluable feedback greatly improved my work during the review process. Additionally, I am thankful to various local institutions for enhancing my teaching experiences. Lastly, I extend my gratitude to all the research institutes that provided me with opportunities to participate in workshops and conferences across India.

I am also grateful to the administrative staff, especially Mrs. Suvarna Bhardwaj, Mr. Yogesh Kolap, and Mrs. Sayalee Damle. I want to express my heartfelt gratitude to my school teachers and my friends, Amol and Omkar, who sparked my interest in mathematics. Their encouragement and support have been invaluable, and I truly appreciate their friendship throughout my journey.

I would like to express my gratitude to Nishad for his invaluable assistance with my PMRF application. His support made a significant difference, and I am truly thankful for his help. My heartfelt gratitude goes out to my friends, especially Dharm, Pranjal, Ronit, Loveleen, Hitendra, Dhruv, Darshan, Kunal, Ramandeep, Clifford, Tushar, Prachi, Divyashree, Bittu, and Subhash. I truly appreciated the enjoyable moments we shared, both academically and beyond. Their presence and support made my journey all the more memorable.

Finally, words cannot express my gratitude to my parents, especially my brother Akshay. He has been a great source of support, helping me cope with pressure and feel stress-free throughout my academic journey. His ability to lead by example and show me the importance of balance has been invaluable.

Abstract

The study of computational complexity has evolved significantly since the classification of the Boolean satisfiability problem as NP-complete in 1971. With thousands of problems now identified as NP-hard, understanding their tractability remains a central challenge, particularly under the widely held assumption that P $\neq$ NP. This thesis investigates the parameterized complexity of several fundamental graph-theoretic problems, providing new algorithmic insights and complexity classifications.

We analyze problems such as Harmless Set, Defensive Alliances, Offensive Alliances, Locally & Globally Minimal Alliances, and $\mathcal{F}$-Free Edge Deletion, as well as the maximum minimal versions of classical problems like Minimum-Separator and Odd Cycle Transversal. Our research provides a comprehensive parameterized complexity landscape for these problems by establishing their fixed-parameter tractability (FPT) status, designing kernelization algorithms, and proving hardness results.

A key contribution of this thesis is the establishment of W[1]-hardness results for several problems, including Harmless Set, Defensive Alliance, and Offensive Alliance, when parameterized by restrictive structural parameters such as feedback vertex set number, pathwidth, treedepth, and cluster vertex deletion number. Additionally, we develop fixed-parameter tractable (FPT) algorithms for problems such as Harmless Set parameterized by vertex integrity, neighborhood diversity, and twin cover, as well as for Locally Minimal Defensive Alliance when parameterized by solution size. In terms of kernelization complexity, we show that Defensive Alliance does not admit a polynomial kernel when parameterized by vertex cover number, unless $\mathsf{coNP} \subseteq \mathsf{NP/poly}$, and we establish an XP algorithm for this problem parameterized by clique-width. Moreover, we resolve open problems in the literature, such as proving the W[1]-hardness of $\mathcal{T}_{h+1}$-Free Edge Deletion parameterized by treewidth and determining the FPT classification of Maximum Minimal st-Separator parameterized by solution size. Finally, we introduce new kernelization techniques and subexponential algorithms for problems like Locally Minimal Defensive Alliance on special graph classes, including planar graphs.

Beyond their theoretical significance, these results have potential applications in network security, social influence analysis, and computational biology, where efficient structural modifications play a crucial role. By leveraging advanced tools from graph theory, combinatorial optimization, and algorithm design, this thesis provides a refined perspective on the tractability of NP-hard problems within the framework of parameterized complexity.

Contents

Chapter 1

Introduction

In 1971, it was proved that the Boolean satisfiability problem is NP-complete. Later in 1972, Karp provided a list of 21 important NP-complete problems. To this date, thousands of problems have been proved to be NP-hard. Unless the famous conjuncture P$\neq$ NP fails, the NP-hard problems do not admit a polynomial-time algorithm in the size of the input. Since these problems cannot be solved efficiently in the general case, various frameworks have been introduced to tackle them, such as Heuristic Algorithms, Approximation Algorithms, Exact Exponential Algorithms, Randomized Algorithms etc. Each of the above listed frameworks compromise either in the quality of the solution or in the running time (or both). Downey and Fellows introduced the framework of Parameterized Complexity in early 90's for dealing with hard problems. This framework measures the computational complexity of the problem using the input size and an additional measurement which is called the parameter. The most natural candidate for the parameter is the size of the solution we are looking for. Although, in many cases we exploit the structural properties of the input instances such as treewidth, vertex cover number, feedback vertex set number, clique width or solution quality, among others. The overall goal is to identify interesting parameterizations of hard problems where we can design algorithms running in time polynomial in the input size but possibly exponential (or worse) in the small parameter. A problem is called fixed-parameter tractable (FPT) if there exists an algorithm that solves it within a polynomial-time bound in the input size, multiplied by a function of the parameter. Not all parameterized problems are fixed-parameter tractable (FPT) under reasonable complexity-theoretic assumptions. There exists a hierarchy of complexity classes, known as the W-hierarchy, which captures the hardness of parameterized problems. Similar to NP-hardness, which provides evidence that a problem is unlikely to be solvable in polynomial time, showing that a parameterized problem is hard for one of the classes in the W-hierarchy provides evidence that the problem is unlikely to be in FPT.

In this dissertation, we consider parameterized algorithms and complexity of the following graph problems: harmless set, defensive and offensive alliances, locally and globally minimal defensive alliances, $\mathcal{F}$-free edge deletion, $\mathcal{T}_{h+1}$-free edge deletion, and maximum minimal st-separation in graphs. We give below, section-wise, the problems considered.

1.1 Harmless Set

Social networks are used not only to stay in touch with friends and family, but also to spread and receive information on specific products and services. Much of human interaction happens on a local level, wherein our opinions and actions are affected most by our peers, families, and neighbours. The spread of information, propagated through such social interactions, is a well-documented and well-studied topic. Kempe, Kleinberg, and Tardos [89] initiated a model to study the spread of influence through a social network. One of the most well-known problems that appear in this context is Target Set Selection introduced by Chen [26] and defined as follows. We are given a graph, modelling a social network, where each node v has a (fixed) threshold $t(v)$. The node will adopt a new product if at least $t(v)$ of its neighbours adopt it. Their goal is to find a small set S of nodes such that targeting the product to S would lead to the adoption of the product by a large number of nodes in the graph. This problem may occur for example in the context of disease propagation, viral marketing or even faults in distributed computing [39, 112]. For example, to explain disease propagation, consider a network in which each person is in one of two states: infected or uninfected. Suppose a person v becomes infected if at least $t(v)$ of its neighbours are infected. We also assume that a person is never cured once infected. Here a target set is a set of people such that if they are infected at the beginning, then eventually everyone in the network becomes infected. According to Dreyer and Roberts [39] "Such a set of people represents a set of individuals a bioterrorist could infect so as to be sure to infect everyone eventually. Defense against infection would then amount to finding good vaccination or quarantine strategies." Target Set Selection has received considerable attention in a series of papers from classical complexity [39, 22, 27, 114], polynomial time approximability [26, 1], parameterized approximability [11], and parameterized complexity [14, 29, 111]. A natural research direction considering this fact is to look for the complexity of variants or constrained versions of this problem.

Consider the following variant of the marketing problem mentioned above: As before, we are given a graph that represents social interactions, and each node v has a threshold $t(v)$. A person buys the product if it is recommended by at least $t(v)$ people. However, social interactions can spread negative reviews as well, and so, experimenting with a novel product can also harm the company's reputation if the product is not well received by the target set. In such a scenario, it is beneficial for the company to approach the largest group of people, whose possibly negative reviews do not influence other members of society. This motivates the idea of a harmless set in a graph. Indeed, the company can approach a target set for marketing purposes post a successful experiment phase.

The notion of a harmless set was introduced by Bazgan and Chopin [10]. Roughly speaking, a harmless set is a subset S of nodes, such that every node of the graph has fewer than $t(v)$ neighbours in the set S. By definition, a harmless set cannot initiate a cascading behaviour in a network, and in some sense is a converse notion of a target set. The harmlessness of the set can be emphasised by noting that the problem searches for isolated vertices when all thresholds are set to 1. Large harmless sets can be used as a strong first line of defence while mitigating disasters, or to maximise profitable actions without spreading adverse effects. Consider the following instances. Suppose a company has network towers in each area of a city. The residents of each area use a combination of surrounding towers to receive network and would switch to some other company if a certain number of towers stop working. If the company wishes to cut costs by removing some towers without losing any customers, then it must look for a maximum harmless set in the graph with towers as nodes and whose edges connect geographically neighbouring towers. Similarly, a harmless set could also be useful to find trees in a forest for conducting experiments, such that the experiment can be done in natural conditions without affecting any other trees. We now define these notions more formally. A harmless set consists of a set S of vertices with the property that no propagation occurs if any subset of S gets activated.

Definition 1.1.1. [10] Let $G = (V, E)$ be an undirected graph, and $t : V \rightarrow \mathbb{N}$ a threshold function. A subset $S \subseteq V$ is a *harmless set* of G, if for every vertex $v \in V$ we have $|N(v) \cap S| < t(v)$.

Note that in the definition of a harmless set, the threshold condition is imposed on every vertex, including those in the solution set S. According to Bazgan and Chopin [10], "Another perhaps more natural definition could have been a set S such that every vertex $v \notin S$ has fewer than $t(v)$ neighbours in S. This definition raises the following two problems. First, it makes Harmless Set meaningless as a trivial solution would be to take the whole set of vertices of the input graph. Second, there might be some propagation steps inside S if some vertices are activated in it." Here, we consider Harmless Set under structural parameters. We define the problem as follows:

Harmless Set
Input: A graph $G = (V, E)$, a threshold function $t : V \to \mathbb{N}$ where $1 \leq t(v) \leq d(v)$ for every $v \in V$, and an integer k.
Question: Is there a harmless set $S \subseteq V$ of size at least k?

If the threshold function is defined by $t(v) = \lceil \frac{d(v)}{2} \rceil$ for all $v \in V$ then we call the problem the Majority Harmless Set problem. Bazgan and Chopin [10] introduced this problem. We show that the problem is W[1]-hard when parameterized by several restrictive structural parameters, including the feedback vertex set number, pathwidth, treedepth, and cluster vertex deletion number of the input graph. On the positive side, we present fixed-parameter tractable (FPT) algorithms when the problem is parameterized by vertex integrity, neighborhood diversity, or twin cover. Additionally, we establish an upper bound on the complexity by providing an XP algorithm parameterized by clique-width.

1.2 Defensive and Offensive Alliances

In real life, an alliance is a collection of people, groups, or states such that the union is stronger than individual. The alliance can be either to achieve some common purpose, to protect against attack, or to assert collective will against others. This motivates the definitions of defensive alliances in graphs. The properties of alliances in graphs were first studied by Kristiansen, Hedetniemi, and Hedetniemi [95]. They introduced defensive, offensive and powerful alliances. An alliance is global if it is a dominating set. The alliance problems have been studied extensively during last fifteen years [56, 120, 25, 117, 121], and generalizations called r-alliances are also studied [119].

Definition 1.2.1. A non-empty set $S \subseteq V$ is a *defensive alliance* in $G = (V, E)$ if $d_S(v)+1 \geq d_{S^c}(v)$ for all $v \in S$.

We often use the terms *defenders* and *attackers* of an element v of a defensive alliance S. By these we mean the sets $N[v] \cap S$ and $N[v] \setminus S$, respectively. A vertex $v \in S$ is said to be *protected* if the number of defenders of v is greater than or equal to the number of attackers of v, that is, $|N[v] \cap S| = d_S(v) + 1 \geq d_{S^c}(v) = |N[v] \setminus S|$. A set $S \subseteq V$ is a defensive alliance if every vertex in S is protected. In this thesis, we consider Defensive Alliance under structural parameters. We define the problem as follows:

Defensive Alliance
Input: An undirected graph $G = (V, E)$ and an integer $k \geq 1$.
Question: Is there a defensive alliance $S \subseteq V(G)$ such that $|S| \leq k$?

We study the parameterized complexity of Defensive Alliance, where the aim is to find a minimum size defensive alliance. Our main results are the following: (1) Defensive Alliance has been studied extensively during the last twenty years, but the question whether it is FPT when parameterized by feedback vertex set has still remained open. We prove that the problem is W[1]-hard parameterized by a wide range of fairly restrictive structural parameters such as the feedback vertex set number, treewidth, pathwidth, and treedepth of the input graph; (2) the problem parameterized by the vertex cover number of the input graph does not admit a polynomial compression unless coNP $\subseteq$ NP/poly, (3) it does not admit $2^{o(n)}$ algorithm under ETH, and (4) Defensive Alliance on circle graphs is NP-complete.

Definition 1.2.2. A non-empty set $S \subseteq V$ is an *offensive alliance* in G if $d_S(v) \geq d_{S^c}(v)+1$ for all $v \in N(S)$.

Since each vertex in $N(S)$ has more neighbors in S than in S^c, we say that every vertex in $N(S)$ is *vulnerable* to possible attack by vertices in S. Equivalently, since an attack by the vertices in S on the vertices in $V \setminus S$ can result in no worse than a "tie" for S, we say that S can effectively attack $N(S)$.

Definition 1.2.3. A non-empty set $S \subseteq V$ is a *strong offensive alliance* in G if $d_S(v) \geq d_{S^c}(v) + 2$ for all $v \in N(S)$.

We consider Offensive Alliance and Strong Offensive Alliance problems under structural parameters. We define these problems as follows:

Offensive Alliance
Input: An undirected graph $G = (V, E)$ and an integer $r \geq 1$.
Question: Is there an offensive alliance $S \subseteq V(G)$ such that $|S| \leq r$?

Strong Offensive Alliance
Input: An undirected graph $G = (V, E)$ and an integer $r \geq 1$.
Question: Is there a strong offensive alliance $S \subseteq V(G)$ such that $|S| \leq r$?

Alliances have been used to study problems such as classification and clustering problems, understanding communities on the internet, protocols for distribution etc. [122, 118]. The data clustering problem relies on the concept of partitioning the vertices of the graph into multiple strong defensive alliances.

The goal here is to provide new insight into the complexity of Offensive Alliance parameterized by the structure of the input graph. We resolve the problem with most of the structural parameters. We mostly discuss the parameters that deal with sparseness of graph. We show that Offensive Alliance is W[1]-hard parameterized by any of the following parameters: the feedback vertex set number, treewidth, pathwidth, and treedepth of the input graph. Interestingly, our result is significantly stronger since we show that the problem is W[1]-hard parameterized by the size of a vertex deletion set into trees of height at most seven. Next, we turn our attention to parameters the vertex cover number of the input graph and the solution size. It is known that the problem admits FPT algorithms parameterized by each of these parameters individually. As there is no hope to get FPT algorithms with small structural parameters, we need to make the most out of these two parameters by obtaining efficient algorithms and kernels. The FPT algorithm mentioned in [43], has a running time $\mathcal{O}^*(2^{\mathcal{O}(r \log r)})$ where r is the solution size. The first question that arises from here is whether we can get a single exponential algorithm? We answer this question in a negative way by proving that unless ETH fails, Offensive Alliance cannot be solved in time $\mathcal{O}^*(2^{o(r \log r)})$. For the parameter vertex cover number, the FPT algorithm mentioned in [92] has running time $\mathcal{O}^*((2^{\texttt{vc(G)}})^{\mathcal{O}(2^{\texttt{vc(G)}})})$ where $\texttt{vc(G)}$ is the vertex cover number of the input graph. In this case, we improve the running time to $\mathcal{O}^*(\mathtt{vc(G)}^{\mathcal{O}(\mathtt{vc(G)})})$. Finally, we show that it is unlikely to get a polynomial kernel when parameterized by both of these parameters combined.

In search of efficient algorithms, alliance problems have been studied on special graph classes. There are polynomial time algorithms for finding minimum alliances in trees [23, 84]. A polynomial time algorithm for finding minimum defensive alliance in series parallel graph is given in [83]. But still, alliance problems remained unexplored on special classes of intersection graphs such as interval graphs, circle graphs, circular arc graphs, unit disk graphs etc. We show that the problem remains NP-hard even when restricted to bipartite, chordal, split and circle graphs. We also prove that the known algorithms on general graphs and apex graphs are unlikely to improve. This is done by showing that OFFENSIVE ALLIANCE cannot be solved in $2^{o(n)}$ time even when restricted to bipartite graphs and also it cannot be solved in $2^{o(\sqrt{n})}$ time even when restricted to apex graphs, unless ETH fails.

1.3 Locally and Globally Minimal Defensive Alliances

Throughout history, humans have formed communities, guilds, faiths, etc in the hope of coming together with a group of people having similar requirements, visions, and goals. Their reasons to do so usually rest on the fact that any group with common interests often provides added mutual benefits to the union in fields of trade, culture, defense, etc, as compared to the individual. Such activities are commonly seen today in geopolitics, cultures, trades, economics, unions, etc. and are popularly termed as *alliances*. Based on an alliance structure, formation, and goals, many variations of the problem exist in graph theory. A defensive alliance is usually formed to defends its members against non-members; hence, it is natural to ask that each member of the alliance should have more friends within the alliance (including oneself) than outside. These lead to algorithmic problems in the formation of alliances. The main focus here is the study of one such problem from the perspective of parameterized complexity.

A non-empty set D of vertices of a graph is a *defensive alliance* if, for each element of D, the majority of its neighbors are in D. In the DEFENSIVE ALLIANCE problem, given a graph G and a positive integer k, the objective is to decide whether there exists a vertex subset of size at most k, that forms a defensive alliance. During the last 20 years, the DEFENSIVE

ALLIANCE problem, and several of its variants, have been studied extensively from both the combinatorial and computational perspective [95, 113, 12, 21, 120, 117, 119, 121, 43, 16, 49, 52, 84, 56, 21, 51]. As expected, the problems of finding small defensive and offensive alliances are NP-complete.

The focus has chiefly been on finding small alliances, although studying large alliances makes a lot of sense given the original motivation behind these notions, and they were actually also outlined in the very first papers on alliances [95]. To remedy this situation, researchers have considered the notion of minimal defensive alliance problems. We first caution that being a defensive alliance is not a hereditary property, that is, a superset or subset of a defensive alliance is not necessarily a defensive alliance. Shafique [82] called an alliance a *locally minimal alliance* if the set obtained by removing any vertex of the alliance is not an alliance. Bazgan et al. [12] considered another notion of alliance that they called a *globally minimal alliance*, which has the property that no proper subset is an alliance. Bazgan et al. [12] proved that deciding if a graph contains a locally minimal defensive alliance of size at least k is NP-complete, even when restricted to bipartite graphs with an average degree less than 5.6. This naturally leads to the question, what is the tractability of this problem when considered from the algorithmic paradigms meant to cope with NP-hardness, such as parameterized complexity?

Here, we take up such a study and consider the LOCALLY MINIMAL DEFENSIVE ALLIANCE problem from the perspective of parameterized complexity. In this problem, given an undirected graph $G = (V, E)$ and an integer $k \in \mathbb{N}$, the goal is to check whether G has a locally minimal defensive alliance of size at least k, say D. Such a vertex set D is called a *solution*. The parameterized complexity of alliance problems is well-studied with both natural and structural parameters. The problem of finding a defensive alliance of size at most k admits an FPT algorithm when parameterized by solution size, see [50]. We complement this positive result by showing that LOCALLY MINIMAL DEFENSIVE ALLIANCE also admits an FPT algorithm, which uses a similar approach of extreme combinatorics mentioned in [48].

A vertex $v \in D$ is said to be *protected* if $\deg_D(v) + 1 \geq \deg_{D^c}(v)$. Here v has $\deg_D(v) + 1$ defenders and $\deg_{D^c}(v)$ attackers in G. A set $D \subseteq V$ is a defensive alliance if every vertex in D is protected.

Definition 1.3.1. A vertex $v \in D$ is said to be *marginally protected* if it becomes unprotected when any of its neighbors in D is moved from D to $V \setminus D$. A vertex $v \in D$ is said to be *overprotected* if it remains protected even when any of its neighbors is moved from D to $V \setminus D$.

Definition 1.3.2. [12] A defensive alliance D is called a *locally minimal defensive alliance* if for any $v \in D$, $D \setminus \{v\}$ is not a defensive alliance.

It is important to note that if D is a locally minimal defensive alliance, then for every vertex $v \in D$, at least one of its neighbors in D is marginally protected.

Definition 1.3.3. [12] A defensive alliance D is a *globally minimal defensive alliance* or shorter *minimal alliance* if no proper subset is a defensive alliance.

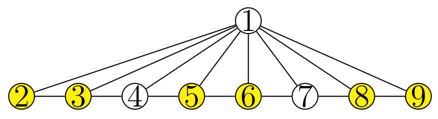


Figure 1.1: The set $D_1 = \{2, 3, 5, 6, 8, 9\}$ is a locally minimal defensive alliance of size 6 and $D_2 = \{1, 2, 4, 6, 8\}$ is a globally minimal defensive alliance of size 5 in G.

It may be noted that every globally minimal defensive alliance is also a locally minimal defensive alliance but the converse is not true. A defensive alliance D is connected if the subgraph induced by D is connected. An alliance D is called a *connected locally minimal alliance* if for any $v \in D$, $D \setminus \{v\}$ is not an alliance and the graph induced by vertices in D is a connected graph. Notice that any globally minimal alliance is always connected. We consider Locally Minimal Defensive Alliance. We define the problem as follows:

Locally Minimal Defensive Alliance
Input: An undirected graph $G = (V, E)$ and an integer k.
Question: Does G have a locally minimal defensive alliance D with $|D| \geq k$?

The parameterized complexity of alliance problems is well-studied with both natural and structural parameters. The problem of finding a defensive alliance of size at most k admits an FPT algorithm when parameterized by solution size, see [50]. We complement this positive result by showing that Locally Minimal Defensive Alliance also admits an FPT algorithm, which uses a similar approach of extreme combinatorics mentioned in [48]. Our main contribution is that Locally Minimal Defensive Alliance is FPT when parameterized by solution size.

In this thesis, we also consider GLOBALLY MINIMAL DEFENSIVE ALLIANCE under various structural parameters. In the literature, a defensive alliance S is called a *global defensive alliance* if S is a dominating set. It is important to note that the globally minimal defensive alliance problem is different from the global defensive alliance problem.

Observation 1.3.1. Let S be a globally minimal defensive alliance of size at least two in G. Then S can never contain a vertex of degree one.

This can be proved by contradiction. Suppose S contains a vertex v of degree one. Note that $\{v\}$ is a proper subset of S and it is a defensive alliance, a contradiction to the fact that S is a globally minimal defensive alliance.

A defensive alliance S is *connected* if the subgraph $G[S]$ induced by S is connected. Notice that any globally minimal defensive alliance is always connected.

Observation 1.3.2. If a non-empty set $S \subseteq V$ is connected and each $v \in S$ is marginally protected, then S is a globally minimal defensive alliance.

Note that although the conditions of this observation are sufficient to assure that S is a globally minimal defensive alliance, they are certainly not necessary. For example, consider the graph G shown in Figure 1.2. Note that $S = \{x, y, u_1, u_2\}$ is a globally minimal defensive alliance, but u_1 and u_2 are not marginally protected.

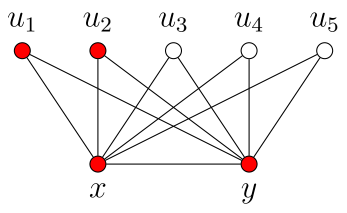


Figure 1.2: $S = \{x, y, u_1, u_2\}$ is a globally minimal defensive alliance in G.

We define the problem as follows:

GLOBALLY MINIMAL DEFENSIVE ALLIANCE
Input: An undirected graph $G = (V, E)$ and an integer $k \geq 2$.
Question: Is there a globally minimal defensive alliance $S \subseteq V(G)$ such that $|S| \geq k$?

We provide new insights into the complexity of Globally Minimal Defensive Alliance parameterized by the structure of the input graph. We show that the problem is FPT parameterized by the neighbourhood diversity of the input graph. This result implies that the problem is also FPT when parameterized by the vertex cover number. We prove that the problem, parameterized by the vertex cover number of the input graph, does not admit a polynomial compression unless coNP $\subseteq$ NP/poly. We show that the problem is W[1]-hard parameterized by a wide range of fairly restrictive structural parameters such as the feedback vertex set number, pathwidth, treewidth and treedepth. We also prove that, given a vertex $r \in V(G)$, deciding if G has a globally minimal defensive alliance of any size containing vertex r is NP-complete.

1.4 $\mathcal{F}$-Free Edge Deletion

Animal diseases pose a risk to public health and cause damage to businesses and the economy at large. Among different reasons for livestock disease, livestock movements constitute major routes for the spread of an infectious livestock disease [76]. For example, the long-range movement of sheep in combination with local transmission resulted in the FMD epidemic in the UK in 2001 [76, 106]. Livestock movements could, therefore, provide insight into the structure of the underlying transmission network and thus allow early detection and more effective management of infectious diseases [90]. To do this, mathematical modelling has been employed widely to describe contact patterns of livestock movements and analyse their potential use for designing disease control strategies [90]. For the purpose of modelling disease spread among farm animals, it is common to consider a transmission network with farms as nodes and livestock movement between farms as edges.

In order to control or limit the spread of a disease on this sort of transmission network, we focus our attention on edge deletion, which might correspond to forbidden trade partners or more reasonably, extra vaccinations or disease surveillance along certain trade routes. Introducing extra control of this kind is costly, so it is important to ensure that this is done as efficiently as possible. Many properties that might be desirable from the point of view of restricting the spread of a disease can be expressed in terms of forbidden subgraphs: delete edges so that each connected component in the resulting graph has at most h vertices, is equivalent to edge-deletion to a graph avoiding all trees on $h + 1$ vertices. In general, given a graph $G = (V, E)$ and a set $\mathcal{F}$ of forbidden subgraphs, we study the $\mathcal{F}$-Free Edge Deletion problem, where the goal is to remove a minimum number of edges such that the resulting graph does not contain any $F \in \mathcal{F}$ as a (not necessarily induced) subgraph. We define the problem as follows:

$\mathcal{F}$-Free Edge Deletion
Input: A graph $G = (V, E)$, a set $\mathcal{F}$ of forbidden subgraphs and a positive integer k.
Question: Does there exist $E' \subseteq E(G)$ with $|E'| = k$ such that $G \setminus E'$ does not contain any $F \in \mathcal{F}$ as a (not necessarily induced) subgraph?

Enright and Meeks (Algorithmica, 2018) gave an algorithm to solve $\mathcal{F}$-Free Edge Deletion whose running time on an n-vertex graph G of treewidth $\mathsf{tw}(G)$ is bounded by $2^{O(|\mathcal{F}|\mathsf{tw}(G)^r)}n$, if every graph in $\mathcal{F}$ has at most r vertices. We complement this result by showing that $\mathcal{F}$-Free Edge Deletion is W[1]-hard when parameterized by $\mathsf{tw}(G) + |\mathcal{F}|$. We also show that $\mathcal{F}$-Free Edge Deletion is W[2]-hard when parameterized by the combined parameters solution size, the feedback vertex set number and pathwidth of the input graph.

1.5 $\mathcal{T}_{h+1}$-Free Edge Deletion

Graph theoretic problems show immense applicability in real-life scenarios such as those involving transportation, flows, relationships and complex systems. The spread of a disease, for instance, can be modeled via such mathematical verbiage. A way to describe this idea is as follows: Livestock are often carriers of various pathogens, and thus their movement, among other reasons, plays a key role in inducing a pandemic amongst humans or livestock [76]. To point at a particular instance, the cause of the 2001 FMD epidemic in the UK can be traced to long-range movements of sheep combined with local transmissions amongst the flocks [76, 106]. Thus, an effective way of mitigating the spread of livestock diseases could be modelled by studying the underlying transmission networks based on the routes of cattle movement. Such models can allow for early detection and better management of disease control strategies [90].

More precisely, we consider a graph with its nodes as livestock farms and edges denoting common routes of livestock movement between farms. Using this graph, we can identify certain edges, or routes, such that connected components of the network obtained by deleting these edges are manageably small in size. Then, these deleted edges will precisely correspond to those trade routes which require more disease surveillance, vaccination stops, movement controls etc, required for disease management. In essence, we have divided our disease control strategies to a few routes and smaller manageable networks. Naturally, for maximum efficiency, one would also like to minimise the number of edges being deleted, and this provides us enough information to chalk out a graph theoretic problem, which we shall describe in the next section. It should be noted that the damaging effect such pandemics have on public health, economies, and businesses, essentially validates the need for such a study.

Many properties that might be desirable from the point of view of restricting the spread of a disease can be expressed in terms of forbidden subgraphs: delete edges so that each connected component in the resulting graph has at most h vertices, is equivalent to edge-deletion to a graph avoiding all trees on $h+1$ vertices. One question of particular relevance to epidemiology would be the complexity of the problem on planar graphs; this would be relevant for considering the spread of a disease based on the geographic location (in situations where a disease is likely to be transmitted between animals in adjacent fields) [44]. Furthermore, in practice animal movement networks can capture more information when considered as directed graphs. The natural generalisation of $\mathcal{T}_{h+1}$-Free Edge Deletion to directed graphs in this contexts follows: given a directed graph and a positive integer k, the goal is to verify whether it is possible to delete at most k edges from a given directed graph so that the maximum number of vertices reachable from any given starting vertex is at most h. The $\mathcal{T}_{h+1}$-Free Edge Deletion problem on planar graph and directed graphs were introduced by Enright and Meeks [44]. Exploiting information on the direction of movements might allow more efficient algorithms for $\mathcal{T}_{h+1}$-Free Edge Deletion; a natural first question would be to consider whether there exists an efficient algorithm to solve this problem on directed acyclic graphs.

We are interested in solving the $\mathcal{T}_{h+1}$-Free Edge Deletion problem. This problem is of particular interest because it can be seen as the problem of removing connections so as to obtain a network where each connected component has at most h vertices, an abstract view of numerous real world problems. Here, we study $\mathcal{T}_{h+1}$-Free Edge Deletion and $\mathcal{T}_{h+1}$-Free Arc Deletion. We define the problems as follows:

$\mathcal{T}_{h+1}$-Free Edge Deletion
Input: An undirected graph $G = (V, E)$, and two positive integers k and h.
Question: Does there exist $E' \subseteq E(G)$ with $|E'| \leq k$ such that each connected component in $G \setminus E'$ has at most h vertices, that is, the graph $G \setminus E'$ does not contain any tree on $h + 1$ vertices as a (not necessarily induced) subgraph?

A natural generalization of this problem to directed graphs in this context would be to consider whether it is possible to delete at most k arcs from a given directed graph so that the maximum number of vertices reachable from any given starting vertex is at most h.

$\mathcal{T}_{h+1}$-Free Arc Deletion
Input: A directed graph $G = (V, E)$, and two positive integers k and h.
Question: Does there exist $E' \subseteq E(G)$ with $|E'| \leq k$ such that the maximum number of vertices reachable from any given starting vertex is at most h in $G \setminus E'$?

Enright and Meeks (Algorithmica, 2018) gave an algorithm to solve $\mathcal{T}_{h+1}$-Free Edge Deletion whose running time on an n-vertex graph G of treewidth $\mathsf{tw}(G)$ is bounded by $O((\mathsf{tw}(G)h)^{2\mathsf{tw}(G)}n)$. We complement this result by showing that $\mathcal{T}_{h+1}$-Free Edge Deletion is W[1]-hard when parameterized by $\mathsf{tw}(G)$ alone. We thereby resolve a conjecture stated by Enright and Meeks (Algorithmica, 2018) concerning the complexity of $\mathcal{T}_{h+1}$-Free Edge Deletion parameterized by the treewidth of the input graph. One question of particular relevance to epidemiology would be the complexity of the problem on planar graphs; this would be relevant for considering the spread of a disease based on the geographic location. We prove that the $\mathcal{T}_{h+1}$-Free Edge Deletion problem is NP-complete even when restricted to planar graphs. We also show that the $\mathcal{T}_{h+1}$-Free Edge Deletion problem is W[2]-hard when parameterized by the solution size on directed acyclic graphs.

1.6 MaxMin Separation Problems

We study fixed-parameter tractability of two fundamental MaxMin separation problems: Maximum Minimum st-Separator (MaxMin st-Sep) and Maximum Minimum Odd Cycle Transversal (MaxMin OCT). In the MaxMin st-Sep problem, the input is an undirected graph G, two distinct vertices s, t of G and a positive integer k. The goal is to determine if there exists a subset of vertices Z, of size *at least* k, such that Z is a *minimal* st-separator in G. That is, the deletion of Z disconnects s and t and the deletion of any proper subset of Z results in a graph where s and t are connected. Similarly, in the MaxMin OCT problem, given G, k, the goal is to determine if there exists a set of vertices Z, of size *at least* k, such that Z intersects all odd length cycles in G and Z is minimal, that is no proper subset of Z intersects all odd length cycles in G.

In contrast to the classical polynomial-time solvable st-Separator problem, where the goal is to find an st-separator of size *at most* k, the MaxMin st-Sep problem is NP-hard [81]. The MaxMin OCT problem can also be shown to be NP-hard by giving a reduction from the MaxMin st-Sep problem.

MaxMin versions of several classical vertex/edge deletion minimization problems have been studied in the literature. The original motivation behind studying such versions is that the size of the solution of the MaxMin versions reflects on the worst-case guarantees of a greedy heuristic. In addition to this, the MaxMin problems have received a lot of attention also because of their deep combinatorial structure which makes them stubborn even towards basic algorithmic ideas. As we will highlight later, neither greedy, nor an exhaustive-search strategy like branching, works for the MaxMin versions of even the "simplest" problems without significant effort. The MaxMin versions are even harder to approximate. For example, the classic Vertex Cover and Feedback Vertex Set problems admit 2-approximation algorithms [5], which are tight under the Unique Games Conjecture [91]. But the MaxMin Vertex Cover admits a $n^{1/2}$-approximation, which is tight unless $P = NP$ [19, 130], and the MaxMin Feedback Vertex Set admits a tight $n^{2/3}$-approximation [41].

We note here that the study of MaxMin versions of classical deletion problems is not the only proposed way of understanding worst-case heursitics guarantees (though in this work we only focus on such versions). Several variations of different problems have been defined whose core is similar. This includes problems like b-Coloring, Grundy Coloring etc, which are analogs for the classic Chromatic Number problem.

We demonstrate that MaxMin versions of two fundamental separation problems: Maximum Minimal st-Separator and Maximum Minimal Odd Cycle Transversal (OCT) problems are fixed-parameter tractable parameterized by k. Our FPT algorithm for Maximum Minimal st-Separator answers the open question by Hanaka, Bodlaender, van der Zanden & Ono [TCS 2019].

1.7 Literature Survey

In this section, we review the existing literature on the specific problems studied in this dissertation.

Harmless Set: Given a graph $G = (V, E)$, a threshold function $t : V \to \mathbb{N}$ and an integer k, we study Harmless Set, where the goal is to find a subset of vertices $S \subseteq V$ of size at least k such that every vertex $v \in V$ has fewer than $t(v)$ neighbours in S. Bazgan and Chopin [10] studied the parameterized complexity of Harmless Set and the approximation of the associated maximization problem. When the parameter is k, they proved that Harmless Set is W[2]-complete in general and W[1]-complete if all thresholds are bounded by a constant. When each threshold is equal to the degree of the vertex, they showed that Harmless Set is fixed-parameter tractable parameterized by k and the maximization version is APX-complete. They gave a polynomial-time approximation scheme for planar graphs. Drange, Muzi and Reidl [38] showed that Harmless Set parameterized by k is fixed-parameter tractable in nowhere dense classes. Bazgan and Chopin [10] gave an algorithm to solve Harmless Set whose running time on an n-vertex graph G of treewidth $\texttt{tw}(G)$ is bounded by $O(k^{O(\texttt{tw}(G))} \cdot n)$ where k is the solution size and we are provided a tree decomposition of width $\texttt{tw}(G)$ as part of the input. However, it remains open whether the problem might belong to FPT when parameterized only by the treewidth $\texttt{tw}(G)$. In this thesis, we resolve this open problem by showing that Harmless Set is indeed W[1]-hard when parameterized by $\texttt{tw}(G)$ alone, even when restricted to bipartite graphs. Drange, Muzi and Reidl [38] proved that Harmless Set is W[1]-hard when parameterized by a modulator to a 2-spider-forest. That is, they independently proved that the problem is W[1]-hard for parameters like treewidth, pathwidth, and even treedepth. They also proved that Harmless Set is fixed-parameter tractable when parameterized by the vertex cover number of the input graph and the problem is solvable in time $O(2^{o(k)} \cdot n)$ on apex-minor-free graphs. Gaikwad and Maity [65] showed that Majority Harmless Set is W[1]-hard when parameterized by the treewidth of the input graph. Bazgan and Chopin [10] introduced the parametric dual problem $(n - k)$-Harmless Set which asks for the existence of a harmless set of size at least $n - k$. Here n is the number of vertices in the input graph, and the parameter is k. They showed that the parametric dual problem $(n - k)$-Harmless Set is fixed-parameter tractable for a large family of threshold functions. Bazgan and Chopin [10] also proved that finding a maximum harmless set can be done in $O(\log(t_{\max}) \cdot n)$ time for trees and finally they showed that given a tree decomposition of width $\mathtt{tw}(G)$ of G, a maximum harmless set can be computed in time $t_{\max}^{O(\mathtt{tw}(G))} \cdot n$ where $t_{\max}$ is the maximum threshold. Gaikwad and Maity [68, 65] provided new insights into the complexity of the Harmless Set problem parameterized by the structure of the input graph. They presented several results, both tractability and intractability, as follows: (i) Harmless Set is FPT when parameterized by any of the following parameters: the neighbourhood diversity, twin cover, and vertex integrity of the input graph. These results are obtained via Integer Linear Programming (ILP). (ii) Majority Harmless Set is W[1]-hard when parameterized by the treewidth of the input graph. (iii) Harmless Set is W[1]-hard when parameterized by the size of a vertex deletion set into trees of height at most 3, even when restricted to bipartite graphs. A similar result is shown independently by Drange, Muzi and Reidl [38]. (iv) Harmless Set is W[1]-hard when parameterized by cluster vertex deletion set. (v) Given a graph G and an irredundant c-expression of G, Harmless Set can be solved in XP-time when parameterized by clique-width.

Defensive and Offensive Alliances: The decision version for several types of alliances have been shown to be NP-complete. For an integer r, a nonempty set $S \subseteq V(G)$ is a *defensive r-alliance* if for each $v \in S$, $|N(v) \cap S| \geq |N(v) \setminus S| + r$. A set is a defensive alliance if it is a defensive (-1)-alliance. A defensive r-alliance S is *global* if S is a dominating set. The defensive r-alliance problem is NP-complete for any r [119]. The defensive alliance problem is NP-complete even when restricted to split, chordal and bipartite graph [84]. There are polynomial time algorithms for finding minimum alliances in trees [23, 84]. A polynomial time algorithm for finding minimum defensive alliance in series parallel graph is presented in [83]. There has also been some work on the parameterized complexity of alliance problems. It is known that Defensive Alliance and Offensive Alliance are fixed-parameter tractable when parameterized by the solution size [43, 50]. Alliance problems have been studied with respect to structural graph parameters. In this paper, we show that the problems for all variants are efficiently solvable for much larger graph classes. Kiyomia and Otachi [92] proved that alliance problems can be solved in polynomial time for graphs of bounded clique-width. They also showed that the problems are fixed-parameter tractable when parameterized by the vertex cover number. Ensico [43] showed that the problems of finding minimum size defensive alliances and global defensive alliances are fixed-parameter tractable when parameterized by the combined parameters treewidth and maximum degree. Treewidth [116, 17] is one of the most extensively studied structural parameters in parameterized complexity. It indicates how close a graph is to being a tree.

It is particularly interesting because there are many hard problems which become tractable on instances of bounded treewidth. It has also been observed that the problem instances for several practical applications exhibit small treewidth [17, 125]. Hence it is very appealing to obtain FPT algorithms for the Defensive Alliance and Offensive Alliance problems using this parameter. Bliem and Woltran [16] proved that defensive alliance problem is W[1]-hard when parameterized by the treewidth of the input graph. Gaikwad and Maity [60, 67] proved that the Offensive Alliance problem is W[1]-hard when parameterized by the treewidth of the input graph. This puts these two problems among the few problems that are FPT when parameterized by solution size but not when parameterized by treewidth (unless FPT=W[1]).

Locally and Globally Minimal Defensive Alliances: Bazgan, Fernau, and Tuza [12] demonstrated that determining whether a graph contains a locally minimal strong defensive alliance of size at least k is NP-complete, even when restricted to bipartite graphs with an average degree of less than 3.6. Similarly, they showed that deciding whether a graph contains a locally minimal defensive alliance of size at least k remains NP-complete under the same restrictions, with an average degree of less than 5.6. Furthermore, the authors proved that determining whether a graph contains a connected locally minimal strong defensive alliance or a connected locally minimal defensive alliance of size at least k is NP-complete, even for bipartite graphs with an average degree of less than $2 + \epsilon$, for any $\epsilon > 0$. Bazgan et al. [12] proved that deciding if a graph contains a globally minimal strong defensive alliance of size at least k is NP-complete, even for cubic graphs. Moreover, deciding if a graph contains a globally minimal defensive alliance of size at least k is NP-complete, even for graphs of degree 3 or 4 [12]. Gaikwad, Maity and Tripathi [72] provided new insights into the complexity of the Locally Minimal Defensive Alliance problem parameterized by the structure of the input graph. They presented several results, both tractability and intractability, as follows: (i) Locally Minimal Defensive Alliance is NP-complete, even when restricted to planar graphs, (ii) it is fixed-parameter tractable (FPT) when parametrized by neighbourhood diversity, (iii) the problem parameterized by treewidth is W[1]-hard and thus not FPT (unless FPT = W[1]), (iv) it can be solved in polynomial time for graphs of bounded treewidth. Gaikwad and Maity [63] gave a polynomial-time algorithm to find a maximum size globally minimal defensive alliance when the input graph happens to be a tree. They also proved that Globally Minimal Defensive Alliance is W[1]-hard when parameterized by the treewidth of the input graph. Gaikwad, Maity and Saurabh [69] showed that the Locally Minimal Defensive Alliance problem is FPT when parameterized by solution size. For graphs that are C_3-free and C_4-free with minimum degree at least 2, they provided a kernel of size $k^{\mathcal{O}(k)}$. When restricted to planar graphs with minimum degree at least 2, they designed an algorithm with running time $k^{\mathcal{O}(\sqrt{k})}n^{\mathcal{O}(1)}$.

$\mathcal{F}$-Free Edge Deletion: If π is a property on graphs or digraphs, the edge-deletion problem can be stated as follows: find the minimum number of edges whose deletion results in a subgraph (or subdigraph) satisfying property π. Several well-studied graph problems can be formulated as edge-deletion problems. Yannakakis [128] showed that the edge-deletion problem is NP-complete for the following properties: (1) without cycles of specified length ℓ, for any fixed $\ell \geq 3$, (2) connected and degree-constrained, (3) outer-planar, (4) transitive digraph, (5) line-invertible, (6) bipartite, (7) transitively orientable. Watanabe, Ae, and Nakamra [126] showed that the edge-deletion problem is NP-complete if π is finitely characterizable by 3-connected graphs. Natanzon, Shamir and Sharan [109] proved the NP-hardness of edge-deletion problems with respect to some well-studied classes of graphs. These include perfect, chordal, chain, comparability, split and asteroidal triple free graphs. This problem has also been studied in generality under paradigms like approximation [57, 105] and parameterized complexity [20, 79]. Cai [20] showed that edge-deletion to a graph class characterisable by a finite set of forbidden induced subgraphs is fixed-parameter tractable when parameterized by k (the number of edges to delete); he gave an algorithm to solve the problem in time $O(r^{2k} \cdot n^{r+1})$, where n is the number of vertices in the input graph and r is the maximum number of vertices in a forbidden induced subgraph. FPT algorithms have been obtained for the problem of determining whether there are k edges whose deletion results in a split graph [75] and to chain, split, threshold, and co-trivially perfect graphs [79]. Enright and Meeks [44] gave an algorithm for the $\mathcal{F}$-Free Edge Deletion problem with running time $2^{O(|\mathcal{F}|\mathsf{tw}(G)^r)}n$ where $\mathsf{tw}(G)$ is the treewidth of the input graph G and r is the maximum number of vertices in any element of $\mathcal{F}$. This is a significant improvement on Cai's algorithm [20] but does not lead to a practical algorithm for addressing real world problems. Gaikwad and Maity [62] complemented this result by showing that $\mathcal{F}$-Free Edge Deletion is W[1]-hard when parameterized by $\mathsf{tw}(G) + |\mathcal{F}|$. They also showed that $\mathcal{F}$-Free Edge Deletion is W[2]-hard when parameterized by the combined parameters: solution size, the feedback vertex set number and pathwidth of the input graph.

$\mathcal{T}_{h+1}$-Free Edge Deletion: The special case of $\mathcal{F}$-Free Edge Deletion problem in which $\mathcal{F}$ is the set of all trees on $h+1$ vertices is of particular interest from the point of view of the control of disease in livestock. Enright and Meeks [44] have proved that $\mathcal{T}_{h+1}$-Free Edge

DELETION is NP-hard for every $h \geq 3$. Enright and Meeks [44] have derived an improved algorithm for this special case, running in time $O((\mathsf{tw}(G)h)^{2\mathsf{tw}(G)}n)$. However, it remained open whether the problem might belong to FPT when parameterized only by the treewidth; they conjectured that treewidth alone is not enough, and that the problem is W[1]-hard with respect to this parameterization. In this thesis, we resolve this conjecture. Gaikwad and Maity [61] gave a polynomial kernel for $\mathcal{T}_{h+1}$-FREE EDGE DELETION parameterized by $k + h$ where k is the solution size. As $\mathcal{T}_{h+1}$-FREE EDGE DELETION is NP-hard for every $h \geq 3$ [44], it is not possible to obtain an FPT-algorithm parameterized by h alone (unless P=NP). However, the parameterized complexity of this problem when parameterized by k alone remains open.

MaxMin Separation Problems: Given the extensive study of the VERTEX COVER problem in parameterized complexity, the parameterized complexity MAXMIN VERTEX COVER has naturally garnered significant attention [129, 19, 130, 3]. More recently, MAXMIN FEEDBACK VERTEX SET problem has also been explored in [41, 98], where several faster FPT algorithms are proposed. The MAXMIN DOMINATING SET problem has been studied in [2, 4, 9, 42], and its edge variant, the MAXMIN EDGE DOMINATING SET, is addressed in [108]. Additionally, MAXMIN and MINMAX formulations have been investigated for a range of other problems, including cut and separation problems [97, 40], knapsack problems [78, 58], matching problems [24], and coloring problems [13]. We elaborate more on the literature around MAXMIN versions of cut and separation problems in the later paragraph.

We remark that for most problems mentioned in the paragraph above, showing that they are FPT parameterized by the solution size is not very difficult (though designing faster FPT algorithms could be much challenging and require deep problem-specific combinatorial insights). The reason is that it is easy to bound the treewidth of the input graph: find a greedy packing of obstructions (edges for MAXMIN VERTEX COVER and cycles for MAXMIN FEEDBACK VERTEX SET); if the packing size is at least k, then the instance is a yes-instance, otherwise there exists a vertex cover (resp. feedback vertex set) of the input graph, of size at most $2k$ (resp. $\mathcal{O}(k \log k)$ from the Erdös-Pósa theorem). In both case, the treewidth of the graph is bounded by a function of k. Since these problems are expressible in Monadic Second Order (MSO) Logic, from Courcelle's theorem a linear-time FPT in k algorithm follows.

The key challenge in the study of cut and separation problems, in contrast to the vertex/edge-deletion problems mentioned in the paragraph above, is that it is not always easy to bound the treewidth of the instances. Having said that the existing work on MaxMin versions of cut and separation problems are still based on treewidth win-win approaches as explained below.

Hanaka et al. [81] studied the parameterized complexity of the MaxMin Separator problem. Here the input is only a connected graph G and a positive integer k, and the goal is to find a minimal vertex set of size at least k whose deletion disconnects the graph. This problem has an easy FPT algorithm parameterized by k based on a win-win approach: if the input graph has large treewidth, then it has a large grid-minor which implies the existence of a large solution; otherwise the treewidth is bounded and since the problem is expressible in MSO, one can solve the problem in linear time on bounded treewidth graphs. Note that this approach completely fails when we are looking for an st-separator (which is the problem we attack in the present work) for fixed vertices s and t. In particular, a large grid minor does not necessarily imply a large st-separator.

In the same work Hanaka et al. designed an explicit FPT algorithm parameterized by treewidth for the MaxMin st-Sep problem and left open the question of determining the parameterized complexity of MaxMin st-Sep parameterized by the solution size k.

Later Duarte et al. [40] studied the edge-deletion variant of the MaxMin st-Sep problem. They termed it the Largest st-Bond problem and showed, amongst other results, that it is FPT parameterized by the solution size k. At the core of this algorithm is another treewidth-based win-win approach, which seems hard to generalize to the vertex-deletion case.

The Maximum Minimal st-Separator problem has been studied in the context of parameterized complexity by Hanaka et al. [81]. Their work focuses on finding a minimal st-separator with maximum weight in a vertex-weighted graph. They showed that the Maximum Weight Minimal st-Separator problem is NP-hard, even on bipartite graphs where all vertex weights are identical. The authors also explored structural parameterization, providing an $\mathcal{O}^*(\mathsf{tw}^{\mathcal{O}(\mathsf{tw})})$-time deterministic algorithm based on tree decompositions, where $\mathcal{O}^*$ notation omits polynomial factors of n. Additionally, they improved the algorithm using a rank-based approach, achieving a running time of $\mathcal{O}(38.2^{\omega})^{\mathsf{tw}}$. Finally, they presented an $\mathcal{O}(9^{\mathsf{tw}} \cdot W^2)$-time randomized algorithm to determine whether there exists a minimal st-separator, where W is its weight and tw is the treewidth of the graph G.

Duarte et al. [40] examined the Largest st-Bond problem, where given an undirected graph G, two vertices $s, t \in V(G)$, and a positive integer k, the task is to find a subset $S \subset V$ such that $s \in S$, $t \notin S$, both $G[S]$ and $G[V \setminus S]$ are connected, and $|\partial(S)| \geq k$, where $\partial(S)$ represents the set of edges with one endpoint in S and the other in $V \setminus S$. They proved that the Largest st-Bond is NP-complete even on planar bipartite graphs, and showed that it does not admit a constant-factor approximation algorithm unless P = NP. The paper also provided an FPT algorithm parameterized by the solution size. The paper presented several FPT algorithms parameterized by different structural parameters such as treewidth, cliquewidth, and twin cover as well.

1.8 An Overview of the Thesis

In Chapter 2, we provide a brief introduction to Parameterized Complexity. We formally define parameterized problems, fixed-parameter tractable algorithms, and kernelization. Additionally, we introduce the concept of fixed-parameter intractability and the W-hierarchy. Finally, we list all the structural graph parameters used in the thesis.

In Chapter 3, we investigate the computational complexity of the Harmless Set problem. This chapter is based on the papers [68, 65]. We show that the problem is W[1]-hard when parameterized by several restrictive structural parameters, including the feedback vertex set number, pathwidth, treedepth, and cluster vertex deletion number of the input graph. On the positive side, we present fixed-parameter tractable (FPT) algorithms when the problem is parameterized by vertex integrity, neighborhood diversity, or twin cover. Additionally, we establish an upper bound on the complexity by providing an XP algorithm when parameterized by clique-width.

In Chapter 4, we analyze the computational complexity of the Defensive Alliance problem. This chapter is based on the paper [61]. We prove that the problem is W[1]-hard when parameterized by several restrictive structural parameters, including the feedback vertex set number, pathwidth, treewidth, treedepth, and clique-width of the input graph, even under the restriction to bipartite graphs. Furthermore, we show that when parameterized by the vertex cover number, the problem does not admit a polynomial compression unless coNP $\subseteq$ NP/poly. We also establish that it cannot be solved in $2^{o(n)}$ time unless the Exponential Time Hypothesis (ETH) fails, and that the problem remains NP-complete on circle graphs. Notably, the constructions used in our hardness proofs extend to problem variants that require finding defensive alliances of exactly a given size.

In Chapter 5, we focus on the Locally Minimal Defensive Alliance problem. This chapter is based on the paper [69]. We demonstrate that the problem is fixed-parameter tractable (FPT) when parameterized by the solution size. For C_3-free and C_4-free graphs with a minimum degree of at least 2, we provide a kernel of size $k^{\mathcal{O}(k)}$. Additionally, when restricted to planar graphs with a minimum degree of at least 2, we design an algorithm with a running time of $k^{\mathcal{O}(\sqrt{k})}n^{\mathcal{O}(1)}$, establishing efficient approaches for these graph classes.

In Chapter 6, we present the main contributions related to the Globally Minimal Defensive Alliance problem. This chapter is based on the papers [63, 64]. We show that the problem is fixed-parameter tractable (FPT) when parameterized by neighborhood diversity. On the other hand, we establish that the problem does not admit a polynomial kernel when parameterized by the vertex cover number. Furthermore, we prove that the problem is W[1]-hard when parameterized by several restrictive structural parameters, including the feedback vertex set number, pathwidth, treewidth, and treedepth of the input graph. Additionally, we demonstrate that deciding whether a given vertex $v \in V(G)$ is part of a globally minimal defensive alliance in G is NP-complete.

In Chapter 7, we analyze the computational complexity of the Offensive Alliance problem. This chapter is based on the papers [60, 67]. We prove that the problem is NP-complete, even when restricted to bipartite, chordal, split, and circle graphs. Furthermore, we establish that the problem is W[1]-hard when parameterized by several restrictive structural parameters, including the feedback vertex set number, treewidth, pathwidth, and treedepth of the input graph, implying that it is not fixed-parameter tractable (FPT) unless FPT = W[1]. This resolves an open question posed by Bernhard Bliem and Stefan Woltran (2018) regarding the complexity of Offensive Alliance parameterized by treewidth. This result is particularly noteworthy because most "subset problems" that are FPT when parameterized by solution size are also FPT for treewidth [33], and Offensive Alliance is computationally easy on trees. On the positive side, we show that the problem can be solved in $\mathcal{O}^*(\text{vc}(G)^{\mathcal{O}(\text{vc}(G))})$ time, where $\text{vc}(G)$ is the vertex cover number of the input graph, and it admits an FPT algorithm when parameterized by the vertex integrity of the input graph. Additionally, we establish a lower bound based on the Exponential Time Hypothesis (ETH), proving that the Offensive Alliance problem cannot be solved in $2^{o(n)}$ time, even when restricted to bipartite graphs, unless ETH fails.

In Chapter 8, we investigate the $\mathcal{F}$-Free Edge Deletion problem, a generalized framework for edge-deletion tasks in graphs. This chapter is based on the papers [65, 62]. The problem asks whether it is possible to remove at most k edges from an input graph G such that the resulting graph avoids all forbidden subgraphs from a given set $\mathcal{F}$. Our results explore both the algorithmic and complexity aspects of this problem. Specifically, we show that the problem is W[1]-hard when parameterized by the treewidth of the input graph combined with the size of $\mathcal{F}$. Additionally, we establish that it is W[2]-hard when parameterized by the combined parameters of solution size, feedback vertex set number, and pathwidth. These hardness results highlight the problem's inherent complexity across a range of structural parameters, advancing the theoretical understanding of edge-deletion problems.

In Chapter 9, we focus on the T_{h+1}-Free Edge Deletion problem, a special case of $\mathcal{F}$-Free Edge Deletion, where $\mathcal{F}$ is the set of all trees with $h+1$ vertices. This chapter is based on the paper [66]. This problem is motivated by practical applications in epidemiology, where minimizing the spread of diseases can be modeled by controlling connected component sizes in networks. We prove that the problem is NP-complete even when restricted to planar graphs. Further, we show that the problem is W[1]-hard when parameterized by treewidth alone, thereby resolving an open conjecture by Enright and Meeks (2018). Additionally, we study its directed variant and prove that the problem is W[2]-hard when parameterized by solution size on directed acyclic graphs. These results underscore the complexity of this edge-deletion task and provide key insights into its parameterized complexity.

In Chapter 10, we investigate the parameterized complexity of the Maximum Minimal st-Separator and Maximum Minimal Odd Cycle Transversal (OCT) problems. This chapter is based on the paper [59]. We establish that both problems are fixed-parameter tractable (FPT) when parameterized by the solution size.

In Chapter 11, we present our conclusions and discuss several open problems.

Chapter 2

Preliminaries

In this chapter, we present some basic definitions in graph theory and introduce terminologies that are used in this thesis. We also cover the basics of parameterized complexity.

2.1 Graph Theory

In this section, we provide some basic definitions from graph theory. For standard notations and definitions in graph theory, we refer to West [127]. Let $G = (V, E)$ denote a finite, simple, and undirected graph. We denote by $V(G)$ and $E(G)$ its vertex and edge set, respectively. For a vertex $v \in V$, we use $N(v) = \{u : (u, v) \in E(G)\}$ to denote the *open neighbourhood* of v in G, and $N[v] = N(v) \cup \{v\}$ to denote the *closed neighbourhood* of v. The degree $d(v)$ of a vertex $v \in V(G)$ is $|N(v)|$. For a subset $S \subseteq V(G)$, we use $N_S(v) = \{u \in S : (u, v) \in E(G)\}$ to denote the (open) neighbourhood of vertex v in S, and $d_S(v)$ to denote the number of neighbours of vertex v in S. We denote by S^c the set of vertices in $G - S$. For $s, t \in V(G)$, an st-path in G refers to a path between s to t in G. For subsets $S, T \subseteq V(G)$, an ST-path is a path between a vertex of S to a vertex of T. A graph H is said to be a subgraph of a graph G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$, denoted by $H \subseteq G$. Given a graph $G = (V, E)$ and a subset of vertices $U \subseteq V$, the induced subgraph $G[U]$ is the subgraph of G that includes all vertices in U and all edges in E that have both endpoints in U. We also define d-scattered set in graphs which will be used later.

Definition 2.1.1 (d-scattered set). *Let $G = (V, E)$ be a graph and let d be a positive integer. A set $S \subseteq V$ is called a d-*scattered set *if for every pair of distinct vertices $u, v \in S$, the distance between u and v in G is greater than d, i.e., $\text{dist}_G(u, v) > d$.*

2.1.1 Graph Types

- A directed graph (or digraph) is a graph where the edges have a direction.
- A graph is said to be *connected* if there is a path between every pair of vertices. If the graph is not connected, it is called a *disconnected graph.* The maximal connected subgraphs are called *connected components* of a graph.
- A graph is called a *tree* if it is connected and acyclic. The height of a rooted tree is the maximum number of vertices on a path from root to a leaf.
- A graph is called *bipartite* if its vertices can be partitioned into two sets V_1 and V_2 such that every edge has one endpoint in V_1 and the other in V_2.
- A graph G is said to be a *chordal graph* if every cycle of G of length at least four has a chord.
- A *split graph* is a special kind of chordal graph where its vertex set $V(G)$ can be partitioned into two subsets V_1 and V_2 such that the graph $G[V_1]$ is a clique and $G[V_2]$ is an independent set.
- A *circle graph* is the intersection graph of a set of chords of a circle. That is, it is an undirected graph whose vertices can be associated with chords of a circle such that two vertices are adjacent if and only if the corresponding chords cross each other.
- A *planar graph* is a graph that can be embedded in the plane, that is, it can be drawn on the plane in such a way that its edges intersect only at their endpoints. In other words, it can be drawn in such a way that no edges cross each other.
- An *apex graph* is a graph that can be made planar by the removal of a single vertex. The deleted vertex is called an *apex* of the graph.

2.1.2 Brekable and Unbreakable Graphs

A pair (X, Y), where $X \cup Y = V(G)$, is called a *separation* if $E(X \setminus Y, Y \setminus X) = \emptyset$. The order of (X, Y) is $|X \cap Y|$. If there exists a separation (X, Y) of order at most k such that $|X \setminus Y| \geq q$ and $|Y \setminus X| \geq q$, then G is (q, k)-*breakable* and the separation (X, Y) is called a *witnessing separation* for the (q, k)-breakability of G. Otherwise, G is (q, k)-*unbreakable.*

2.1.3 Monadic Second Order Logic

The syntax of Monadic Second Order Logic (MSO) formula for graphs includes the logical connectives $\vee, \wedge, \neg, \implies, \iff$, variables for vertices, edges, sets of vertices, and sets of edges, the quantifiers $\forall$ and $\exists$, and five binary relations:

1. $u \in U$, where u is a vertex variable and U is a vertex set variable;
2. $d \in D$, where d is an edge variable and D is an edge set variable;
3. $\mathbf{inc}(d, u)$, where d is an edge variable, u is a vertex variable, and the interpretation is that the edge d is incident to u;
4. $\mathbf{adj}(u, v)$, where u and v are vertex variables, and the interpretation is that u and v are adjacent;
5. equality of variables representing vertices, edges, vertex sets, and edge sets.

Counting Monadic Second Order Logic (CMSO) extends MSO by including atomic sentences testing whether the cardinality of a set is equal to q modulo r, where q and r are integers such that $0 \leq q < r$ and $r \geq 2$. That is, CMSO is MSO with the following atomic sentence:

$$\mathbf{card}_{q,r}(S) = \mathbf{true} \text{ if and only if } |S| \equiv q \mod r,$$

where S is a set. For our applications, we will restrict ourselves to MSO, whereas the next proposition holds for the more general CMSO.

2.2 Introduction to Parameterized Complexity

The time taken by an algorithm is a key measure of its efficiency. For many problems, there are algorithms with time complexities that are polynomial in the size of the input. However, for a large number of problems, no polynomial-time algorithm is known to exist. The algorithms that are known for such problems are typically exponential in the size n of the input. As a result, the time complexity grows much more rapidly with n, making the problem computationally intractable even for relatively small instances. Downey and Fellows [35] introduced the framework of Parameterized Complexity in early 90's for dealing with such hard problems. This framework measures the computational complexity of the problem using the input size and an additional measurement which is called the parameter. We refer to [31, 36] for further details on parameterized complexity.

2.2.1 Fixed Parameter Tractability

A popular method to soften this exponential blow is to look at parameterized problems. Here, every instance is attached with a parameter k, thereby expanding the set of all instances to $\Sigma^* \times \mathbb{N}$. Next, we move to the formal definition of parameterized problems.

Definition 2.2.1. *[31] A parameterized problem Π is a subset of $\Sigma^* \times \mathbb{N}$, where Σ is a (fixed) finite alphabet set. An instance of a parameterized problem is a tuple (x, k), where k is called the parameter.*

A parameterized problem now asks a Yes/No question, often linking x to k, and is characterised by a subset $L \subseteq \Sigma^* \times \mathbb{N}$ of yes instances. Parameterized complexity introduces the notion of fixed-parameter tractability (FPT), which classifies problems based on their complexity with respect to the parameter. A problem is called FPT if there exists an algorithm that solves it within a polynomial-time bound in the input size, multiplied by a function of the parameter. This characterization allows for a fine-grained analysis of complexity. Here we provide a formal definition of a fixed parameter tractable problem.

Definition 2.2.2. *[31] A parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is called fixed parameter tractable if there exists an algorithm $\mathcal{A}$, a computable function $f : \mathbb{N} \to \mathbb{N}$, and a constant c such that, given $(x, k) \in \Sigma^* \times \mathbb{N}$, the algorithm $\mathcal{A}$ correctly decides whether $(x, k) \in L$ in* *time bounded by $f(k) \cdot |(x,k)|^c$. The complexity class containing all fixed parameter tractable problems is called FPT.*

2.3 Standard Techniques used to Design FPT Algorithms

There are several standard techniques for designing FPT algorithms. In this section, we discuss a few of them that are primarily used in this thesis.

2.3.1 Kernelization

Kernelization stands as a cornerstone of parameterized complexity, offering a powerful tool to process complex problem instances to their essential cores. Formally, a kernelization algorithm takes as input an instance of a problem along with a parameter and outputs an equivalent instance whose size is bounded by a polynomial function of the parameter. This output, known as a kernel, retains the essence of the original problem while being significantly smaller in size. Moreover, kernelization serves as a bridge between parameterized complexity theory and practical algorithm design. By providing a systematic framework for preprocessing problem instances, it allows the construction of fixed-parameter tractable (FPT) algorithms that exhibit polynomial-time complexity when the parameter is considered small. Now we provide a formal definition of Kernelization Algorithm.

Definition 2.3.1. *[31] A kernelization algorithm, or simply a kernel, for a parameterized problem Q is an algorithm $\mathcal{A}$ that, given an instance (I,k) of Q, works in polynomial time and return an equivalent instance (I',k') of Q. Moreover, we require that $size_{\mathcal{A}}(k) \leq g(k)$ for some computable function $g : \mathbb{N} \to \mathbb{N}$.*

2.3.2 Bounded search trees

Branching search trees is also known as branching technique. Branching, is one of the simplest and most commonly used techniques in parameterized complexity that uses in general the idea of systematic search. Branching algorithms traverse a search tree representing possible solutions to a given problem instance, with each node corresponding to a partial solution. At each branching point, decisions are made based on the current state of the solution and the values of additional parameters, branching into subproblems that explore different choices. This iterative process continues until either a solution is found or the search space is exhausted, providing valuable information about the problem's complexity. Branching algorithms often exhibit exponential worst-case complexity, especially when exploring all possible branches of the search tree. This exponential growth arises from the combinatorial explosion of choices at each branching point, making exhaustive exploration impractical for large instances. The complexity of branching algorithms depends upon both the branching factor and the depth of the search tree. A high branching factor or depth can significantly increase the number of nodes explored, leading to longer execution times and increased memory requirements.

2.3.3 Parameterized Reduction

Lets recall that the idea of polynomial time reduction in classical complexity allows us to prove NP-hardness results. In the polynomial time reduction algorithm, we map an instance of problem A to an equivalent instance of problem B in polynomial time. If there is a polynomial time reduction from problem A to a problem B then it implies that if a problem B can be solved in polynomial time then the problem A also admits a polynomial time algorithm. The underlying idea in the parameterized reduction is similar, except here we have to maintain the size of a parameter in the reduced problem. Formally, a parameterized reduction maps instances of one parameterized problem to instances of another, such that the transformed instance preserves the parameterized complexity of the original. This preservation ensures that the inherent difficulty captured by additional parameters remains intact, which helps us to analyze the complexity landscape of problems through a parameterized lens. Here, we provide a formal definition of parameterized reduction.

Definition 2.3.2. *[31] Let $A, B \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems. A parameterized reduction from problem A to B is an algorithm that, given an instance (x, k) of A, outputs an instance (x', k') of B such that*

1. *(x, k) is a yes instance of A if and only if (x', k') is a yes instance of B,*

2. *$k' \leq g(k)$ for some computable function g, and*

3. the running time is $f(k).|x|^{\mathcal{O}(1)}$ for some computable function f.

The parameterized reduction provides a way to prove that a problem is FPT in the following way.

Theorem 2.3.1. *[31] If there is a parameterized reduction from problem A to B and B is FPT, then A is FPT as well.*

2.3.4 The ILP Technique

This technique is based on a computational problem called integer linear programming (ILP). A special and important case of the parameterized reduction technique is to map a problem to an integer linear programming (ILP) problem. Integer linear programming (ILP) is defined as follows:

p-ILP
Input: A matrix $A \in Z^{m\times p}$, and vectors $b \in Z^p$ and $c \in Z^p$.
Question: Find a vector $x \in Z^p$ that satisfies the m inequalities, that is, $A \cdot x \geq b$.
Parameter: p, the number of variables.

Lenstra [99] showed that p-ILP is FPT with running time doubly exponential in p, where p is the number of variables. Later, Kannan [87] proved an algorithm for p-ILP running in time $p^{O(p)}$. In our algorithms, we need the optimization version of p-ILP rather than the feasibility version. We state the minimization version of p-ILP as presented by Fellows et al. [47]. The problem and the result are formally stated as follows.

p-Opt-ILP
Input: A matrix $A \in Z^{m\times p}$, and vectors $b \in Z^p$ and $c \in Z^p$.
Question: Find a vector $x \in Z^p$ that minimizes the objective function $c^T \cdot x$ and satisfies the m inequalities, that is, $A \cdot x \geq b$.
Parameter: p, the number of variables.

Lemma 2.3.2 (Fellows et al. [47]). p-Opt-ILP *can be solved using $O(p^{2.5p+o(p)} \cdot L \cdot \log(MN))$ arithmetic operations and space polynomial in L. Here L is the number of bits in the input,*

N is the maximum absolute value any variable can take, and M is an upper bound on the absolute value of the minimum taken by the objective function.

Given any problem P with parameter k, we must try to reformulate it as an ILP problem with at most $f(k)$ variables for some computable function f in order to apply the ILP technique. This is the most crucial requirement for this technique. If then, substituting L, M and N gives a running time polynomial in n, we shall obtain an FPT algorithm for the problem P.

2.4 Standard Techniques used to Design Lower Bounds

In this section, we will focus on the tools used in this thesis to establish lower bounds on the complexity of certain computational problems.

2.4.1 The W-hierarchy

As discussed in the previous section, the first technique is parameterized reduction. We know that given a parameterized reduction from problem A to B implies that if the problem B is FPT then so is problem A. Similar to the theory of NP-completeness, we can also say that problem A is at least as hard as problem B. Note that, if we assume that the problem A is not FPT then it implies that the problem B is not FPT as well. If we accept as a working hypothesis that Clique is not fixed-parameter tractable, then given a reduction from Clique to other problems suggests that the other problem is not fixed-parameter tractable either. Lets look at some simple examples. There is a simple parameterized reduction from Clique to Independent Set. This implies that Clique is at least as hard as Independent Set. Let us recall that given any two NP-complete problems, there is always exists a polynomial time reduction from one to the other. This kind of phenomenon seem to fail in the realm of parameterized complexity. We know that there is parameterized reduction from Clique to Dominating Set but there is no known parameterized reduction from Dominating Set to Clique. This suggests that there are different levels in the parameterized complexity of hard problems. More details of W-hierarchy can be in Chapter 13 in [31]. As far as this thesis is concerned, we are mainly interested in whether given problem and a parameter, does there exists a FPT algorithm or not. If there is no FPT algorithm then we do not pay much attention to the different levels of hardness in the W-hierarchy. The following theorem are used to provide a number of lower bounds on the complexity of some problems.

Theorem 2.4.1. *[35] INDEPENDENT SET is W[1]-hard.*

Theorem 2.4.2. *[31] DOMINATING SET, SET COVER, and HITTING SET are W[2]-hard.*

It is important to note that the above theorem are true under the assumption that FPT$\neq$W[1] and not P$\neq$NP.

2.4.2 Lower Bounds based on Exponential Time Hypothesis

The *Exponential Time Hypothesis (ETH)* is a conjecture stating that n-variable 3-SAT cannot be solved in time $2^{o(n)}$. Standard techniques for designing FPT algorithms—such as *branching*, *kernelization*, *color coding*, and *iterative compression*—typically yield algorithms with running times of the form $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$, $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$, or even $2^{\text{poly}(k)} \cdot n^{\mathcal{O}(1)}$. These forms suggest that FPT problems admit an internal *hierarchy* based on the parameter dependence of their running times. The ETH conjecture serves as a useful tool in establishing this hierarchy. It allows us to show that, for many problems, the known FPT algorithms are essentially *optimal* under ETH. In the context of parameterized complexity, ETH is frequently used to prove *lower bounds* of the form that a problem cannot be solved in time $2^{o(k)} \cdot n^{\mathcal{O}(1)}$, $k^{o(k)} \cdot n^{\mathcal{O}(1)}$, or even $f(k) \cdot n^{o(k)}$ for any computable function f.

2.4.3 Lower bounds on Kernelization

As mentioned in the previous section, kernelization is one of the important tool to construct FPT algorithms. There are number of classical problems for which polynomial kernels have been constructed. For example, VERTEX COVER admits a kernel of size $2k$, and FEEDBACK VERTEX SET admits a kernel of size $\mathcal{O}(k^2)$. The list is quite long. One might ask whether, given a parameterized problem, it is always possible to construct a kernel of polynomial size. Unfortunately, the answer is no. There are many ways to prove the non-existence of a polynomial kernel. Here, we discuss one such method called Polynomial Parameter

Transformation(PPT) which is used multiple times in this thesis. This technique allows us to transfer the hardness from one problem to another. To do this, we provide the following definition.

Definition 2.4.1 ([53]). *Let $P, Q \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems. An algorithm $\mathcal{A}$ is called a polynomial parameter transformation from P to Q if, given an instance (x, k) of problem P, $\mathcal{A}$ works in polynomial time and outputs an equivalent instance (x', k') of problem Q, that is, $(x, k) \in P$ if and only if $(x', k') \in Q$, such that $k' \leq p(k)$ for some polynomial $p(.)$.*

The above definition allows us to state the following.

Theorem 2.4.3 ([53]). *Let $P, Q \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems and assume there exists a polynomial parameter transformation from P to Q. Then, if P does not admit a polynomial compression, neither does Q.*

We refer to [53] for further details on kernelization.

2.5 Structural Graph Parameters

In this section, we recall the definitions of the structural graph parameters discussed in this thesis and provide information on how these parameters are related to one another.

Definition 2.5.1. A set $S \subseteq V(G)$ is a *vertex cover* of $G = (V, E)$ if each edge in E has at least one endpoint in S. The *size* of a smallest vertex cover of G is the *vertex cover number* of G.

Next, we define Twin Cover, Neighbourhood Diversity, and Vertex Integrity of a graph.

According to Ganian [73], "The twin-cover is a direct generalization of vertex cover towards richer graph classes, including dense graph classes (which have unbounded treewidth)." Here we relax the definition of vertex cover so that not all edges need to be covered.

Definition 2.5.2. *[73] An edge $(u, v) \in E(G)$ is a* twin edge *of G if $N[u] = N[v]$.*

Definition 2.5.3. *[73] A set $X \subseteq V(G)$ is a* twin-cover *of G if every edge in G is either twin edge or incident to a vertex in X. We then say that G has* twin-cover number *k if k is the minimum possible size of a twin-cover of G.*

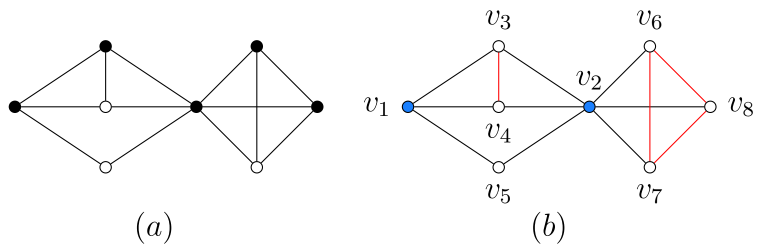


Figure 2.1: (a) A minimum vertex cover (size 5–*depicted in black*) (b) A minimum twin-cover (size 2–*depicted in blue*). The twin edges are shown in red.

Two different vertices u, v are called *true twins* if $N[u] = N[v]$. Likewise, u, v are called *false twins* if $N(u) = N(v)$. In general, u, v are called *twins* if they are either true twins or false twins. If they are twins, we say that they have the same *neighbourhood type*.

Definition 2.5.4. *[96] A graph $G = (V, E)$ has* neighbourhood diversity *at most d, if there exists a partition of V into at most d sets (we call these sets type classes) such that all the vertices in each set have the same neighbourhood type.*

If the neighbourhood diversity of a graph is bounded by an integer d, then there exists a partition $\{C_1, C_2, \ldots, C_d\}$ of $V(G)$ into d type classes. We would like to point out that it is possible to compute the neighbourhood diversity of a graph in linear time using fast modular decomposition algorithms [124]. Notice that each type class could either be a clique or an independent set by definition. For algorithmic purposes it is often useful to consider the *type graph H* of G, where each vertex of H is a type class in G, and two vertices C_i and C_j are adjacent if and only if every vertex of C_i is adjacent to every vertex of C_j in G. It is not difficult to see that there will be either no edges between C_i and C_j or every vertex of C_i is adjacent to every vertex of C_j. The key property of graphs of bounded neighbourhood diversity is that their type graphs have bounded size. For example, a graph G with neighbourhood diversity four and its corresponding type graph H are illustrated in Figure 2.2.

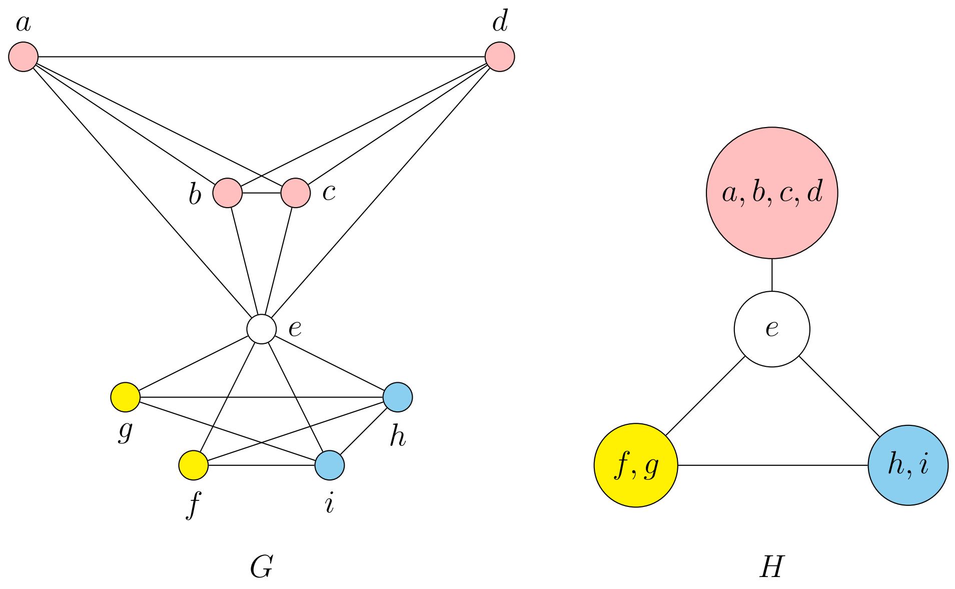


Figure 2.2: A graph G with neighbourhood diversity 4 and its corresponding type graph H.

Next, we move to vertex integrity. This vulnerability measure was introduced by Barefoot et al. [7] in 1987. For an overview of structural results on vertex integrity, we refer the reader to a survey on vertex integrity by Bagga et al. [6].

Definition 2.5.5. *[7] The* vertex integrity *of a graph G, denoted $\texttt{vi}(G)$, is the minimum integer k satisfying that there is $X \subseteq V(G)$ such that $|X| + |V(C)| \leq k$ for each component C of $G - X$. We call such S a k-$\texttt{vi}$-set of G.*

Gima et al. [77] described an equivalence relation among components. For a vertex set X of G, we define an equivalence relation $\sim_{G,X}$ among components of $G - X$ by setting $C_1 \sim_{G,X} C_2$ if and only if there is an isomorphism g from $G[X \cup V(C_1)]$ to $G[X \cup V(C_2)]$ that fixes X; that is, $g|_X$ is the identity function. When $C_1 \sim_{G,X} C_2$, we say that C_1 and C_2 have the same (G, X)-*type* (or just the same *type* if G and X are clear from the context). See Figure 2.3.

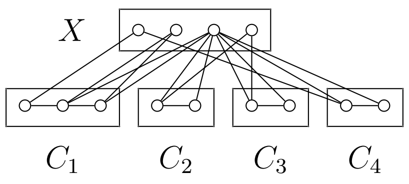


Figure 2.3: The components C_2 and C_3 of $G - X$ have the same $(G, X)-$type.

This equivalence relation induces a set of equivalence classes $\mathcal{C}_1, \mathcal{C}_2, \ldots$. We can choose a representative of each equivalence class.

Next, we define cluster vertex deletion number of a graph. One can observe that this is a generalization of twin cover in dense graphs.

Definition 2.5.6. *[34] The* cluster vertex deletion number *of a graph is the minimum number of its vertices whose deletion results in a disjoint union of complete graphs.*

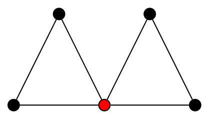


Figure 2.4: The red vertex form a cluster vertex deletion set of the given graph.

An example of a cluster vertex deletion set is given in Figure 2.4.

Next, we will define feedback vertex set and treedepth of a graph.

Definition 2.5.7. A *feedback vertex set* of a graph G is a set of vertices whose removal turns G into a forest. The minimum size of a feedback vertex set in G is the *feedback vertex set number* of G, denoted by $\texttt{fvs}(G)$.

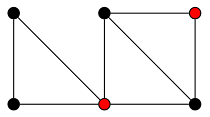


Figure 2.5: The red vertices form a feedaback vertex set of the given graph.

An example of a feedback vertex set is given in Figure 2.5.

A rooted forest is a disjoint union of rooted trees. Given a rooted forest F, its *transitive closure* is a graph H in which $V(H)$ contains all the nodes of the rooted forest, and $E(H)$ contain an edge between two vertices whenever those two vertices form an ancestor-descendant pair in the forest F.

Definition 2.5.8. The *treedepth* of a graph G is the minimum height of a rooted forest F whose transitive closure contains the graph G as a subgraph. It is denoted by $\texttt{td}(G)$.

Example is given in figure 2.6.

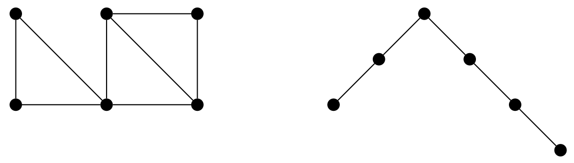


Figure 2.6: The input graph is on the left and a corresponding rooted forest is on the right.

Next, we review the concept of a tree decomposition introduced by Robertson and Seymour [116]. Treewidth is a measure of how "tree-like" the graph is.

Definition 2.5.9 (Robertson and Seymour [116]). *A tree decomposition of a graph $G = (V, E)$ is a tree T together with a collection of subsets X_t (called* bags*) of V labeled by the nodes t of T such that $\bigcup_{t \in T} X_t = V$ and (1) and (2) below hold:*

1. *For every edge $(u, v) \in E(G)$, there is some t such that $\{u, v\} \subseteq X_t$.*
2. *(Interpolation Property) If t is a node on the unique path in T from t_1 to t_2, then $X_{t_1} \cap X_{t_2} \subseteq X_t$.*

Definition 2.5.10 (Robertson and Seymour [116]). *The width of a tree decomposition is the maximum value of $|X_t| - 1$ taken over all the nodes t of the tree T of the decomposition. The* treewidth $\texttt{tw}(G)$ *of a graph G is the minimum width among all possible tree decompositions of G.*

Example 1. *Figure 2.7 gives an example of a tree decomposition of width 2.*

Definition 2.5.11. If the tree T of a tree decomposition is a path, then we say that the tree decomposition is a *path decomposition*. The *pathwidth* $\texttt{pw}(G)$ of a graph G is the minimum width among all possible path decompositions of G.

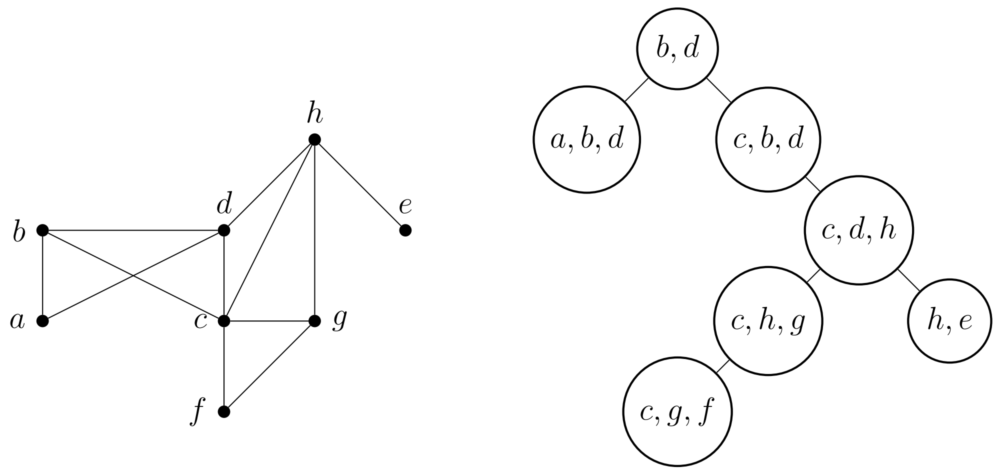


Figure 2.7: Example of a tree decomposition of width 2

Finally, we define cliquewidth of a graph.

Definition 2.5.12. *The clique-width of a graph G, denoted by $\texttt{cw}(G)$, is the minimum number of labels needed to construct G using the following four operations:*

1. *Create a new graph with a single vertex v with label i (written $i(v)$).*
2. *Take the disjoint union of two labelled graphs G_1 and G_2 (written $G_1 \oplus G_2$).*
3. *Add an edge between every vertex with label i and every vertex with label j, $i \neq j$ (written $\eta_{i,j}$).*
4. *Relabel every vertex with label i to have label j (written $\rho_{i \to j}$).*

We say that a construction of a graph G with the four operations is a c-expression if it uses at most c labels. Thus the clique-width of G is the minimum c for which G has a c-expression. A c-expression is a rooted binary tree T such that

1. *each leaf has label i for some $i \in \{1, \ldots, c\}$,*
2. *each non-leaf node with two children has label $\oplus$, and*
3. *each non-leaf node with only one child has label $\rho_{i \to j}$ or $\eta_{i,j}$ $(i, j \in \{1, \ldots, c\}, i \neq j)$.*

Example 2. Consider the graph P_n, which is simply a path on n vertices. Note that $\texttt{cw}(P_1) = 1$ and $\texttt{cw}(P_2) = \texttt{cw}(P_3) = 2$. Now consider a path on four vertices v_1, v_2, v_3, v_4, in that order. Then this path can be constructed using the four operations (using only three labels) as follows:

$$\eta_{3,2}(3(v_4) \oplus \rho_{3\to2}(\rho_{2\to1}(\eta_{3,2}(3(v_3) \oplus \eta_{2,1}(2(v_2) \oplus 1(v_1)))))).$$

Figure 2.8 shows a 3-expression of P_4. This construction can readily be generalized to longer

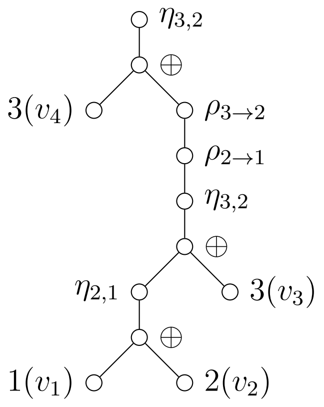


Figure 2.8: A 3-expression of P_4.

paths for $n \geq 5$. It's easy to see that $\texttt{cw}(P_n) \leq 3$ for all $n \geq 4$.

A c-expression represents the graph represented by its root. A c-expression of an n-vertex graph G has $O(n)$ vertices. A c-expression of a graph is *irredundant* if for each edge $\{u, v\}$, there is exactly one node $\eta_{i,j}$ that adds the edge between u and v. It is known that a c-expression of a graph can be transformed into an irredundant one with $O(n)$ nodes in linear time [30]. From here on, we assume we are given an irredundant c-expression.

Computing the clique-width and a corresponding c-expression of a graph is NP-hard. For $c \leq 3$, we can compute a c-expression of a graph of clique-width at most c in $O(n^2m)$ time, where n and m are the number of vertices and edges, respectively. For fixed $c \geq 4$, it is not known whether one can compute the clique-width and a corresponding c-expression of a graph in polynomial time. On the other hand, it is known that for any fixed c, one can compute a $(2^{c+1}-1)$-expression of a graph of clique-width c in $O(n^3)$ time. For more details, see [86].

Figure 2.9 shows how these parameters are related to each other.

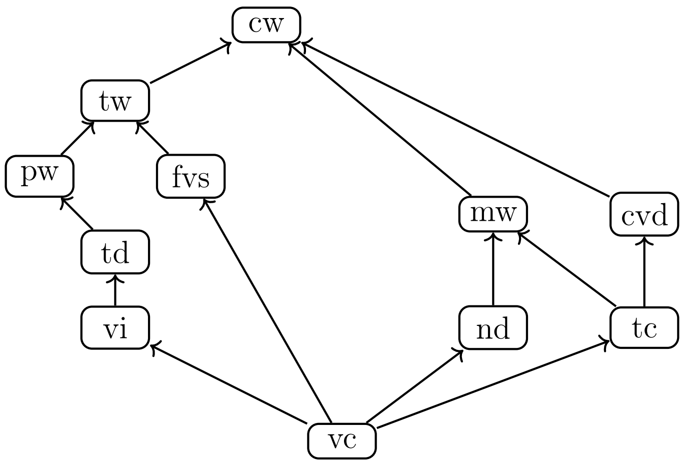


Figure 2.9: Relationship between vertex cover [vc] (see Definition 2.5.1), neighbourhood diversity [nd] (see Definition 2.5.4), twin cover [tc] (see Definition 2.5.3), modular width [mw] (see [107]), cluster vertex deletion number [cvd] (see Definition 2.5.6), feedback vertex set [fvs] (see Definition 2.5.7), pathwidth [pw] (see Definition 2.5.11), treewidth [tw] (see Definition 2.5.10) and clique width [cw] (see Definition 2.5.12). Note that $A \to B$ means that there exists a function f such that for all graphs, $f(A(G)) \geq B(G)$.

Chapter 3

Harmless Set

In this chapter, we study the Harmless Set problem from a parameterized complexity perspective. Given a graph $G = (V, E)$, a threshold function $t : V \to \mathbb{N}$ and an integer k, we study Harmless Set, where the goal is to find a subset of vertices $S \subseteq V$ of size at least k such that every vertex $v \in V$ has fewer than $t(v)$ neighbours in S. On the positive side, we obtain fixed-parameter algorithms for the problem when parameterized by the neighbourhood diversity, the twin-cover number and the vertex integrity of the input graph. We complement two of these results from the negative side. On dense graphs, we show that the problem is W[1]-hard parameterized by cluster vertex deletion number – a natural generalization of the twin-cover number. We show that the problem is W[1]-hard parameterized by a wide range of fairly restrictive structural parameters such as the feedback vertex set number, pathwidth, and treedepth – a natural generalization of the vertex integrity. We thereby resolve one open question stated by Bazgan and Chopin (*Discrete Optimization*, 2014) concerning the complexity of Harmless Set parameterized by the treewidth of the input graph. We also show that Harmless Set for a special case where each vertex has the threshold set to half of its degree (the so-called Majority Harmless Set problem) is W[1]-hard when parameterized by the treewidth of the input graph. Given a graph G and an irredundant c-expression of G, we prove that Harmless Set can be solved in XP-time when parameterized by clique-width. This chapter is based on the papers [65, 68].

3.1 FPT via Integer Linear Programming

In this section, we study the parameterized complexity of Harmless Set with respect to various structural parameters. Drange, Muzi and Reidl [38] proved that Harmless Set is fixed-parameter tractable when parameterized by the vertex cover number of the input graph. We present FPT algorithms for Harmless Set parameterized by the neighbourhood diversity, twin-cover number, and vertex integrity of the input graph. Studying the parameterized complexity of Harmless Set with respect to these parameters improves the understanding of the boundary between tractable and intractable parameterizations. Our algorithms use an integer linear programming (ILP) subroutine. ILP is a well known framework for formulating problems and a powerful tool for the development of fixed-parameter algorithms for optimization problems.

Suppose A and B are two parameterized problems. It is known that if A fpt-reduces to B and B is fixed-parameter tractable, then A is fixed-parameter tractable as well. In this section, we shall let B be the classical Integer Linear Programming (ILP) problem, and show how FPT algorithms can be designed for Harmless Set, via the ILP method. Given any problem P with parameter k, we must try to reformulate it as an ILP problem with at most $f(k)$ variables for some computable function f in order to apply the ILP technique. This is the most crucial requirement for this technique.

3.1.1 Harmless Set **Parameterized by Neighbourhood Diversity**

Let us first recall the definition of neighbourhood diversity. Two different vertices u, v are called *true twins* if $N[u] = N[v]$. Likewise, u, v are called *false twins* if $N(u) = N(v)$. In general, u, v are called *twins* if they are either true twins or false twins. If they are twins, we say that they have the same *neighbourhood type.*

Definition 3.1.1 (Lampis [96]). *has* neighbourhood diversity *at most d, if there exists a partition of V into at most d sets (we call these sets type classes) such that all the vertices in each set have the same neighbourhood type.*

The following result explains why the vertices with low thresholds are included in the solution.

Lemma 3.1.1. Let C_i be a type class and $(v_1, v_2, \ldots, v_{n_i})$ be an increasing ordering of $V(C_i)$ according to threshold values, that is, $t(v_1) \leq t(v_2) \leq \cdots \leq t(v_{n_i})$. Let S be a maximum size harmless set in G and $S_i = C_i \cap S$. Then $S' = (S \setminus S_i) \cup \{v_1, v_2, \ldots, v_{|S_i|}\}$ is also a maximum size harmless set in G.

Proof. Clearly, $|S| = |S'|$. To show S' is a harmless set, it is enough to show that each vertex v in C_i has fewer than $t(v)$ neighbours in S'. Recall that C_i is either an independent or a clique type class and every vertex in C_i has the same neighbourhood in G. We consider two cases:

Case 1: Suppose that $v \in \{v_1, \ldots, v_{|S_i|}\}$, that is, v has a smaller threshold value. If $v \in S$, then it will remain in S'. Therefore $d_{S'}(v) = d_S(v) < t(v)$. If $v \notin S$, then to obtain S' from S we remove a vertex $v' \in S \setminus \{v_1, \ldots, v_{|S_i|}\}$ having a larger threshold value $t(v') \geq t(v)$ from S and include v in S. Note that $d_{S'}(v) = d_S(v) < t(v)$ if C_i is an independent type class whereas $d_{S'}(v) = d_S(v) - 1 < t(v)$ if C_i is a clique type class. Therefore v satisfies the threshold condition.

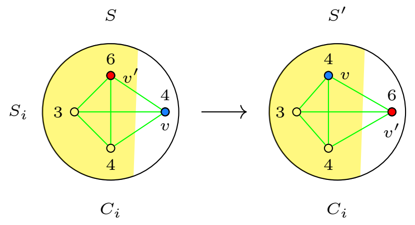


Figure 3.1: Illustration of Case 1 where v is a vertex with small threshold value but outside S. Here C_i is a clique type class and $S_i = S \cap C_i$ is shown in yellow. Clearly $d_{S'}(v) = d_S(v) - 1$. If v satisfies the threshold condition for S then it also satisfies the threshold condition for S'.

Case 2: Suppose that $v \in \{v_{|S_i|+1}, \ldots, v_{n_i}\}$, that is, v has a larger threshold value. If $v \notin S$ then $v \notin S'$, therefore $d_{S'}(v) = d_S(v) < t(v)$. If $v \in S$ then, to obtain S' from S we remove v from S and include a vertex $v' \in \{v_1, v_2, \ldots, v_{|S_i|}\} \setminus S$ having a smaller threshold value $t(v') \leq t(v)$. If C_i is an independent type class then $d_{S'}(v) = d_S(v) < t(v)$. If C_i is a clique type class, then we have $d_S(v') = d_S(v) + 1$ and also $d_{S'}(v) = d_S(v) + 1$. It implies that $d_{S'}(v) = d_S(v) + 1 = d_S(v') < t(v') \leq t(v)$. That is, v has fewer than $t(v)$ neighbours in S'. Therefore, S' is a harmless set. $\square$

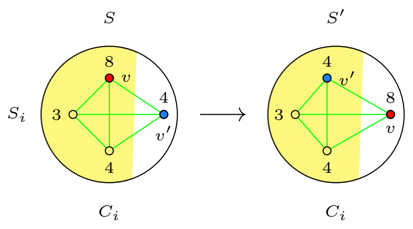


Figure 3.2: Illustration of Case 2 where v is a vertex with a larger threshold value but inside S. Here C_i is a clique type class and $S_i = S \cap C_i$ is shown in yellow.

In this section, we prove the following theorem:

Theorem 3.1.2. Harmless Set *can be solved in time* $d^{O(d)} \cdot n^{O(1)}$ *where* d *is the neighbourhood diversity of the input graph.*

Given a graph $G = (V, E)$ with neighbourhood diversity $\texttt{nd}(G) \leq d$, we first find a partition of the vertices into at most d type classes $C_1, \dots, C_d$. Let $\mathcal{C}$ be the set of all clique type classes and $\mathcal{I}$ be the set of all independent type classes. The case where some C_i are singletons can be considered as cliques or independent sets. For simplicity, we consider singleton type classes as independent sets.

ILP formulation: Our goal here is to find a largest harmless set S of G. For each C_i, we associate a variable x_i that indicates $|S \cap C_i| = x_i$. As the vertices in C_i have the same neighbourhood, the variables x_i determine S uniquely, up to isomorphism. The threshold $t(C_i)$ of a type class C_i is defined to be

$$t(C_i) = \min\{t(v) \mid v \in C_i\}.$$

Let $\alpha(C_i)$ be the number of vertices in C_i with threshold value $t(C_i)$. We define $\mathcal{C}_< = \{C_i \in \mathcal{C} \mid x_i < \alpha(C_i)\}$ and $\mathcal{C}_\geq = \{C_i \in \mathcal{C} \mid x_i \geq \alpha(C_i)\}$. We next guess if a clique type class C_i belongs to $\mathcal{C}_<$ or $\mathcal{C}_\geq$. There are at most 2^d guesses as each clique type class C_i has two options: either it is in $\mathcal{C}_<$ or in $\mathcal{C}_\geq$. We reduce the problem of finding a maximum harmless set to at most 2^d integer linear programming problems with d variables. Since integer linear programming is fixed-parameter tractable when parameterized by the number of variables [99], we conclude that our problem is FPT when parameterized by the neighbourhood diversity d. We consider the following cases based on whether C_i is in $\mathcal{I}, \mathcal{C}_<$ or $\mathcal{C}_\geq$. See Figure 3.3 for an illustration.

Case 1: Assume that C_i is in $\mathcal{I}$.

Lemma 3.1.3. Let C_i be an independent type class. Let u_0 be a vertex in C_i with threshold $t(C_i)$. Then every vertex u in C_i has fewer than $t(u)$ neighbours in S if and only if u_0 has fewer than $t(C_i)$ neighbours in S.

Proof. Suppose each $u \in C_i$ has fewer than $t(u)$ neighbours in S. Then obviously $u_0 \in C_i$ has fewer than $t(u_0) = t(C_i)$ neighbours in S. Conversely, suppose u_0 has fewer than $t(C_i)$ neighbours in S. Let u be an arbitrary vertex of C_i. As u and u_0 are two vertices in the same type class C_i, we have $d_S(u) = d_S(u_0)$. Moreover, for each $u \in C_i$, we have $t(C_i) \leq t(u)$ by definition of $t(C_i)$. Therefore, $d_S(u) = d_S(u_0) < t(C_i) \leq t(u)$. □

Note that $d_S(u_0) = \sum_{C_j \in N_H(C_i)} x_j$. By Lemma 3.1.3, every vertex u in C_i has fewer than $t(u)$ neighbours in S if and only if

$$\sum_{C_j \in N_H(C_i)} x_j < t(C_i).$$

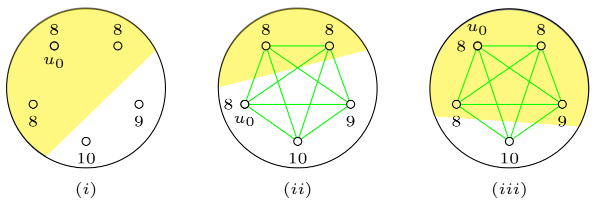


Figure 3.3: (*i*) Case 1: An independent type class C_i with $t(C_i) = 8$ and $x_i = 3$. (*ii*) Case 2: A clique type class C_i in $\mathcal{C}_<$ with $t(C_i) = 8$, $\alpha(C_i) = 3$ and $x_i = 2 < \alpha(C_i)$. (*iii*) Case 3: A clique type class C_i in $\mathcal{C}_\geq$ with $t(C_i) = 8$, $\alpha(C_i) = 3$ and $x_i = 4 \geq \alpha(C_i)$; $S \cap C_i$ is shown in yellow.

Case 2: Assume that C_i is in $\mathcal{C}_<$. That is, C_i is a clique type class and $x_i < \alpha(C_i)$. Assumption $x_i < \alpha(C_i)$ ensures that there exists at least one vertex in $S^c \cap C_i$ with threshold $t(C_i)$.

Lemma 3.1.4. Let $C_i \in \mathcal{C}_<$ and u_0 be a vertex in $S^c \cap C_i$ with threshold $t(C_i)$. Then every vertex u in C_i has fewer than $t(u)$ neighbours in S if and only if u_0 has fewer than $t(C_i)$ neighbours in S.

Proof. Suppose every vertex u in C_i has fewer than $t(u)$ neighbours in S. Then obviously u_0 has fewer than $t(u_0) = t(C_i)$ neighbours in S. Conversely, suppose u_0 has fewer than $t(C_i)$ neighbours in S. Let u be an arbitrary vertex of C_i. If $u \in S \cap C_i$, then Lemma 3.1.1 and the condition $x_i < \alpha(C_i)$ ensure u has threshold $t(C_i)$. Note that $d_S(u) = d_S(u_0) - 1 < t(C_i) - 1 < t(C_i) = t(u)$. If $u \in S^c \cap C_i$, then we have $d_S(u) = d_S(u_0) < t(C_i) \leq t(u)$. Therefore, every vertex in C_i satisfies the threshold condition. □

Note that $d_S(u_0) = x_i + \sum_{C_j \in N_H(C_i)} x_j$. By Lemma 3.1.4, every vertex u in C_i has fewer than $t(u)$ neighbours in S if and only if

$$x_i + \sum_{C_j \in N_H(C_i)} x_j < t(C_i).$$

Case 3: Assume that C_i is in $\mathcal{C}_{\geq}$. That is, C_i is a clique type class and $x_i \geq \alpha(C_i)$. By Lemma 3.1.1, all the vertices with threshold $t(C_i)$ are inside S.

Lemma 3.1.5. Let $C_i \in \mathcal{C}_{\geq}$ and u_0 be a vertex in $S \cap C_i$ with threshold $t(C_i)$. Then every vertex u in C_i has fewer than $t(u)$ neighbours in S if and only if u_0 has fewer than $t(C_i)$ neighbours in S.

Proof. Suppose every vertex u in C_i has fewer than $t(u)$ neighbours in S. Then obviously u_0 has fewer than $t(u_0) = t(C_i)$ neighbours in S. Conversely, suppose u_0 has fewer than $t(C_i)$ neighbours in S. Let u be an arbitrary vertex of C_i. If $u \in S \cap C_i$, then $d_S(u) = d_S(u_0) < t(C_i) \leq t(u)$. Suppose $u \in S^c \cap C_i$. Note that such an element u may not always exist, it is possible that all vertices in C_i are included in S (that is, $x_i = |C_i|$). Let us assume that such u exists. Since u is outside the solution and by Lemma 3.1.1, all the vertices with threshold $t(C_i)$ are inside the solution, we get $t(u) \geq t(C_i) + 1$. It is easy to note that $d_S(u) = d_S(u_0) + 1 < t(C_i) + 1 \leq t(u)$. □

Here $d_S(u_0) = (x_i - 1) + \sum_{C_j \in N_H(C_i)} x_j$. By Lemma 3.1.5, every vertex u in C_i has fewer than $t(u)$ neighbours in S if and only if

$$(x_i - 1) + \sum_{C_j \in N_H(C_i)} x_j < t(C_i).$$

The next lemma follows readily from the three lemmas above and the definition of the sequence $(x_1, x_2, \ldots, x_d)$ and harmless set.

Lemma 3.1.6. Let $G = (V, E)$ be a graph such that V can be partitioned into at most d type classes $C_1, \ldots, C_d$. The sequence $(x_1, x_2, \ldots, x_d)$ represents a harmless set S of G if and only if $(x_1, x_2, \ldots, x_d)$ satisfies

1. $x_i \in \{0, 1, \ldots, |C_i|\}$ for $i = 1, 2, \ldots, d$
2. $\sum_{C_j \in N_H(C_i)} x_j < t(C_i)$ for all $C_i \in \mathcal{I}$.
3. $x_i + \sum_{C_j \in N_H(C_i)} x_j < t(C_i)$ and $x_i < \alpha(C_i)$ for all $C_i \in \mathcal{C}_<$
4. $(x_i - 1) + \sum_{C_j \in N_H(C_i)} x_j < t(C_i)$ and $\alpha(C_i) \leq x_i \leq |C_i|$ for all $C_i \in \mathcal{C}_\geq$.

In the following, we present an ILP formulation for Harmless Set parameterized by neighbourhood diversity for a given guess:

$$\text{Maximize} \sum_{C_i} x_i$$

Subject to

$$x_i \in \{0, 1, \ldots, |C_i|\} \quad \text{for } i = 1, 2, \ldots, d$$

$$\sum_{C_j \in N_H(C_i)} x_j < t(C_i), \quad \text{for all } C_i \in \mathcal{I},$$

$$x_i + \sum_{C_j \in N_H(C_i)} x_j < t(C_i) \quad \text{and } x_i < \alpha(C_i) \text{ for all } C_i \in \mathcal{C}_<$$

$$(x_i - 1) + \sum_{C_j \in N_H(C_i)} x_j < t(C_i) \text{ and } \alpha(C_i) \leq x_i \leq |C_i| \quad \text{for all } C_i \in \mathcal{C}_\geq$$

Example 3. Consider a graph composed of a clique C of size 5 plus a vertex u adjacent to a vertex v of the clique as shown in Figure 3.4. We set unanimity thresholds, so $t(u) = 1$, $t(v) = 5$ and $t(x) = 4$ for all x in $C \setminus \{v\}$. The type classes are $C_1 = \{u\}$, $C_2 = \{v\}$ and

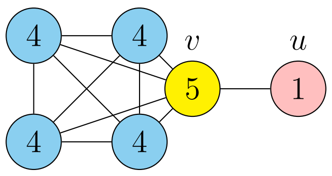


Figure 3.4: The graph in Example 3.

$C_3 = C \setminus \{v\}$. Here $\alpha(C_1) = 1$, $\alpha(C_2) = 1$ and $\alpha(C_3) = |C_3| = 4$. We guess $C_1, C_2 \in \mathcal{I}$ and $C_3 \in \mathcal{C}_{\geq}$. Then we end up with the following ILP:

$$\begin{aligned} \max \quad & x_1 + x_2 + x_3 \\ \text{s.t.} \quad & x_2 < 1 \\ & x_1 + x_3 < 5 \\ & x_3 - 1 + x_2 < 4 \text{ and } x_3 = 4 \end{aligned}$$

Note that $x_2 < 1$ implies that $x_2 = 0$; $x_1 + x_3 < 5$ and $x_3 = 4$ imply $x_1 = 0$. It is easy to see that $x_1 = x_2 = 0$, $x_3 = 4$ is an optimal solution and represent a valid harmless set for the graph.

Running time: In our formulation for Harmless Set, we have at most d variables. The value of the objective function is bounded by n and the value of any variable in the integer linear programming is also bounded by n. The constraints can be represented using $O(d^2 \log n)$ bits. There are at most 2^d guesses, and due to Lemma 2.3.2, each ILP formula for a given guess can be solved in time $d^{O(d)} \cdot n^{O(1)}$. Thus Theorem 6.1.1 holds.

3.1.2 Harmless Set Parameterized by Twin Cover

In this section, we present an FPT algorithm for Harmless Set parameterized by twin-cover. That is, we prove the following theorem:

Theorem 3.1.7. Harmless Set *can be solved in time* $2^{O(k2^k)} \cdot n^{O(1)}$ *where* k *is the twin-cover of the input graph.*

Outline of the algorithm. Given an n-vertex graph G with $\mathtt{tc}(G) \leq k$, we first find a twin cover X of size at most k. We next guess $S_X = S \cap X$ where S is a largest harmless set in G. Define $S^c = V \setminus S$. There are at most 2^k candidates for S_X as each member of X has two options: either in $S \cap X$ or $S^c \cap X$. Finally we reduce the problem of finding the rest of S to an integer linear programming (ILP) optimization with at most 2^k variables. Again, using ILP optimization which is fixed-parameter tractable when parameterized by the number of variables [47], we can conclude that our problem is fixed-parameter tractable when parameterized by the twin-cover number.

Characterizations of a harmless set S with a twin-cover X. Let $G = (V, E)$ be a graph and $X \subseteq V$ be a twin-cover of G. Then $\mathcal{C} = G - X$ is a collection of disjoint cliques, that is $\mathcal{C} = \{C_1, C_2, \ldots\}$. For the graph in Figure 3.5, we have $X = \{v_1, v_2\}$ and the disjoint cliques are $C_1 = \{v_3, v_4\}$, $C_2 = \{v_5\}$ and $C_3 = \{v_6, v_7, v_8\}$. The threshold $t(C_i)$ of a clique C_i is defined similar to the previous section to be

$$t(C_i) = \min\{t(v) \mid v \in C_i\}.$$

Let $\alpha(C_i)$ be the number of vertices in C_i with threshold value $t(C_i)$. It may be observed that from a clique it is always better to include the vertices with lower thresholds in the solution. If a and b are true twins with $t(a) < t(b)$ then any solution S containing b but not a can be replaced with a solution containing a but not b. If a is in the solution but not b, then a has one fewer neighbour in the solution compared to b. The fact that a has a lower degree in the solution helps a to satisfy the required threshold condition as $t(a) < t(b)$. We define $\mathcal{C}_{>} = \{C \in \mathcal{C} \ : \ \alpha(C) > t(C) - |N_{S_X}(C)|\}$ and $\mathcal{C}_{\leq} = \{C \in \mathcal{C} \ : \ \alpha(C) \leq t(C) - |N_{S_X}(C)|\}$ where $N_{S_X}(C)$ denotes the neighbourhood of C in S_X, that is, the set of all vertices in S_X adjacent to all elements of C. A clique $C \in \mathcal{C}$ is said to be S-*good* if every vertex $u \in C$ has fewer than $t(u)$ neighbours in S.

Lemma 3.1.8. Assume that C is in $\mathcal{C}_{>}$. Then C is S-good if and only if $|S \cap C| \leq t(C) - |N_{S_X}(C)| - 1$.

Proof. Suppose C is S-good and suppose for the sake of contradiction, that $|S \cap C| \geq t(C) - |N_{S_X}(C)|$. If $|S \cap C| = t(C) - |N_{S_X}(C)|$ then $|S \cap C| < \alpha(C)$. This implies there exists a vertex $u_0 \in S^c \cap C$ with $t(u_0) = t(C)$. Note that $d_S(u_0) = |S \cap C| + |N_{S_X}(C)| = t(C) = t(u_0)$, a contradiction to the assumption that C is S-good. Let us assume that $|S \cap C| \geq t(C) - |N_{S_X}(C)| + 1$. Let u be an arbitrary vertex in C. Then $d_S(u) \geq |S \cap$

$C| - 1 + |N_{S_X}(C)| \geq t(C) \geq t(u)$, which is again a contradiction. This proves the forward direction.
On the other hand, let us assume that $|S \cap C| \leq t(C) - |N_{S_X}(C)| - 1$. It implies that $|S \cap C| < \alpha(C)$ and this ensures that there exists at least one vertex u_0 in $S^c \cap C$ with threshold $t(C)$. By Lemma 3.1.4, it is enough to check whether $u_0 \in S^c \cap C$ with threshold $t(C)$ satisfies the threshold condition. Clearly, $d_S(u_0) = |S \cap C| + |N_{S_X}(C)| \leq t(C) - |N_{S_X}(C)| - 1 + |N_{S_X}(C)| = t(C) - 1$ satisfies the threshold condition. Therefore, C is S-good. □

Lemma 3.1.9. Assume that C is in $\mathcal{C}_{\leq}$. Then C is S-good if and only if $|S \cap C| \leq t(C) - |N_{S_X}(C)|$.

Proof. Suppose C is S-good. Let $u_0 \in C$ with threshold $t(u_0) = t(C)$. Then u_0 also has fewer than $t(u_0)$ neighbours in S, that is, $d_S(u_0) \leq |S \cap C| + N_{S_X}(C) < t(u_0) = t(C)$. Therefore, we get $|S \cap C| \leq t(C) - |N_{S_X}(C)|$.

On the other hand, first suppose that $|S \cap C| = t(C) - |N_{S_X}(C)|$. Therefore, we can say that $|S \cap C| \geq \alpha(C)$. It means that all the vertices with least threshold are inside the solution. Let u be an arbitrary vertex in $S \cap C$ with threshold $t(u)$. By Lemma 3.1.5, it is enough to check whether u satisfies the threshold condition. It is easy to observe that $d_S(u) = |S \cap C| - 1 + |N_{S_X}(C)| = t(C) - 1 < t(C) = t(u)$, that is, u satisfies the threshold condition. Now suppose $|S \cap C| = t(C) - |N_{S_X}(C)| - \delta$ for some integer $\delta \geq 1$. Take an arbitrary vertex $u \in C$. We have $d_S(u) \leq |S \cap C| + |N_{S_X}(C)| = t(C) - \delta < t(C) \leq t(u)$. This implies that C is S-good. □

We partition the family $\mathcal{C}$ of cliques into twin classes $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_t$, where $t \leq 2^k$. Two cliques C_i and C_j are in the same twin class if and only if they have the same neighbours in X, that is, $N_X(C_i) = N_X(C_j)$. For example, the graph G in Figure 3.5 has two twin classes $\mathcal{C}_1$ and $\mathcal{C}_2$ where $\mathcal{C}_1$ consists of cliques $C_1 = \{v_3, v_4\}$, $C_2 = \{v_5\}$ and $\mathcal{C}_2$ consists of $C_3 = \{v_6, v_7, v_8\}$. For each twin class $\mathcal{C}_i$, we associate a variable x_i that indicates the number of vertices from $\mathcal{C}_i$ that are in the solution S, that is, $x_i = \sum\limits_{C \in \mathcal{C}_i} |C \cap S|$. The variables x_i determine S. The objective is to maximize $\sum\limits_{i=1}^{t} x_i$ under the conditions

$$0 \leq x_i \leq \sum_{C \in \mathcal{C}_{>} \cap \mathcal{C}_i} \Big(t(C) - |N_{S_X}(C)| - 1\Big) + \sum_{C \in \mathcal{C}_{\leq} \cap \mathcal{C}_i} \Big(t(C) - |N_{S_X}(C)|\Big)$$

and the additional conditions described below. The above constraints make sure that the vertices of twin class $\mathcal{C}_i$ satisfy the threshold condition. Next, we add k more constraints to make sure that all the vertices in X satisfy threshold condition. Let $u \in X$ be an arbitrary vertex. We need $d_S(u) < t(u)$. Therefore, for each $u \in X$, we add the constraint $d_S(u) = d_{S_X}(u) + \sum_{\mathcal{C}_i \,:\, N(u) \cap C \neq \emptyset \ \forall \ C \in \mathcal{C}_i} x_i < t(u)$. In the following, we present an ILP formulation for Harmless Set for a given S_X:

$$\text{Maximize} \sum_{i=1}^{t} x_i, \ t \leq 2^k$$

Subject to

$$0 \leq x_i \leq \sum_{C \in \mathcal{C}_> \cap \mathcal{C}_i} \Big(t(C) - |N_{S_X}(C)| - 1\Big) + \sum_{C \in \mathcal{C}_\leq \cap \mathcal{C}_i} \Big(t(C) - |N_{S_X}(C)|\Big) \qquad \forall \mathcal{C}_i$$

$$d_S(u) = d_{S_X}(u) + \sum_{\mathcal{C}_i \,:\, N(u) \cap C \neq \emptyset \ \forall \ C \in \mathcal{C}_i} x_i < t(u) \qquad \forall u \in X$$

Let $(x_1, x_2, \ldots, x_t)$ be an optimal solution of ILP for a given S_X where x_i is the number of vertices from the twin class $\mathcal{C}_i$ that are in S. In order to obtain S, we need to choose x_i vertices from the cliques in $\mathcal{C}_i$ for all i. We arbitrarily choose at most $t(C) - |N_{S_X}(C)|$ vertices from the clique C if $C \in \mathcal{C}_\leq \cap \mathcal{C}_i$ and at most $t(C) - |N_{S_X}(C)| - 1$ vertices from the clique C if $C \in \mathcal{C}_> \cap \mathcal{C}_i$ so that the total number of vertices selected from $\mathcal{C}_i$ is x_i. Note that such a selection is possible because of the constraint on x_i in the ILP.

Running time: In our ILP formulation for Harmless Set parameterized by twin-cover, we have at most 2^k variables. The value of the objective function is bounded by n and the value of any variable in the integer linear programming is also bounded by n. The constraints can be represented using $O(2^k \log n)$ bits. Lemma 2.3.2 implies that we can solve the problem for a given S_X in time $2^{O(k2^k)} \cdot n^{O(1)}$. As there are at most 2^k candidates for S_X, Theorem 3.1.7 holds.

Example 4. *Consider the graph shown in Figure 3.5. We set unanimity thresholds, so* $t(v_5) = 2$, $t(v_2) = 6$, *and* $t(v_1) = t(v_3) = t(v_4) = t(v_6) = t(v_7) = t(v_8) = 3$. *Here* $X =$

$\{v_1, v_2\}$ is a minimum size twin-cover and $G \setminus X$ is the collection of disjoint cliques $C_1 = \{v_3, v_4\}$, $C_2 = \{v_5\}$ and $C_3 = \{v_6, v_7, v_8\}$. These three cliques are partitioned into two twin classes $\mathcal{C}_1$ and $\mathcal{C}_2$ where $\mathcal{C}_1$ consists of cliques C_1, C_2 and $\mathcal{C}_2$ consists of C_3. We have $t(C_1) = 3$, $t(C_2) = 2$, $t(C_3) = 3$ and $\alpha(C_1) = 2$, $\alpha(C_2) = 1$, $\alpha(C_3) = 3$. Take $S_X = \{v_1\}$,

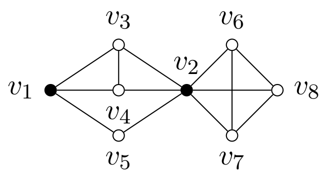


Figure 3.5: The graph in Example 4.

that is, v_1 is in the solution. Given $S_X = \{v_1\}$, we observe that C_1, C_2, C_2 are in $\mathcal{C}_{\leq}$ and we end up with the following ILP:

$$\begin{aligned} \textit{max} \quad & x_1 + x_2 \\ \textit{such that} \quad & 0 \leq x_1 \leq (t(C_1) - |N_{S_X}(C_1)|) + (t(C_2) - |N_{S_X}(C_2)|) = 3 \qquad \textit{for } \mathcal{C}_1 \\ & 0 \leq x_2 \leq (t(C_3) - |N_{S_X}(C_3)|) = 3 \qquad \textit{for } \mathcal{C}_2 \\ & d_S(u_1) = x_1 < 3 \\ & d_S(u_2) = x_1 + x_2 < 6 \end{aligned}$$

It is easy to see that $x_1 = 2$, $x_2 = 3$ is an optimal solution. As $x_1 = 2$ we randomly choose two vertices say v_3 and v_5 from $\mathcal{C}_1$ and as $x_2 = 3$ we choose three vertices v_6, v_7, v_8 from $\mathcal{C}_2$. Thus $x_1 = 2$, $x_2 = 3$ represent a valid harmless set $\{v_1, v_3, v_5, v_6, v_7, v_8\}$ of size 6 for the graph. Similarly for $S_X = \{v_2\}$ and $S_X = \emptyset$, we get valid harmless sets of size 5. It may be noted that $S_X = \{v_1, v_2\}$ is not a valid choice as v_5 does not satisfy the threshold condition if both v_1 and v_2 are in the solution. Therefore G has a maximum harmless set of size 6.

3.1.3 Harmless Set Parameterized by Vertex Integrity

In this section, we present an FPT algorithm for Harmless Set parameterized by vertex integrity. This vulnerability measure was introduced by Barefoot et al. [7] in 1987. For an overview of structural results on vertex integrity, we refer the reader to a survey on vertex integrity by Bagga et al. [6].

Theorem 3.1.10. Harmless Set *can be solved in time* $2^{2^{O(k^2)}} \cdot n^{O(1)}$ *where* k *is the vertex integrity of the input graph.*

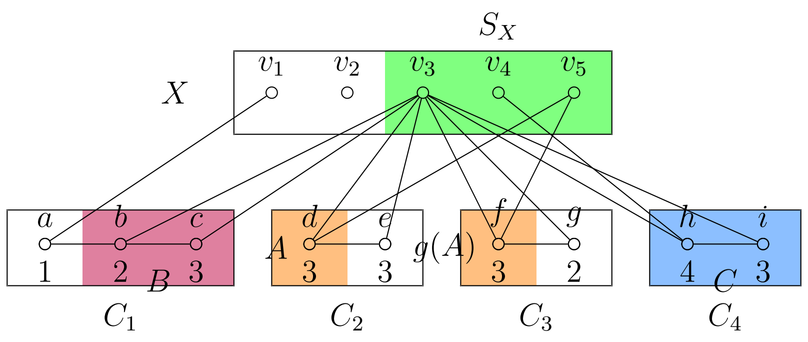
Outline of the algorithm. Let X be a k-vi-set of G. Such a set can be found in $O(k^{k+1}n)$ time [37]. We next guess $S_X = S \cap X$ where S is a largest harmless set in G. There are at most 2^k candidates for S_X as each member of X has two options: either in $S \cap X$ or $S^c \cap X$. Finally we reduce the problem of finding the rest of S to an integer linear programming (ILP) optimization with number of variables depend only on k.

Characterizations of a harmless set S *with a* k-vi-*set* X*.* Let $G = (V, E)$ be a graph and $X \subseteq V$ be a k-vi-set of G. Then $\mathcal{C} = G - X$ is a collection of disjoint components, that is $\mathcal{C} = \{C_1, C_2, \ldots\}$ such that $|X| + |C_i| \leq k$ for all i. We know $\mathcal{C}$ can be partitioned into equivalent classes $\mathcal{C}_1, \mathcal{C}_2, \ldots$. Let C_l be a representative of the equivalence class $\mathcal{C}_l$ and let $v \in C_l$. Note that v has neighbours only in $X \cup V(C_l)$, that is, $N(v) \subseteq X \cup V(C_l)$. Suppose the intersection of the solution S with X is $S_X = S \cap X$ and the intersection of S with the component C_l is $A = S \cap C_l \subseteq C_l$. Therefore $v \in C_l$ satisfies the threshold condition if $d_S(v) = |N_{S_X}(v)| + |N_A(v)| < t(v)$. We say $A \subseteq C_l$ is a *valid selection* for C_l if every vertex of C_l satisfies the threshold condition when the vertices of $A \cup S_X$ are in the solution. Similarly, we say $A \subseteq C_l$ is a *valid selection* for $C \in \mathcal{C}_l$, $C \neq C_l$, if every vertex of C satisfies the threshold condition when the vertices of $g(A) \cup S_X$ are in the solution, where g is an isomorphism from $G[X \cup V(C_l)]$ to $G[X \cup V(C)]$ that fixes X. It is important to note that given two connected component C_1 and C_2 from the same equivalence class $\mathcal{C}_l$, a subset $A \subseteq C_l$ might be valid selection for one connected component but may not be valid for the other connected component as the threshold values of vertices in C_1 and C_2 can differ.

Given $S_X = S \cap X$, for every equivalence class $\mathcal{C}_l$, we define the set of valid selection

$$\mathcal{VS}(l) = \Big\{ A \subseteq C_l \mid A \text{ is a valid selection for some } C \in \mathcal{C}_l \Big\}.$$

We denote by $\rho(A, l)$ the number of components in $\mathcal{C}_l$ where A is a valid selection. Similarly, we denote by $\rho(A_{i_1}, A_{i_2}, \ldots, A_{i_r}, l)$ the number of components in $\mathcal{C}_l$ where $A_{i_1}, A_{i_2}, \ldots, A_{i_r}$ are valid selection. Note that for each component there may be more than one valid selections. But while forming the harmless set we can pick at most one valid selection for every component. Observe that as long as we are taking exactly one valid selection for each



S_X
X
v_1 v_2 v_3 v_4 v_5
a b c d e f g h i
1 2 3 3 3 3 2 4 3
A
$g(A)$
B
C
C_1 C_2 C_3 C_4

Figure 3.6: The components C_2 and C_3 of $G - X$ have the same $(G, X)-$type. That is, C_2 and C_3 are in the same equivalence class. Here the four components are partitioned into three equivalent classes $\mathcal{C}_1 = \{C_1\}$, $\mathcal{C}_2 = \{C_2, C_3\}$ and $\mathcal{C}_3 = \{C_4\}$. Note that given $S_X = \{v_3, v_4, v_5\}$, $A = \{d\}$ is a valid selection for C_2 but $g(A) = \{f\}$ is not a valid selection for C_3, thus $\rho(A, \mathcal{C}_2) = 1$; B is not a valid selection for C_1; C is a valid selection for C_4.

connected component, we are guaranteed that all the vertices in $G - X$ satisfy the threshold conditions. We need to add constraints in ILP separately to make sure that every vertex of X also satisfies the threshold condition. Let $x(A, l)$ denote the number of components in $\mathcal{C}_l$ for which A has been picked as a valid selection in the final solution. They satisfy the following properties:

$$x(A_i, l) \leq \rho(A_i, l) \quad \text{for all} \quad A_i \in \mathcal{VS}(l)$$

$$x(A_i, l) + x(A_j, l) \leq \rho(A_i, l) + \rho(A_j, l) - \rho(A_i, A_j, l) \quad \text{for all} \quad A_i, A_j \in \mathcal{VS}(l)$$

$$\vdots$$

$$\sum_{A_i \in \mathcal{VS}(l)} x(A_i, l) = |\mathcal{C}_l|$$

Example 5. *Suppose the equivalent class $\mathcal{C}_l$ has three components C_1, C_2, C_3. Suppose A_1, A_2, A_3 are valid selections for the component C_1; A_1, A_3 are valid selections for the component C_2, and A_2, A_3 are valid selections for the component C_3. Clearly, $\rho(A_1, l) = 2$, $\rho(A_2, l) = 2$, $\rho(A_3, l) = 3$, $\rho(A_1, A_2, l) = 1$, $\rho(A_1, A_3, l) = 2$, $\rho(A_2, A_3, l) = 2$ and $\rho(A_1, A_2, A_3, l) = 1$. Then $x(A_1, l)$, $x(A_2, l)$ and $x(A_3, l)$ satisfy the following constraints:*

1. $x(A_1, l) \leq 2, x(A_2, l) \leq 2, x(A_3, l) \leq 3$

2. $x(A_1, l) + x(A_2, l) \leq \rho(A_1, l) + \rho(A_2, l) - \rho(A_1, A_2, l) = 2 + 2 - 1 = 3$
   $x(A_1, l) + x(A_3, l) \leq \rho(A_1, l) + \rho(A_3, l) - \rho(A_1, A_3, l) = 2 + 3 - 2 = 3$

$$x(A_2,l) + x(A_3,l) \leq \rho(A_2,l) + \rho(A_3,l) - \rho(A_2,A_3,l) = 2+3-2 = 3$$

3. $x(A_1,l) + x(A_2,l) + x(A_3,l) = |\mathcal{C}_l| = 3$

Therefore the possible solutions are $(x(A_1,l), x(A_2,l), x(A_3,l)) = (0,0,3)$*,* $(0,1,2)$*,* $(0,2,1)$*,* $(1,1,1)$*,* $(1,0,2)$*,* $(1,2,0)$*,* $(2,0,1)$*,* $(2,1,0)$*. Note that* $(0,0,3)$ *indicates* A_3 *is valid selection in three components* C_1*,* C_2 *and* C_3*; similarly* $(0,1,2)$ *indicates* A_2 *is a valid selection in one component and* A_3 *is a valid selection in two components.*

In the following, we present an ILP formulation for Harmless Set for a given S_X:

$$\text{Maximize} \sum_{l} \sum_{A \in \mathcal{VS}(l)} |A| \times x(A,l)$$

Subject to

$$x(A_i,l) \leq \rho(A_i,l) \qquad \forall\ l \ \&\ \forall\ A_i \in \mathcal{VS}(l)$$

$$x(A_i,l) + x(A_j,l) \leq \rho(A_i,l) + \rho(A_j,l) - \rho(A_i,A_j,l) \quad \forall\ l \ \&\ \forall\ A_i, A_j \in \mathcal{VS}(l)$$

$$\vdots$$

$$\sum_{A_i \in \mathcal{VS}(l)} x(A_i,l) = |\mathcal{C}_l| \qquad \forall\ l$$

$$\sum_{l} \sum_{A \in \mathcal{VS}(l)} |N(u) \cap A| \times x(A,l) < t(u) \qquad \forall \quad u \in X$$

Running time: In our ILP formulation for Harmless Set parameterized by vertex integrity, the number of variables is bounded by $2^{O(k^2)}$. The reason is this. The number of equivalence classes is bounded by $2^{O(k^2)}$ as each component of $G - X$ has at most k vertices and hence at most $O(k^2)$ edges. Also for each equivalence class the number of valid intersections is bounded by 2^k as choosing a subset of vertices fixes the intersection. This implies that the number of variables is at most $2^{O(k^2)}$. The value of the objective function is bounded by n and the value of any variable in the integer linear programming is also bounded by n. The constraints can be represented using $2^{O(k^2)} \cdot \log n$ bits. Lemma 2.3.2 implies that we can solve the problem for a given S_X in time $2^{2^{O(k^2)}} \cdot n^{O(1)}$. There are at most 2^k candidates for S_X, thus Theorem 3.1.10 holds.

3.2 W[1]-Hardness results

3.2.1 Harmless Set Parameterized by Treewidth

Bazgan and Chopin [10] proved that Harmless Set is FPT when parameterized by the combined parameters treewidth and solution size. However, it remains open whether the problem might belong to FPT when parameterized only by the treewidth of the input graph. In this section we resolve this open problem by showing that Harmless Set is indeed W[1]-hard when parameterized by treewidth alone, even when restricted to bipartite graphs. This result is also proved simultaneously and independently by Drange, Muzi and Reidl [38]. Our proof is very different from their proof. First, we show that Harmless Set is W[1]-hard parameterized by a vertex deletion set to trees of height at most three, that is, a subset D of the vertices of the graph such that every component in the graph, after removing D, is a tree of height at most three. Clearly trees of height at most three are trivially acyclic. It is easy to verify that such trees have pathwidth [93] at most three and hence treewidth at most three, which implies that Harmless Set is W[1]-hard when parameterized by the treewidth of the input graph.

We show our hardness result for Harmless Set using reduction from Multidimensional Relaxed Subset Sum (MRSS).

Multidimensional Subset Sum (MSS)
Input: An integer k, a set $S = \{s_1, \ldots, s_n\}$ of vectors with $s_i \in \mathbb{N}^k$ for every i with $1 \leq i \leq n$ and a target vector $g \in \mathbb{N}^k$.
Parameter: k
Question: Is there a subset $S' \subseteq S$ such that $\sum\limits_{s \in S'} s = g$?

We consider a variant of MSS that we require in our proofs. In the Multidimensional Relaxed Subset Sum (MRSS) problem, an additional integer k' is given (which will be part of the parameter) and we ask whether there is a subset $S' \subseteq S$ with $|S'| \leq k'$ such that $\sum\limits_{s \in S'} s \geq g$. This variant can be formalized as follows:

Multidimensional Relaxed Subset Sum (MRSS)
Input: An integer k, a set $S = \{s_1, \ldots, s_n\}$ of vectors with $s_i \in \mathbb{N}^k$ for every i with $1 \leq i \leq n$, a target vector $g \in \mathbb{N}^k$ and an integer k'.
Parameter: $k + k'$
Question: Is there a subset $S' \subseteq S$ with $|S'| \leq k'$ such that $\sum\limits_{s \in S'} s \geq g$?

It is known that MRSS is W[1]-hard when parameterized by the combined parameter $k+k'$, even if all integers in the input are given in unary [74]. In this section, we prove the following theorem:

Theorem 3.2.1. Harmless Set *is W[1]-hard when parameterized by the size of a vertex deletion set into trees of height at most 3, even when restricted to bipartite graphs.*

Proof. Let (k, k', S, g) be an instance of MRSS. From this we construct an instance (G, t, r) of Harmless Set the following way. We introduce three types of vertices: necessary vertices, forbidden vertices and normal vertices. We want to make sure that *necessary vertices* are always inside every solution and *forbidden vertices* are always outside every solution; and a *normal vertex* could be inside or outside the solution. For each vector $s \in S$, we introduce a tree T^s of height three. For $s \in S$, $\max(s)$ is the value of the largest coordinate of s and $\max(S)$ is maximum of $\max(s)$ values. The tree T^s consists of vertices $V(T^s) = A^s \cup B^s \cup \{c^s\}$ where the vertices in $A^s = \{a^s_1, \ldots, a^s_{\max(S)}\}$ and c^s are normal vertices and the vertices in $B^s = \{b^s_1, \ldots, b^s_{\max(S)}\}$ are necessary vertices. Make c^s adjacent to every vertex of B^s. We also make a^s_i adjacent to b^s_i for all $1 \leq i \leq \max(s)$. Next, we create a set $U = \{u_1, \ldots, u_k\}$ of k necessary vertices into G. For each $1 \leq i \leq k$ and $s \in S$, make u_i adjacent to exactly $s(i)$ many vertices of A^s arbitrarily. We introduce three cycles C_1, C_2, C_3 of length four where $V(C_1) = \{a_1, a_2, a_3, a_4\}$, $V(C_2) = \{b_1, b_2, b_3, b_4\}$ and $V(C_3) = \{c_1, c_2, c_3, c_4\}$. We will prove why the vertices of these three cycles are always outside every solution. We make a_1 adjacent to every vertex of $\bigcup\limits_{s \in S} A^s$, make b_1 adjacent to every vertex of $\bigcup\limits_{s \in S} B^s$ and make c_1 adjacent to every vertex of $\bigcup\limits_{s \in S} \{c^s\}$. This completes the construction of graph G. Note that G is a bipartite graph with bipartition

$$V_1 = U \cup \{a_1, a_3, b_2, b_4, c_1, c_3\} \cup \bigcup_{s \in S} B^s$$

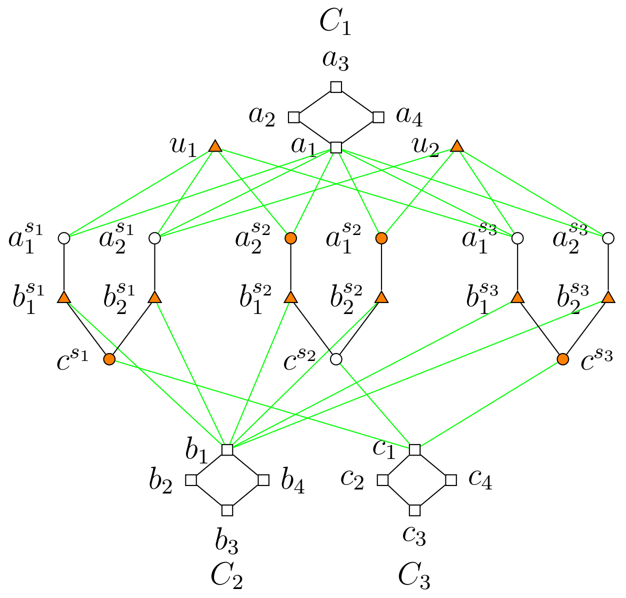


Figure 3.7: The graph G in the proof of Theorem 3.2.1 constructed from MRSS instance $S = \{(2,1),(1,1),(1,2)\}, g = (3,3), k = 2, k' = 2$. The set $S' = \{(2,1),(1,2)\}$ forms a solution of MRSS instance and the set $H = \{u_1,u_2\} \cup B^{s_1} \cup B^{s_2} \cup B^{s_3} \cup A^{s_2} \cup \{c^{s_1}, c^{s_3}\}$ forms a harmless set in G.

and

$$V_2 = \{a_2, a_4, b_1, b_3, c_2, c_4\} \cup \bigcup_{s\in S} A^s \cup \{c^s\}.$$

We observe that if we delete the set $U \cup \{a_1,a_2,a_3,a_4,b_1,b_2,b_3,b_4,c_1\}$ of size $k+9$ from G then we are left with trees of height at most three. We define the threshold function as follows:

$$t(u) = \begin{cases} d(u) & \text{if } u \in \{b_1\} \bigcup_{s\in S} A^s \cup \{c^s\} \\ 1 & \text{if } u \in \{a_2,a_3,a_4,b_2,b_3,b_4,c_2,c_3,c_4\} \\ 2 & \text{if } u \in \bigcup_{s\in S} B^s \\ (n-k')\cdot \max(S)+1 & \text{if } u = a_1 \\ k'+1 & \text{if } u = c_1 \\ \sum_{s\in S} s(i) - g(i) + 1 & \text{if } u = u_i \text{ for all } 1 \le i \le k \end{cases}$$

We set $r = k + n\cdot \max(S) + (n-k')\cdot \max(S) + k'$. It may be noted that the sets $\{c^s\}$

and A^s play the role of complimentary sets, that is, only one of them can contribute to the solution. If both $\{c^s\}$ and A^s contribute to the solution then a vertex in B^s will fail to satisfy the threshold condition. Also we argue in the later part of the proof that either we have $\{c^s\} \subseteq H$ and $A^s \cap H = \emptyset$, or $A^s \subseteq H$ and $\{c^s\} \cap H = \emptyset$ for all $s \in S$. The intention is that for each solution $S' \subseteq S$ of instance I we have a solution candidate H in I' such that $s \in S'$ entails $\{c^s\} \subseteq H$ and $A^s \cap H = \emptyset$, and $s \notin S'$ entails $A^s \subseteq H$ and $\{c^s\} \cap H = \emptyset$.

Now we show that our reduction is correct. That is, we claim (k, k', S, g) is a yes instance of MRSS if and only if (G, t, r) is a yes instance of Harmless Set. Towards showing the forward direction, let $S' \subseteq S$ be such that $|S'| \leq k'$ and $\sum\limits_{s \in S'} s \geq g$. We claim that the set

$$H = U \cup \bigcup_{s \in S} B^s \cup \bigcup_{s \in S \setminus S'} A^s \cup \bigcup_{s \in S'} \{c^s\}$$

is a harmless set of size at least r. It is easy to see that $|H| \geq r$. Next, we show that all the vertices in G satisfy the threshold condition. Let u be an arbitrary vertex of G. If $u = u_i$ is an element of U, then $d(u_i) = \sum\limits_{s \in S} s(i)$ and $\sum\limits_{s \in S'} s \geq g$. Hence, we get $d_H(u_i) \leq \sum\limits_{s \in S} s(i) - g(i) < t(u_i) = \sum\limits_{s \in S} s(i) - g(i) + 1$. If u is an element of $\bigcup\limits_{s \in S} A^s$, then $d_H(u) = d(u) - 1 < d(u) = t(u)$ as $a_1 \notin H$. Similarly, every vertex of $\bigcup\limits_{s \in S} \{c_s\}$ satisfies the threshold condition as $c_1 \notin H$. For each $b_i^s \in \bigcup\limits_{s \in S} B^s$, we have either a_i^s or c^s inside the solution and $b_1 \notin H$. Thus $d_H(b_i^s) = 1 < 2 = t(b_i^s)$. Hence all the vertices in $\bigcup\limits_{s \in S} B^s$ satisfy the threshold condition. As $|S'| \leq k'$, we see that c_1 has at most k' neighbours inside H, thus $d_H(c_1) = k' < k' + 1 = t(c_1)$. For the rest of the vertices in H it is easy to verify that the threshold condition is satisfied.

Towards showing the reverse direction of the claim, let H be a harmless set of size at least r in G. We need the following three simple observations. First, $H \cap \{a_i, b_i, c_i \mid 1 \leq i \leq 4\} = \emptyset$. Otherwise one of the vertices of $\{a_i, b_i, c_i \mid 2 \leq i \leq 4\}$ will fail to satisfy the threshold condition. Therefore the vertices in $\{a_i, b_i, c_i \mid 1 \leq i \leq 4\}$ are forbidden vertices. Second, for each $s \in S$, $\{c^s\}$ and A^s play the role of complimentary sets, that is, only one of them can contribute to the solution H. If both $\{c^s\}$ and A^s contribute to H then some vertex b in B^s will have two neighbours in the solution but $t(b) = 2$. Therefore b will not satisfy the threshold condition, a contradiction to the assumption that H is a harmless set. Third, the vertices of U and $\bigcup\limits_{s \in S} B^s$ are always inside every solution of size r. Because of the first two observations, it would not be possible to get a harmless set of size at least r unless we include all the vertices of U and $\bigcup\limits_{s\in S} B^s$ in the solution. Clearly U and $\bigcup\limits_{s\in S} B^s$ can contribute to the solution at most k and $n \cdot \max(S)$ vertices respectively. We also observe that the set $\bigcup\limits_{s\in S} A^s$ can contribute at most $(n-k') \cdot \max(S)$ vertices to the solution due to the fact that $t(a_1) = (n-k') \cdot \max(S) + 1$. Therefore, the only way to have a harmless set of size at least r is that $\bigcup\limits_{s\in S} \{c^s\}$ contributes at least k' elements to H. Since $t(c_1) = k' + 1$, the set $\bigcup\limits_{s\in S} \{c^s\}$ can contribute at most k' elements to H. Therefore, the set $\bigcup\limits_{s\in S} \{c^s\}$ contributes exactly k' elements to H. We define

$$S' = \{s \in S \mid c^s \in H\}.$$

From here, we see that $H = U \cup \bigcup\limits_{s\in S} B^s \bigcup\limits_{s\in S\setminus S'} A^s \bigcup\limits_{s\in S'} \{c^s\}$. Since each $u_i \in U$ satisfies the threshold condition, we have $d_H(u_i) = \sum\limits_{s\in S\setminus S'} s(i) = \sum\limits_{s\in S} s(i) - \sum\limits_{s\in S'} s(i) < t(u_i) = \sum\limits_{s\in S} s(i) - g(i) + 1$. This implies that $\sum\limits_{s\in S'} s(i) \geq g(i)$ for all $1 \leq i \leq k$. Therefore (k, k', S, g) is a yes-instance. □

Clearly trees of height at most three are trivially acyclic. Moreover, it is easy to verify that such trees have pathwidth [93] and treedepth [110] at most three, which implies:

Theorem 3.2.2. Harmless Set *is W[1]-hard when parameterized by any of the following parameters:*

- *the feedback vertex set number,*
- *the pathwidth and hence also treewidth of the input graph,*
- *the size of a minimum set of vertices whose deletion results in components of pathwidth/treedepth at most three,*

even when restricted to bipartite graphs.

3.2.2 Harmless Set Parameterized by Cluster Vertex Deletion Number

We prove that Harmless Set is W[1]-hard when parameterized by the cluster vertex deletion number of the input graph. The *cluster vertex deletion number* of a graph is the minimum number of its vertices whose deletion results in a disjoint union of complete graphs [34]. This generalizes the vertex cover number, provides an upper bound to the clique-width and is related to the previously studied notion of the twin cover of the graph under consideration. We show our hardness result for Harmless Set using reduction from Multidimensional Relaxed Subset Sum (MRSS). We prove the following theorem:

Theorem 3.2.3. Harmless Set *is W[1]-hard when parameterized by the cluster vertex deletion number of the input graph.*

Proof. Let (k, k', S, g) be an instance of MRSS. From this we construct an instance (G, t, r) of Harmless Set the following way. We introduce two types of vertices: forbidden vertices and normal vertices. We want to make sure that *forbidden vertices* are always outside every solution and a *normal vertex* could be inside or outside the solution.

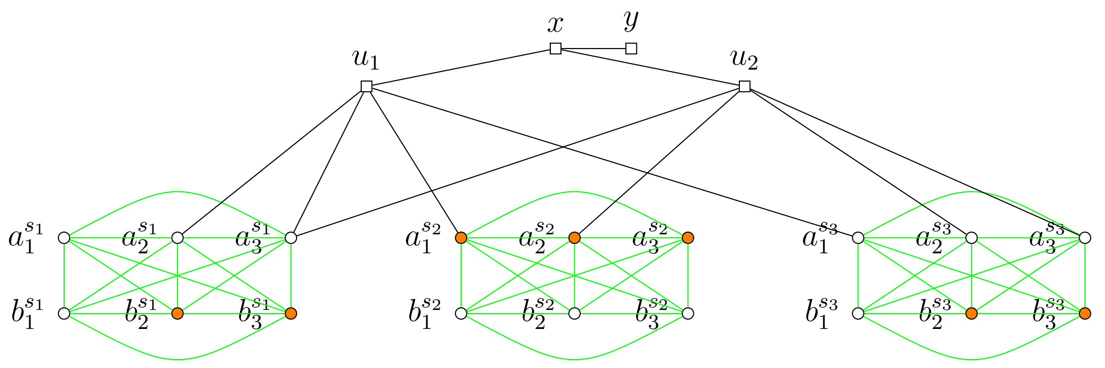


Figure 3.8: The graph G in the proof of Theorem 3.2.3 constructed from MRSS instance $S = \{(2,1),(1,1),(1,2)\}, g = (3,3), k = 2, k' = 2$. The set $S' = \{(2,1),(1,2)\}$ forms a solution of MRSS instance and the set $H = B^{s_1} \setminus b_1^{s_1} \cup B^{s_3} \setminus b_1^{s_3} \cup A^{s_2}$ forms a harmless set in G.

For each vector $s \in S$, we introduce a clique C^s. For $s \in S$, $\max(s)$ is the value of the largest coordinate of s and $\max(S)$ is maximum of $\max(s)$ values. The clique C^s consists of normal vertices $V(C^s) = A^s \cup B^s$ where $A^s = \{a_1^s, \dots, a_{\max(S)+1}^s\}$ and $B^s = \{b_1^s, \dots, b_{\max(S)+1}^s\}$. We introduce a set of k forbidden vertices $U = \{u_1, \dots, u_k\}$ into G. For each $1 \le i \le k$ and $s \in S$, make u_i adjacent to exactly $s(i)$ many vertices of A^s arbitrarily. Finally, we add two forbidden vertices x and y; make x adjacent to every vertex of U and y. This completes the construction of graph G. Note that deletion of the set U of size k from G

results in a disjoint union of complete graphs. We define the threshold function as follows:

$$t(u) = \begin{cases} 1 & \text{if } u \in \{x, y\} \\ \max(S) + 1 & \text{if } u \in \bigcup\limits_{s \in S} A^s \\ \max(S) + 2 & \text{if } u \in \bigcup\limits_{s \in S} B^s \\ \sum\limits_{s \in S} s(i) - g(i) + 1 & \text{if } u = u_i \text{ for all } 1 \le i \le k \end{cases}$$

We set $r = k' \cdot \max(S) + (n - k') \cdot (\max(S) + 1)$. The vertices of $U \cup \{x, y\}$ are always outside every solution as $t(x) = t(y) = 1$. Therefore the solution can take vertices only from n cliques. We shall observe that $|H \cap (A^s \cup B^s)| \le \max(S) + 1$ and this bound can be achieved if and only if $H \cap (A^s \cup B^s) = A^s$.

Now we show that our reduction is correct. That is, we claim (k, k', S, g) is a yes-instance of MRSS if and only if (G, t, r) is a yes-instance of Harmless Set. Towards showing the forward direction, let $S' \subseteq S$ be such that $|S'| \le k'$ and $\sum\limits_{s \in S'} s \ge g$. We claim that

$$H = \bigcup_{s \in S'} B^s \backslash \{b_1^s\} \cup \bigcup_{s \in S \backslash S'} A^s$$

is a harmless set in G of size at least r. Clearly $|H| \ge r$. Next, we show that each vertex of G satisfies the threshold condition. Since x and y have no neighbours in H and $t(x) = t(y) = 1$, they satisfy the threshold condition. Since each vertex of $\bigcup\limits_{s \in S'} C^s$ has at most $\max(S)$ neighbours in H and has threshold value $\max(S) + 1$ or $\max(S) + 2$, therefore each vertex of $\bigcup\limits_{s \in S'} C^s$ satisfies the threshold condition. Every vertex of $\bigcup\limits_{s \in S \backslash S'} A^s$ has exactly $\max(S)$ neighbours in H and has threshold value $\max(S) + 1$; every vertex of $\bigcup\limits_{s \in S \backslash S'} B^s$ has exactly $\max(S) + 1$ neighbours in H and has threshold value $\max(S) + 2$. Therefore, every vertex of $\bigcup\limits_{s \in S \backslash S'} C^s$ satisfies the threshold condition. Let u_i be an arbitrary element in U. As $\sum\limits_{s \in S'} s(i) \ge g(i)$, we get $d_H(u_i) = \sum\limits_{s \in S \backslash S'} s(i) = \sum\limits_{s \in S} s(i) - \sum\limits_{s \in S'} s(i) \le \sum\limits_{s \in S} s(i) - g(i) < t(u_i)$.

Towards showing the reverse direction, let H be a harmless set of size at least r in G. It may be noted that $H \cap (U \cup \{x, y\}) = \emptyset$, otherwise one of the vertices in $\{x, y\}$ will fail to satisfy the threshold condition. Observe that any clique C^s can contribute at most $\max(S) + 1$ vertices, otherwise vertices of A^s will fail to satisfy the threshold condition. We now prove the following simple claim.

Claim 3.2.1. *If* $|C^s \cap H| = \max(S) + 1$ *then* $C^s \cap H = A^s$.

Proof. Targeting a contradiction, suppose $|C^s \cap H| = \max(S) + 1$ but $C^s \cap H \neq A^s$. That is, there exists a vertex $a^s \in A^s$ such that $a^s \notin H$. As $|C^s \cap H| = \max(S) + 1$, we have $d_H(a^s) = \max(S) + 1 = t(a^s)$, which is a contradiction. This proves the claim. □

Note that n cliques together contribute at least $r = k' \cdot \max(S) + (n - k') \cdot (\max(S) + 1)$ vertices to H and each clique can contribute at most $\max(S) + 1$ vertices. Therefore, by Pigeonhole principle, there are at least $(n - k')$ cliques C^s for which $|H \cap C^s| = \max(S) + 1$. By the above claim, there are at least $(n - k')$ cliques C^s for which $H \cap C^s = A^s$. We define

$$S' = \{s \in S \mid H \cap C^s \neq A^s\}.$$

Clearly for $s \in S'$, we have $|C^s \cap H| \leq \max(S)$. We construct H' from H as follows:

$$H' = \big(H \setminus \bigcup_{s\in S'} A^s\big) \cup \bigcup_{s \in S'} B^s \setminus \{b_1^s\}.$$

Clearly $|H'| \geq |H|$ and H' is a harmless set. Every vertex of $\bigcup\limits_{s\in S'} C^s$ satisfies the threshold condition because every vertex of $\bigcup\limits_{s\in S'} C^s$ has $\max(S)$ neighbours in H and has threshold value $\max(S) + 1$. For each $u_i \in U$, we see that $d_{H'}(u_i) \leq d_H(u_i) < t(u_i)$. For rest of the vertices, we can easily verify that the threshold conditions are satisfied. This implies that H' is a harmless set of size at least r. So we consider the harmless set to be of the form $H' = \bigcup\limits_{s\in S\setminus S'} A^s \bigcup\limits_{s\in S'} B^s \setminus \{b_1^s\}$. Since each $u_i \in U$ satisfies the threshold condition, we have $d_{H'}(u_i) = \sum\limits_{s\in S\setminus S'} s(i) = \sum\limits_{s\in S} s(i) - \sum\limits_{s\in S'} s(i) < t(u_i) = \sum\limits_{s\in S} s(i) - g(i) + 1$. This implies that $\sum\limits_{s\in S'} s(i) \geq g(i)$ for $1 \leq i \leq k$. Therefore (k, k', S, g) is a yes-instance. □

3.3 XP algorithm parameterized by clique-width

This section presents an XP-time algorithm for Harmless Set parameterized by clique-width.

Theorem 3.3.1. *Given an n-vertex graph G and an irredundant c-expression T of G,* Harmless Set *is solvable in $O(cn^{4c})$ time.*

For each node t in a c-expression T, let G_t be the vertex-labeled graph represented by t. We denote by V_t the vertex set of G_t. For each label i, we denote the set of i-vertices in G_t by V_t^i. For each node t in T, we construct a table $dp_t(\mathtt{r}, \mathtt{s}) \in \{\text{true, false}\}$ where $\mathtt{r} = (\mathtt{r}(1), \dots, \mathtt{r}(c))$ and $\mathtt{s} = (\mathtt{s}(1), \dots, \mathtt{s}(c))$ are c-dimensional vectors; for each index $i \in \{1, 2, \dots, c\}$, $\mathtt{r}(i)$ can take value from the set $\{0, \dots, n\}$ and $\mathtt{s}(i)$ can take value from the set $\{-n+1, \dots, n-1\} \cup \{\infty\}$. We set $dp_t(\mathtt{r}, \mathtt{s}) = \text{true}$ if and only if there exists a set S in V_t such that for all $i \in \{1, 2, \dots, c\}$

- $\mathtt{r}(i) = |S \cap V_t^i|$;
- $\mathtt{s}(i) = \min_{v \in V_t^i} \Big\{ t(v) - |N_{G_t}(v) \cap S| \Big\}$, with $\mathtt{s}(i) = \infty$ if $V_t^i = \emptyset$.

That is, $\mathtt{r}(i)$ denotes the number of the i-vertices in S and $\mathtt{s}(i)$ is the "surplus" at the weakest i-vertex in S.

Let τ be the root of the c-expression T of G. Then G has a harmless set of size h if there exist $\mathtt{r}, \mathtt{s}$ satisfying

1. $dp_\tau(\mathtt{r}, \mathtt{s}) = \text{true}$;
2. $\sum_{i=1}^{c} r(i) = h$
3. $\min\Big\{\mathtt{s}(i)\Big\} \geq 1$.

In the following, we compute all entries $dp_t(\mathtt{r}, \mathtt{s})$ in a bottom-up manner. There are $(n+1)^c \cdot (2n)^c = O(n^{2c})$ possible tuples $(\mathtt{r}, \mathtt{s})$. Thus, to prove Theorem 3.3.1, it is enough to prove that each entry $dp_t(\mathtt{r}, \mathtt{s})$ can be computed in time $O(cn^{2c})$ assuming that the entries for the children of t are already computed.

Lemma 3.3.2. For a leaf node t with label i, $dp_t(\mathtt{r}, \mathtt{s})$ can be computed in $O(c)$ time.

Proof. Observe that $dp_t(\mathtt{r}, \mathtt{s}) = \mathtt{true}$ if and only if $\mathtt{r}(j) = 0$, $\mathtt{s}(j) = \infty$ for all $j \neq i$, and either

- $\mathtt{r}(i) = 0$, $\mathtt{s}(i) = \infty$ or
- $\mathtt{r}(i) = 1$, $\mathtt{s}(i) \geq 1$.

The first case corresponds to $S = \emptyset$, and the second case corresponds to $S = V_t^i$. These conditions can be checked in $O(c)$ time.

Lemma 3.3.3. For a $\oplus$ node t, $dp_t(\mathtt{r}, \mathtt{s})$ can be computed in $O(cn^{2c})$ time.

Proof. Let t_1 and t_2 be the children of t in T. Then $dp_t(\mathtt{r}, \mathtt{s}) = \mathtt{true}$ if and only if there exist $(\mathtt{r}_1, \mathtt{s}_1)$ and $(\mathtt{r}_2, \mathtt{s}_2)$ such that $dp_{t_1}(\mathtt{r}_1, \mathtt{s}_1) = \mathtt{true}$, $dp_{t_2}(\mathtt{r}_2, \mathtt{s}_2) = \mathtt{true}$, $\mathtt{r}(i) = \mathtt{r}_1(i) + \mathtt{r}_2(i)$, and $\mathtt{s}(i) = \min\Big\{\mathtt{s}_1(i), \mathtt{s}_2(i)\Big\}$ for all i. The number of possible pairs for $(\mathtt{r}_1, \mathtt{r}_2)$ is $(n+1)^c$ as $\mathtt{r}_2$ is uniquely determined by $\mathtt{r}_1$. There are at most $(2n)^c$ possible pairs for $(\mathtt{s}_1, \mathtt{s}_2)$ as either $\mathtt{s}_1(i) = \mathtt{s}(i)$ or $\mathtt{s}_2(i) = \mathtt{s}(i)$ for all i. In total, there are $(n+1)^c \cdot (2n)^c = O(n^{2c})$ candidates. Each candidate can be checked in $O(c)$ time. Thus the lemma holds.

Lemma 3.3.4. For a $\eta_{i,j}$ node t, $dp_t(\mathtt{r}, \mathtt{s})$ can be computed in $O(c)$ time.

Proof. Let t' be the child of t in T. Then, $dp_t(\mathtt{r}, \mathtt{s}) = \mathtt{true}$ if and only if $dp_{t'}(\mathtt{r}, \mathtt{s}') = \mathtt{true}$ for some $\mathtt{s}'$ with the following conditions:

- $\mathtt{s}(h) = \mathtt{s}'(h)$ hold for all $h \in \{1, 2, \ldots, c\} \setminus \{i, j\}$;
- $\mathtt{s}(i) = \mathtt{s}'(i) - \mathtt{r}(j)$ and $\mathtt{s}(j) = \mathtt{s}'(j) - \mathtt{r}(i)$.

We now explain the condition for $s(i)$. Recall that T is irredundant. That is, the graph $G_{t'}$ does not have any edge between the i-vertices and the j-vertices. In G_t, an i-vertex has exactly $\mathtt{r}(j)$ more neighbours in S and similarly a j-vertex has exactly $\mathtt{r}(i)$ more neighbours in S. Thus we have $\mathtt{s}(i) = \mathtt{s}'(i) - \mathtt{r}(j)$ and $\mathtt{s}(j) = \mathtt{s}'(j) - \mathtt{r}(i)$. The lemma holds as there is only one candidate for each $\mathtt{s}'(i)$ and $\mathtt{s}'(j)$.

Lemma 3.3.5. For a $\rho_{i\to j}$ node t, $dp_t(\mathtt{r}, \mathtt{s})$ can be computed in $O(cn^2)$ time.

Proof. Let t' be the child of t in T. Then, $dp_t(\mathtt{r}, \mathtt{s}) = \mathtt{true}$ if and only if there exist $\mathtt{r}', \mathtt{s}'$ such that $dp_{t'}(\mathtt{r}', \mathtt{s}') = \mathtt{true}$, where :

- $\mathtt{r}(i) = 0$, $\mathtt{r}(j) = \mathtt{r}'(i) + \mathtt{r}'(j)$, and $\mathtt{r}(h) = \mathtt{r}'(h)$ for all $h \in \{1, 2, \ldots, c\} \setminus \{i, j\}$;

- $\mathtt{s}(i) = \infty$, $\mathtt{s}(j) = \min\{\mathtt{s}'(i), \mathtt{s}'(j)\}$, and $\mathtt{s}(h) = \mathtt{s}'(h)$ for all $h \in \{1, 2, \ldots, c\} \setminus \{i, j\}$;

The number of possible pairs for $(\mathtt{r}'(i), \mathtt{r}'(j))$ is $O(n)$ as $\mathtt{r}'(j)$ is uniquely determined by $\mathtt{r}'(i)$. There are at most $O(n)$ possible pairs for $(\mathtt{s}'(i), \mathtt{s}'(j))$. In total, there are $O(n^2)$ candidates. Each candidate can be checked in $O(c)$ time, thus the lemma holds.

3.4 Closing Remarks and Future directions

Our results close a wide gap in the understanding of the complexity landscape of Harmless Set parameterized by structural parameters. We have shown that Harmless Set is W[1]-hard parameterized by a wide range of fairly restrictive structural parameters such as the feedback vertex set number, pathwidth, treedepth, and cluster vertex deletion number of the input graph. On the positive side, we have given FPT algorithms when parameterized by any of the following parameters: vertex integrity, neighbourhood diversity and twin cover. To give an upper bound on the complexity, we give an XP-algorithm parameterized by clique-width. The figure 3.9 provides a summary of results.

It is interesting to figure out the parameterized complexity of Harmless Set with respect to the parameters such as vertex deletion to disjoint paths and modular width. Also it will be inetresting to know the exact class of these problems in W-hierarchy. For Majority Harmless Set, it will be interesting to see if the parameters such as treedepth, feedback vertex set and cluster vertex deletion set allow FPT algorithms or the problem still remains intractable.

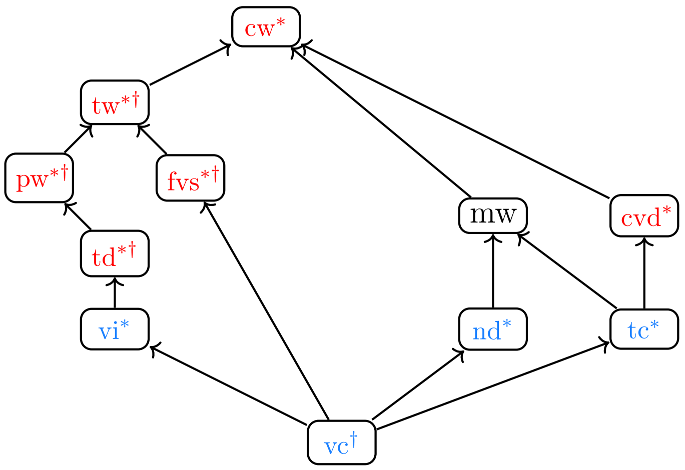


Figure 3.9: This is an overview of the parameterized complexity landscape for the Harmless Set problem with general thresholds. The stars highlight parameters that are covered in this study. The † symbol highlight parameters that are simultaneously and independently proved by Drange, Muzi and Reidl [38]. The problem is FPT parameterized by blue colored parameters and W[1]-hard when parameterized by red colored parameters. The problem can be solved in XP-time when parameterized by clique-width; hence the problem is in XP when parameterized by each of the following parameters: tw, pw, fvs, td and cvd. The problem remains unsettled when parameterized by mw.

Chapter 4

Defensive Alliance in Graphs

A set S of vertices of a graph is a *defensive alliance* if, for each element of S, the majority of its neighbours are in S. In this chapter, we study the parameterized complexity of DEFENSIVE ALLIANCE, where the aim is to find a minimum size defensive alliance. Our main results are the following: (1) DEFENSIVE ALLIANCE has been studied extensively during the last twenty years, but the question whether it is FPT when parameterized by feedback vertex set has still remained open. We prove that the problem is W[1]-hard parameterized by a wide range of fairly restrictive structural parameters such as the feedback vertex set number, treewidth, pathwidth, and treedepth of the input graph; (2) the problem parameterized by the vertex cover number of the input graph does not admit a polynomial compression unless coNP $\subseteq$ NP/poly, (3) it does not admit $2^{o(n)}$ algorithm under ETH, and (4) DEFENSIVE ALLIANCE on circle graphs is NP-complete. This chapter is based on the papers [61, 72].

4.1 Hardness Results of Defensive Alliance

In this section we show that DEFENSIVE ALLIANCE is W[1]-hard when parameterized by the size of a vertex deletion set into collection of stars, i.e., the size of a subset D of the vertices of the graph such that every component in the graph, after removing D, is a star. To show W[1]-hardness of DEFENSIVE ALLIANCE, we reduce from the following problem:

Multidimensional Relaxed Subset Sum (MRSS)
Input: An integer k, a set $S = \{s_1, \ldots, s_n\}$ of vectors with $s_i \in \mathbb{N}^k$ for every i with $1 \leq i \leq n$, a target vector $t \in \mathbb{N}^k$ and an integer k'.
Parameter: $k + k'$
Question: Is there a subset $S' \subseteq S$ with $|S'| \leq k'$ such that $\sum\limits_{s \in S'} s \geq t$?

It is known that MRSS is W[1]-hard when parameterized by the combined parameter $k+k'$, even if all integers in the input are given in unary [74]. To prove W[1]-hardness of Defensive Alliance, we reduce MRSS to Defensive Alliance.

Construction: Let $I = (k, k', S, t)$ be an instance of MRSS. From this we construct an instance $I' = (G, r)$ of Defensive Alliance the following way. See Figure 4.1 for an illustration. Before we formally define our reduction, we briefly describe the intuition. We introduce three types of vertices: necessary vertices, forbidden vertices and normal vertices. We often indicate necessary vertices by means of a triangular node shape, and forbidden vertices by means of a square node shape, and normal vertices by means of a circular node shape. We want to make sure that *necessary vertices* are always inside every solution and *forbidden vertices* are always outside every solution; and a *normal vertex* could be inside or outside the solution. Note that it is easy to force a forbidden vertex outside every solution by simply increasing its degree in the graph. Specifically, if we are looking for a defensive alliance of size at most r then vertices of degree more than $2r$ are always outside every solution. The challenging part is to force inclusion of all necessary vertices in every solution. Our strategy is to first make sure that if even one necessary vertex is included in a solution then that will force inclusion of all necessary vertices in the solution. In order to do this, we have introduced a “chain" like gadget consisting of some necessary vertices. Next, in order to force inclusion of at least one necessary vertex in every solution, we consider protection of normal vertices. For all normal vertices, we add some forbidden and necessary neighbours such that their protection will require at least one necessary vertex.

Let $s = (s(1), s(2), \ldots, s(k)) \in S$ and let $\max(s)$ denote the value of the largest coordinate of s. Set $N = \sum\limits_{s \in S}(2\max(s) + 2)$. The vertex set of the constructed graph G is defined as follows:

1. We introduce a set of k necessary vertices $U = \{u_1, u_2, \ldots, u_k\}$. For every $u_i \in U$,

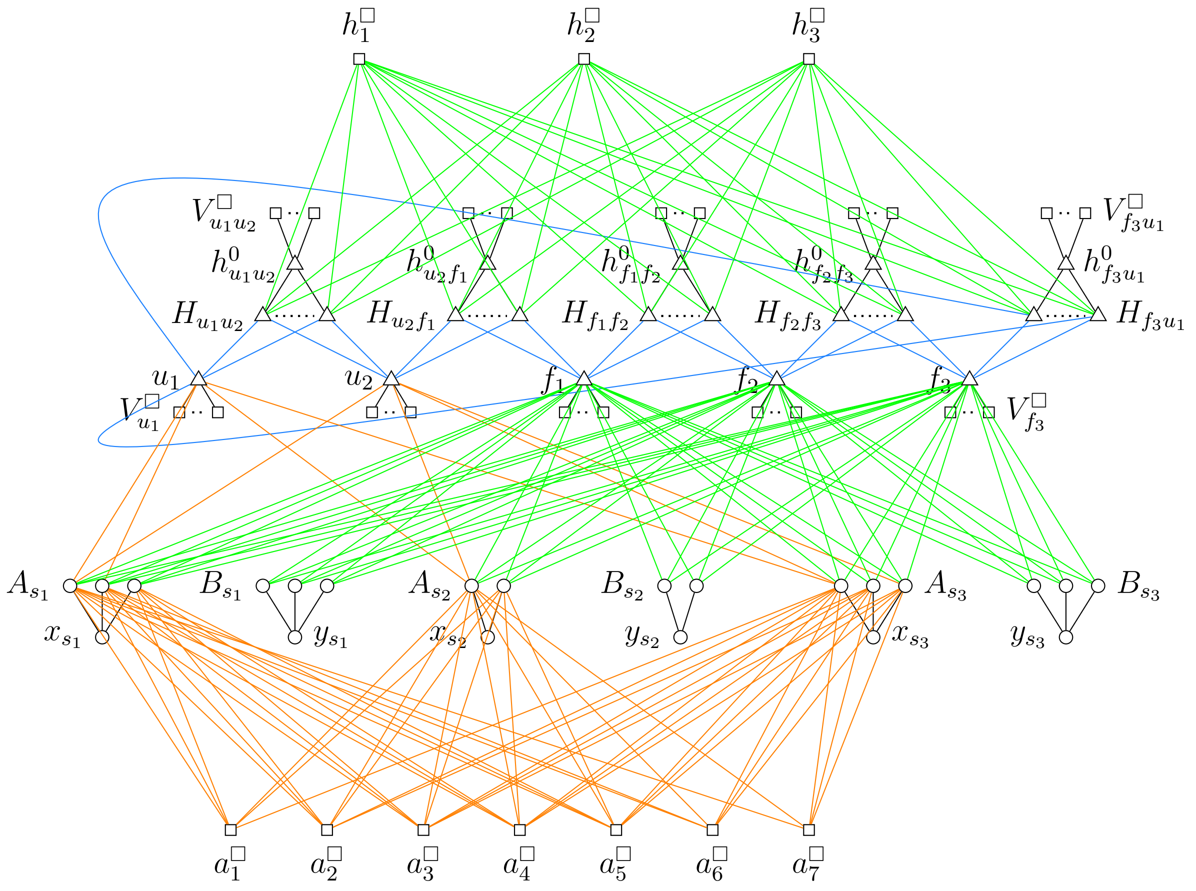


Figure 4.1: The graph G in the proof of Theorem 4.1.4 constructed for MRSS instance $S = \{(2,1),(1,1),(1,2)\}$, $t = (3,3)$, $k = 2$ and $k' = 2$. The vertices t and t', and their adjacency are not shown. Every square vertex is adjacent to $2r+2$ vertices which are also not shown in the diagram.

create a set $V_{u_i}^{\square}$ of $2N - \sum_{s\in S} s(i) + 2t(i)$ forbidden vertices.

2. We introduce a set of three necessary vertices $F = \{f_1, f_2, f_3\}$. For every $f \in F$, create a set $V_f^{\square}$ of $2N$ forbidden vertices.

3. Consider the vertices of $U \cup F$ in the following order: $u_1, \ldots, u_k, f_1, f_2, f_3, u_1$. Let $P = \{(u_1,u_2), \ldots, (u_{k-1},u_k), (u_k,f_1), (f_1,f_2), (f_2,f_3), (f_3,u_1)\}$ be the set of pairs of consecutive vertices. For each pair $(x,y) \in P$, create a set of N necessary vertices $H_{xy} = \{h_{xy}^1, \ldots, h_{xy}^N\}$, a set $V_{xy}^{\square}$ of N forbidden vertices, and a special necessary vertex h_{xy}^0.

4. We introduce a set of three forbidden vertices $H^{\square} = \{h_1^{\square}, h_2^{\square}, h_3^{\square}\}$

5. For each vector $s \in S$, we introduce two stars. The first star has internal node x_s and $\max(s)+1$ leaves $A_s = \{a_1^s, \ldots, a_{\max(s)+1}^s\}$; the second star has internal node y_s and $\max(s)+1$ leaves $B_s = \{b_1^s, \ldots, b_{\max(s)+1}^s\}$.

6. We introduce a set $V_a^{\square} = \{a_1^{\square}, \ldots, a_{k+5}^{\square}\}$ of $k+5$ forbidden vertices.

We now create the edge set of G.

1. For every $u_i \in U$, make u_i adjacent to every vertex of $V_{u_i}^{\square}$.

2. For every $f \in F$, make f adjacent to every vertex of $V_f^{\square}$.

3. For each pair $(x,y) \in P$, make x and y adjacent to every vertex of H_{xy}; make h_{xy}^0 adjacent to every vertex of $H_{xy} \cup V_{xy}^{\square}$.

4. For each pair $(x,y) \in P$, make every vertex of H_{xy} adjacent to every vertex of $H^{\square}$.

5. For each $u_i \in U$ and for each $s \in S$, we make u_i adjacent to exactly $s(i)$ many vertices of A_s in an arbitrary manner. Make every vertex of F adjacent to every vertex of $\bigcup\limits_{s\in S} A_s \cup B_s$.

6. For each $s \in S$, we make each $a^s \in A_s$ adjacent to exactly $|N_U(a^s)| + 5$ many vertices of $V_a^{\square}$ arbitrarily.

We define the set of necessary vertices $V_{\triangle} = U \cup F \cup \bigcup\limits_{(x,y)\in P} H_{xy} \cup \{h_{xy}^0\}$, the set of forbidden vertices $V_{\square} = V_a^{\square} \cup H^{\square} \cup \bigcup\limits_{(x,y)\in P} V_{xy}^{\square} \cup \bigcup\limits_{u\in U} V_u^{\square} \cup \bigcup\limits_{f\in F} V_f^{\square}$ and $r = (k+3)N + 2k + 6 + \sum\limits_{s\in S}(\max(s)+1) + k'$. For every $x \in V_{\square}$, create a set V_x of $2r+2$ vertices, and make x adjacent to every vertex of V_x. We also add two vertices t and t'. The newly added vertices t, t' and the vertices in V_x are not shown in Figure 4.1. Make t adjacent to every vertex of V_x for $x \in \bigcup\limits_{(x,y)\in P} V_{xy}^{\square} \cup \bigcup\limits_{u\in U} V_u^{\square} \cup \bigcup\limits_{f\in F} V_f^{\square}$ and make t' adjacent to every vertex of V_x for $x \in V_a^{\square} \cup H^{\square}$. This completes the construction of I'.

The proof of Lemma 4.1.3, explains how inclusion of one vertex from $U \cup F$ ensures inclusion of all necessary vertices $V_{\triangle}$ in the solution. In order to make sure that at least one necessary vertex from $V_{\triangle}$ is included in every solution, we consider protection of normal vertices. For each normal vertex, we add some forbidden and necessary neighbours such that its protection requires at least one necessary vertex. As all the vertices in $V_\triangle$ are inside every solution, the idea in the remaining part of the reduction is hidden in the protection of vertices in U. Note that a normal vertex may or may not be included in the solution. The vertices of two star graphs that we have introduced corresponding to each vector in S are normal vertices. Note that the vertices in U have neighbours only in star graphs containing A_s's. Now, the construction makes sure that if a defensive alliance uses a vertex from A_s then it must pay a cost of one extra vertex x_s by forcing it to be inside the solution which is not true for B_s. As this is a minimization problem, every defensive alliance uses either A_s for protection of vertices in U and pay a cost of one extra vertex or uses B_s. At the end, we argue that if the defensive alliance includes A_s then we add the corresponding vector s to the solution of MRSS, otherwise not.

Lemma 4.1.1. *The size of a vertex deletion set of G into collection of stars, is $3k + 16$. Moreover, G is a bipartite graph.*

Proof. Observe that if we remove the set $U \cup F \cup H^\square \cup V_a^\square \cup \{h_{xy}^0 \mid (x,y) \in P\} \bigcup \{t, t'\}$ of $3k + 16$ vertices from G then we are left with only star graphs. Moreover, G is a bipartite graph with bipartition

$$V_1 = H^\square \cup V_a^\square \cup U \cup F \cup \{t'\} \bigcup_{x \in V_\square \setminus (H^\square \cup V_a^\square)} V_x \cup \bigcup_{(x,y)\in P} \{h_{xy}^0\} \cup \bigcup_{s\in S} \{x_s, y_s\}$$

and

$$V_2 = \{t\} \cup \bigcup_{(x,y)\in P} (V_{xy}^\square \cup H_{xy}) \cup \bigcup_{u\in U} V_u^\square \cup \bigcup_{f\in F} V_f^\square \cup \bigcup_{x\in H^\square \cup V_a^\square} V_x \cup \bigcup_{s\in S} (A_s \cup B_s).$$

□

Lemma 4.1.2. *If (k, k', S, t) is a positive instance of* MRSS *then G has a defensive alliance of size at most r.*

Proof. Let $S' \subseteq S$ be such that $|S'| \leq k'$ and $\sum_{s\in S'} s \geq t$. We claim that the set

$$R = U \cup F \cup \bigcup_{(x,y)\in P} H_{xy} \cup \{h_{xy}^0\} \cup \bigcup_{s\in S'} A_s \cup \{x_s\} \cup \bigcup_{s\in S\setminus S'} B_s$$

is a defensive alliance in G such that $|R| \leq r$. Let x be an arbitrary element of R.
Case 1: If $x = u_i \in U$, then the neighbours of u_i in R are the elements in $H_{u_{i-1}u_i} \cup H_{u_i u_{i+1}}$ and $s(i)$ elements of A_s if $s \in S'$. Thus

$$d_R(u_i) = \sum_{s \in S'} s(i) + 2N.$$

The neighbours of u_i in R^c are the elements of $V^{\square}_{u_i}$ and $s(i)$ elements of A_s if $s \in S \setminus S'$. Thus

$$d_{R^c}(u_i) = \sum_{s \in S \setminus S'} s(i) + |V^{\square}_{u_i}| = \sum_{s \in S \setminus S'} s(i) + \underbrace{2N - \sum_{s \in S} s(i) + 2t(i)}_{\text{size of } V^{\square}_{u_i}}$$

Note that

$$\begin{aligned}
d_{R^c}(u_i) &= 2N + \sum_{s \in S \setminus S'} s(i) - \sum_{s \in S} s(i) + 2t(i) \\
&= 2N - \Big(\sum_{s \in S} s(i) - \sum_{s \in S \setminus S'} s(i) \Big) + 2t(i) \\
&= 2N - \sum_{s \in S'} s(i) + 2t(i) \\
&= 2N + \sum_{s \in S'} s(i) + 2 \underbrace{\Big(t(i) - \sum_{s \in S'} s(i) \Big)}_{\leq 0} \\
&\leq 2N + \sum_{s \in S'} s(i) \\
&= d_R(u_i)
\end{aligned}$$

Therefore, we have $d_R(u_i) + 1 \geq d_{R^c}(u_i)$, and hence u_i is protected.

Case 2: If $x = a^s \in A_s$, then $d_R(a^s) = |N_U(a^s)| + |\{f_1, f_2, f_3, x_s\}| = |N_U(a^s)| + 4$ and $d_{R^c}(a^s) = |N_U(a^s)| + 5$. Therefore, we get $d_R(a^s) + 1 \geq d_{R^c}(a^s)$.

Case 3: If $x = f_1 \in F$, then $N_R(f_1) = \bigcup\limits_{s \in S'} A_s \cup \bigcup\limits_{s \in S \setminus S'} B_s \cup H_{u_2 f_1} \cup H_{f_1 f_2}$ and $N_{R^c}(f_1) = \bigcup\limits_{s \in S'} B_s \cup \bigcup\limits_{s \in S \setminus S'} A_s \cup V^{\square}_{f_1}$. As $|A_s| = |B_s|$, $|H_{u_2 f_1}| = |H_{f_1 f_2}| = N$ and $|V^{\square}_{f_1}| = 2N + 1$, we have $d_R(f_1) + 1 \geq d_{R^c}(f_1)$. We can similarly check that $\{f_2, f_2\}$ are also protected.

For the rest of the vertices in R, it is easy to see that $d_R(x) + 1 \geq d_{R^c}(x)$. Therefore, (G, r)

is a yes-instance of DEFENSIVE ALLIANCE. □

Lemma 4.1.3. *If G has a defensive alliance of size at most r then (k, k', S, t) is a positive instance of* MRSS.

Proof. Let R be a defensive alliance in G of size at most r. It is easy to see that $(V_\square \cup \{t, t'\}) \cap R = \emptyset$ as any defensive alliance of size at most r cannot contain vertices of degree greater than $2r$. This also shows that $\bigcup\limits_{x \in V_\square} V_x \cap R = \emptyset$. Now we show that $V_\triangle = U \cup F \bigcup\limits_{(x,y)\in P} H_{xy} \cup \{h^0_{xy}\} \subseteq R$. We claim that if $V_\triangle \cap R \neq \emptyset$ then $V_\triangle \subseteq R$.

Case 1: Suppose R contains u_1 from U. We observe that if some $h_{u_1u_2} \in H_{u_1u_2}$ is not in R then $h^0_{u_1u_2}$ is also not in R. This implies that no vertex from $H_{u_1u_2}$ is in R. Since the elements of $H_{u_1u_2} \cup V^\square_{u_1}$ are not in R, we get $d_{R^c}(u_1) \geq |H_{u_1u_2}| + |V^\square_{u_1}| = N + \underbrace{2N - \sum\limits_{s\in S} s(1) + 2t(1)}_{\text{size of } V^\square_{u_1}}$.

On the other hand the elements of $H_{f_3u_1}$ and $s(1)$ nodes from A_s for each $s \in S$ could be in R. Thus $d_R(u_1) \leq N + \sum\limits_{s\in S} s(1)$. This implies that u_1 is not protected in R which is a contradiction as $u_1 \in R$. This implies that $H_{u_1u_2} \cup \{h^0_{u_1u_2}\} \subseteq R$. Applying the same argument for $h_{f_3,u_1} \in H_{f_3u_1}$, we see that $H_{f_3u_1} \cup \{h^0_{f_3u_1}\} \subseteq R$. Clearly, this shows that f_3 and u_2 are in R for protection of vertices in $H_{u_1u_2} \cup H_{f_3u_1}$. Applying the same argument for u_2 and f_3, we get $H_{u_2u_3} \cup \{h^0_{u_2u_3}\} \subseteq R$ and $H_{f_3u_1} \cup \{h^0_{f_3u_1}\} \subseteq R$, respectively. Repeatedly applying the above argument, we get $V_\triangle \subseteq R$. Observe that this argument can be easily extended to all the vertices of U and F. Therefore, we see that if $(U \cup F) \cap R \neq \emptyset$, then $V_\triangle \subseteq R$.

Case 2: Suppose R contains h_{xy} from H_{xy} for some $(x, y) \in P$. Clearly, this implies R contains both x and y from $U \cup F$. Using Case 1, we get $V_\triangle \subseteq R$.

Case 3: Suppose R contains h^0_{xy} for some $(x, y) \in P$. Clearly, this implies that $H_{xy} \subseteq R$. Using Case 2, we get $V_\triangle \subseteq R$.

Therefore we proved that if $V_\triangle \cap R \neq \emptyset$ then $V_\triangle \subseteq R$. Next we claim that if R is non-empty then R contains the set $V_\triangle$. Since R is non-empty, we see that R must contain a vertex from graph G. We consider the following cases:

Case 1: Suppose R contains a^s from A_s for some $s \in S$. Then we know that $d_{R^c}(a^s) \geq$

$|N_U(a^s)| + 5$. We see that a^s is protected if and only if $F \cap R \neq \emptyset$. This implies that $V_\triangle \cap R \neq \emptyset$ which implies $V_\triangle \subseteq R$.

Case 2: Suppose R contains x_s for some $s \in S$. We know that x_s has at least two neighbours in A_s as $\max(s) + 1 \geq 2$. This implies that x_s is protected if and only if at least one vertex $a^s \in A_s$ is in R. Now, Case 1 implies that $V_\triangle \subseteq R$.

Case 3: Suppose R contains $b^s \in B_s$ for some $s \in S$. Then we know that $N(b^s) = \{y_s\} \cup F$. Clearly, the protection of b^s requires at least one vertex from F. This implies that $F \cap R \neq \emptyset$. Therefore, we have $V_\triangle \cap R \neq \emptyset$ and hence $V_\triangle \subseteq R$.

Case 4: Suppose R contains y_s for some $s \in S$. We know that y_s has at least two neighbours in B_s as $\max(s) + 1 \geq 2$. This implies that y_s is protected if and only if at least one vertex $b^s \in B_s$ is in R. Now, Case 3 implies that $V_\triangle \subseteq R$.

This shows if R is non-empty then $V_\triangle \subseteq R$. We know $V_\triangle$ contains exactly $(k+3)N + 2k + 6$ many vertices; thus besides the vertices of $V_\triangle$, there are at most $\sum\limits_{s \in S}(\max(s) + 1) + k'$ many vertices in R. Since $f_1 \in R$ and $d_{V_\triangle}(f_1) = d_{V_\square}(f_1) = 2N$, it must have at least $\sum\limits_{s \in S}(\max(s)+1)$ many neighbours in R from the set $\bigcup\limits_{s \in S} A_s \cup B_s$. We also observe that if a vertex a^s from the set A_s is in the solution then x_s also lie in the solution for the protection of a^s. This shows that at most k' many sets of the form A_s contribute to the solution as otherwise the size of solution exceeds r. Therefore, any arbitrary defensive alliance R of size at most r can be transformed to another defensive alliance R' of size at most r as follows:

$$R' = V_\triangle \bigcup_{x_s \in R} A_s \cup \{x_s\} \bigcup_{x_s \in V(G) \setminus R} B_s.$$

We define a subset $S' = \Big\{ s \in S \mid x_s \in R' \Big\}$. Clearly, $|S'| \leq k'$. We claim that $\sum\limits_{s \in S'} s(i) \geq t(i)$ for all $1 \leq i \leq k$. Assume for the sake of contradiction that $\sum\limits_{s \in S'} s(i) < t(i)$ for some $i \in \{1, 2, \ldots, k\}$. Note that

$$d_{R'}(u_i) = \sum_{s \in S'} s(i) + 2N$$

and

$$d_{R'^c}(u_i) = \sum_{s\in S\setminus S'} s(i) + |V^{\square}_{u_i}| = \sum_{s\in S\setminus S'} s(i) + 2N - \sum_{s\in S} s(i) + 2t(i)$$

Then, we have

$$\begin{aligned} d_{R'^c}(u_i) &= 2N - \Big(\sum_{s\in S} s(i) - \sum_{s\in S\setminus S'} s(i)\Big) + 2t(i) \\ &= 2N - \sum_{s\in S'} s(i) + 2t(i) \\ &= 2N + \sum_{s\in S'} s(i) + 2\ \underbrace{\Big(t(i) - \sum_{s\in S'} s(i)\Big)}_{>0 \text{ by assumption}} \\ &> 2N + \sum_{s\in S'} s(i) = d_{R'}(u_i) \end{aligned}$$

We also know that $u_i \in R'$, which is a contradiction to the fact that R' is a defensive alliance. Therefore, $\sum\limits_{s\in S'} s(i) \geq t(i)$ for all $1 \leq i \leq k$. This shows that $I = (k, k', S, t)$ is a yes-instance. □

We now prove our main theorem in this section:

Theorem 4.1.4. Defensive Alliance is W[1]-hard when parameterized by the size of a vertex deletion set into collection of stars, even when restricted to bipartite graphs.

Proof. Given an instance (k, k', S, t) of RBDS, we use the above construction to create an instance (G, r) of Defensive Alliance parameterized by the size of a vertex deletion set into collection of stars. Lemma 4.1.2 and Lemma 4.1.3 show the correctness of our reduction, while Lemma 4.1.1 provides the bound on the size of a vertex deletion set into collection of stars, showing that our new parameter is linearly bounded by the original parameter k. This completes the proof. □

Clearly stars are trivially acyclic. Moreover, it is easy to verify that stars have pathwidth [93] and treedepth [110] at most two, which implies:

Theorem 4.1.5. The Defensive Alliance problem is W[1]-hard when parameterized by any of the following parameters:

- the feedback vertex set number,
- the treewidth and clique width of the input graph,
- the pathwidth and treedepth of the input graph,

even when restricted to bipartite graphs.

4.2 No Polynomial Kernel Parameterized by Vertex Cover Number

A set $C \subseteq V$ is a vertex cover of $G = (V, E)$ if each edge $e \in E$ has at least one endpoint in X. The minimum size of a vertex cover in G is the *vertex cover number* of G, denoted by $vc(G)$. Parameterized by vertex cover number vc, Defensive Alliance is FPT [92] and in this section we prove the following kernelization hardness of Defensive Alliance.

Theorem 4.2.1. Defensive Alliance parameterized by the vertex cover number of the input graph does not admit a polynomial compression unless coNP $\subseteq$ NP/poly.

To prove Theorem 4.2.1, we give a polynomial parameter transformation (PPT) from the well-known Red Blue Dominating Set problem (RBDS) to Defensive Alliance parameterized by vertex cover number. Recall that in RBDS we are given a bipartite graph $G = (T \cup S, E)$ and an integer k, and we are asked whether there exists a vertex set $X \subseteq S$ of size at most k such that every vertex in T has at least one neighbour in X. We also refer to the vertices of T as *terminals* and to the vertices of S as *sources* or *nonterminals*. The following theorem is known:

Theorem 4.2.2. [55] RBDS parameterized by $|T|$ does not admit a polynomial compression unless coNP $\subseteq$ NP/poly.

4.2.1 Proof of Theorem 4.2.1

By Theorem 6.2.2, RBDS parameterized by $|T|$ does not admit a polynomial compression unless coNP $\subseteq$ NP/poly. To prove Theorem 4.2.1, we give a PPT from RBDS parameterized by $|T|$ to Defensive Alliance parameterized by the vertex cover number. Given an instance $(G = (T \cup S, E), k)$ of RBDS, we construct an instance (G', k') of Defensive Alliance as follows. Take three distinct copies T_0, T_1, T_2 of T, and let t_i be the copy of $t \in T$ in T_i. Similarly, take two distinct copies S_0, S_1 of S, and let s_i be the copy of $s \in S$ in S_i. Now for every vertex $t \in T_1 \cup T_2$, we introduce a set $V_t = \{t_1, \ldots, t_\ell\}$ of vertices and make them adjacent to t where the number ℓ is defined later in the proof. Moreover, create three vertices a, b and c ans make them adjacent to every vertex of $\bigcup\limits_{t \in T_1 \cup T_2} V_t$. We also make a and b adjacent to every vertex of T_0; and make a adjacent to every vertex of S_1. If $(t, s) \in E(G)$ then we add the edges $(t_0, s_0), (t_0, s_1), (t_1, s_1)$ and (t_2, s_0) in $E(G')$. Finally, we add a vertex x^* which is adjacent to every vertex of S_1 and also adjacent to exactly $|S|$ many arbitrary vertices from V_t for some $t \in T_1 \cup T_2$. We set $k' = |T| + |S| + k + 1$ and $l = 2k' + 1$. Clearly (G', k') can be computed in polynomial time. We observe that $C = T_0 \cup T_1 \cup T_2 \cup \{a, b, c, x^*\}$ is a vertex cover of G'. Therefore the vertex cover size of G' is bounded by $3|T| + 4$. See Figure 4.2 for an illustration. We now claim that (G, k) is a yes-instance of RBDS if and only if (G', k') is a yes-instance of Defensive Alliance.

Suppose there exists a vertex set $X \subseteq S$ of size at most k in G such that every vertex in T has at least one neighbour in X. We claim that the set $R = S_1 \cup \big\{s_0 \in S_0 \mid s \in X\big\} \cup T_0 \cup \{x^*\}$ is a defensive alliance in graph G'. Let x be an arbitrary element of R. We prove that x is protected in R.

Case 1: Suppose $x \in S_1$. Note that $N_R(x) = N_{T_0}(x) \cup \{x^*\}$. Thus, including itself, it has $d_G(x) + 2$ defenders in G'. The attackers of x consist of elements of $N_{T_1}(x)$ and element a. Hence x has $d_G(x) + 1$ attackers. This shows that x has at least as many defenders as attackers; hence x is protected.

Case 2: Suppose $x \in \big\{s_0 \in S_0 \mid s \in X\big\}$. Note that $N_R(x) = N_{T_0}(x)$. Thus, including itself, it has $d_G(x) + 1$ defenders in G'. The attackers of x consist of elements of $N_{T_2}(x)$. Hence x has $d_G(x)$ attackers in G'. This shows that x is protected.

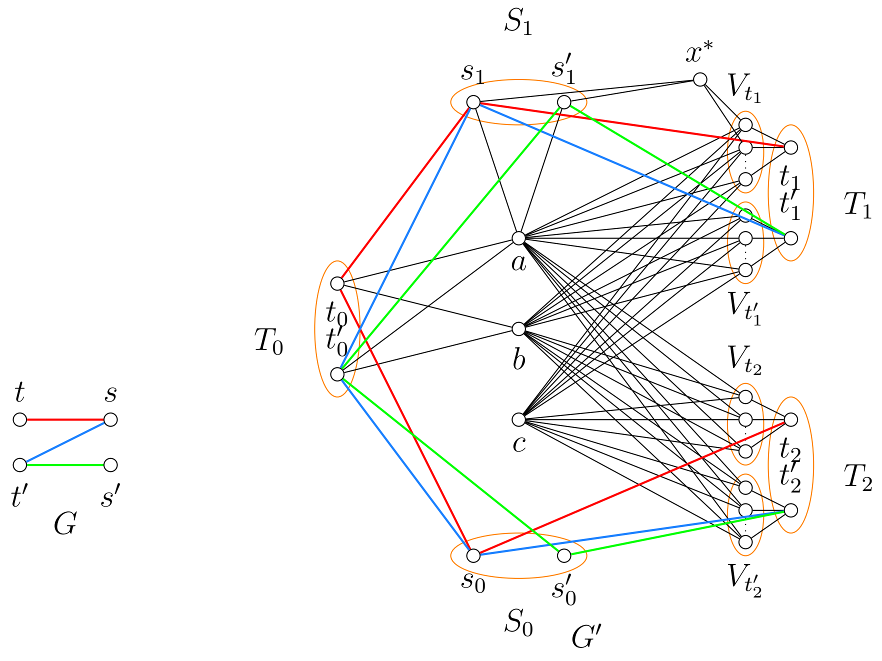


Figure 4.2: PPT from RBDS to Defensive Alliance, where graph G is show on the left and G' is shown on the right.

Case 3: Suppose $x \in T_0$. Clearly, including itself, x has $2d_G(x) + 3$ neighbours in G'. Thus it requires at least $d_G(x) + 2$ many defenders in G'. Note that, including itself, x has $d_G(x) + 1$ neighbours in $S_1 \subseteq R$. Therefore, it requires at least one neighbour from the set $\Big\{s_0 \in S_0 \mid s \in X\Big\}$ inside the solution and this is true because (G, k) is a yes-instance of RBDS.

Case 4: Suppose $x = x^*$. It has the same number of defenders and attackers in G'. This shows that x is protected.

Conversely, suppose there exists a defensive alliance R of size at most k' in G'. Clearly, no vertex from the set $Q = T_1 \cup T_2 \cup \{a, b, c\} \bigcup\limits_{t \in T_1 \cup T_2} V_t$ can be part of R as $|R| \leq k'$ and $d_{G'}(v) > 2k'$ for all $v \in Q$. Since R is non-empty, it must contain a vertex from one of the sets $\{x^*\}, S_1, T_0$ or S_0.

Case 1: Suppose $x^* \in R$. Since x^* has $|S|$ many neighbours in Q, it implies that all the neighbours of x^* in S_1 must be inside the solution for protection of x^*. This implies that $S_1 \subseteq R$. Let v be an arbitrary vertex in S_1. Note that v has $d_G(v)$ neighbours in T_0, and it has $d_G(v) + 1$ neighbours in Q. For protection of v all the neighbours of v in T_0 must be part of the solution. This implies that $T_0 \subseteq R$ as all the vertices in S_1 must be protected. Note that till now we have added $|S| + |T| + 1$ many vertices in the solution. Therefore, we can add at most k vertices to the solution from the set S_0 as otherwise the solution size will exceed k'. Suppose we add a set $X \subseteq S_0$ of size at most k to the solution. Consider the protection of vertices in T_0. If v is a vertex of T_0, then it has $d_G(v)$ neighbours in S_0 and similarly $d_G(v)$ neighbours in S_1. Excluding itself, v has $2d_G(v) + 2$ neighbours in G'. Thus it requires at least $d_G(v)+1$ many neighbours inside the solution. We know that $d_G(v)$ neighbours are inside the solution due to the fact that $S_1 \subseteq R$. Therefore, it requires at least one neighbour from S_0 inside the solution. Since there exists a set $X \subseteq S$ of size at most k such that all the vertices in T_0 are protected, it shows that all vertices in T_0 have at least one neighbour in X.

Case 2: Suppose R contains a vertex v from the set S_1. In this case, the protection of v requires x^* to be inside the solution and then the same argument as in Case 1 will lead to the proof.

Case 3: Suppose R contains a vertex v from the set T_0. Excluding itself, v has $2d_G(v) + 2$ neighbours in G'. Thus it requires at least $d_G(v) + 1$ many neighbours from $S_0 \cup S_1$ inside the solution. This implies that at least one neighbour from the set S_1 must be inside the solution. Now the same argument as in Case 2 will lead to the proof.

Case 4: Suppose R contains a vertex v from the set S_0. Clearly, it has $d_G(v)$ neighbours in $T_2 \subseteq Q$ and $d_G(v)$ neighbours in T_0. Since the vertices in the set Q cannot be part of the solution, the protection of v will imply that all the neighbours of v in T_0 are part of the solution. In other words, there exists a vertex in T_0 which is inside the solution. Now the same argument as in Case 3 will lead to the proof.

This proves that (G, k) is a yes-instance of RBDS. $\square$

4.3 Defensive Alliance has no Subexponential Algorithm

In this section, we prove lower bound based on ETH for the time needed to solve the Defensive Alliance problem. In order to prove that a too fast algorithm for Defensive Alliance contradicts ETH, we give a reduction from Vertex Cover in graphs of maximum degree 3 and argue that a too fast algorithm for Defensive Alliance would solve Vertex Cover in graphs of maximum degree 3 in time $2^{o(n)}$. Using Theorem 1.1 in [85], we get that there is an algorithm which finds a minimum size vertex cover in an arbitrary graph on n vertices in time sub-exponential in n if and only if there is an algorithm which finds a minimum size vertex cover for graphs that have maximum degree at most 3 in time sub-exponential in n. Assuming ETH, there is no algorithm which finds a minimum size vertex cover in an arbitrary graph on n vertices in time sub-exponential in n [100]. This implies, assuming ETH, there is no algorithm which finds a minimum size vertex cover for graphs that have maximum degree at most 3 in time sub-exponential in n.

Theorem 4.3.1. Unless ETH fails, Defensive Alliance does not admit a $2^{o(n)}$ algorithm where n is the number of vertices of the input graph.

Proof. We give a linear reduction from Vertex Cover in graphs of maximum degree 3 to Defensive Alliance, that is, a polynomial-time algorithm that takes an instance (G, k) of Vertex Cover, where G has n vertices and $m = O(n)$ edges, and outputs an equivalent instance of Defensive Alliance whose size is bounded by $O(n)$. We assume that the input graph contains at least two edges otherwise the problem will be polynomial time solvable. We construct an equivalent instance (G', k') of Defensive Alliance the following way. See Figure 4.3 for an illustration.

1. We introduce the vertex sets X and Y into G', where $X = V(G) = \{v_1, \ldots, v_n\}$ and $Y = E(G) = \{e_1, e_2, \ldots, e_m\}$, the edge set of G. We make v_i adjacent to e_j if and only if v_i is an endpoint of e_j.

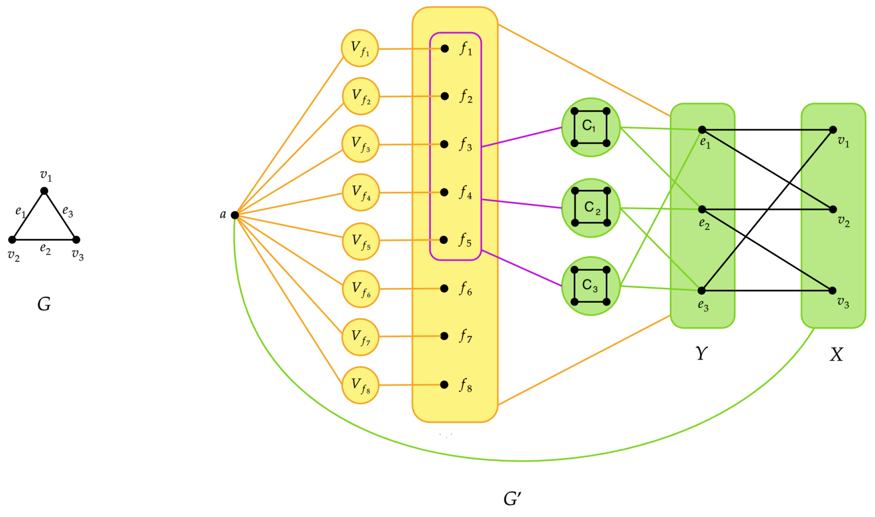


Figure 4.3: The reduction from Vertex Cover to Defensive Alliance.

2. For every $1 \leq i \leq m$, we introduce a cycle C_i of length 4. For every $1 \leq i \leq m-1$, make every vertex of C_i adjacent to e_i and e_{i+1}; and make every vertex of C_m adjacent to e_m and e_1.

3. We add a set $F = \{f_1, f_2, \ldots, f_8\}$ of 8 new vertices into G'. Set $k' = 5m + k$. For every vertex $f \in F$ we introduce a set V_f of $4k'$ new vertices into G' and make them adjacent to f. We make every vertex of $\{f_1, f_2, f_3, f_4, f_5\}$ adjacent to every vertex of C_i for $i = 1, 2, \ldots, m$. We also make every vertex of F adjacent to every vertex of Y.

4. Finally, we introduce a vertex a and make it adjacent to every vertex of $X \cup \bigcup\limits_{f \in F} V_f$.

We now argue equivalence of the instances. Suppose there exists a vertex cover S of size at most k in G. We show that $D = S \cup Y \bigcup\limits_{i=1}^{m} V(C_i)$ is a defensive alliance of size at most k' in G'. It is easy to verify that all the vertices in D are protected.

To prove the reverse direction of the equivalence, suppose now that D is a defensive alliance of size at most k' in G'. Clearly, no vertex from $Q = \{a\} \cup F \cup \bigcup\limits_{f \in F} V_f$ can be part of D. To prove the reverse direction, we need the following simple claim:

Claim 4.3.1. *Every defensive alliance D of G' contains the set $Y \bigcup\limits_{i=1}^{m} V(C_i)$.*

Proof. Since the defensive alliance is non-empty, it must contain a vertex from $X \cup Y \bigcup_{i=1}^{m} V(C_i)$.

Case 1: Suppose D contains e_i from Y. We observe that $\deg_{G'}(e_i) = 18$. Note that eight neighbours of e_i in F cannot be part of the solution as they belong to the forbidden set Q. This implies that we need to add at least one vertex from the set $V(C_{i-1}) \cup V(C_i)$ to the solution for the protection of e_i. Without loss of generality, suppose we include one vertex from $V(C_i)$ in the solution. Inclusion of one vertex from the set $V(C_i)$ in the solution forces $V(C_i) \subseteq D$. This in turn forces e_{i+1} in the solution. Repeatedly applying the above argument, we see that $Y \bigcup_{i=1}^{m} V(C_i) \subseteq D$.

Case 2: Suppose D contains an arbitrary vertex from the set $X \bigcup_{i=1}^{m} V(C_i)$. Then the protection of that vertex forces at least one vertex from Y in the solution. Using the argument in Case 1, it implies that $Y \bigcup_{i=1}^{m} V(C_i) \subseteq D$. This completes the proof of the claim. ⌟

We observe that for every vertex $e \in Y$, we have included eight out of its 18 neighbours in the solution. For the protection of e, we need to include at least one more of its neighbours from X in the solution. As we have already added $5m$ vertices in the solution, we can add a set $S \subseteq X$ of at most k vertices in D such that every vertex $e \in Y$ has at least one neighbour in S. If such a set S exists then it forms a vertex cover of G. This shows that (G, k) is a yes-instance of Vertex Cover. □

4.4 Defensive Alliance on Circle Graphs

Recall that a *circle graph* is the intersection graph of a set of chords of a circle. That is, it is an undirected graph whose vertices can be associated with chords of a circle such that two vertices are adjacent if and only if the corresponding chords cross each other. Here, we prove that Defensive Alliance is NP-complete even when restricted to circle graphs, via a reduction from Dominating Set. It is known that Dominating Set on circle graphs is NP-hard [88].

Theorem 4.4.1. Defensive Alliance on circle graphs is NP-complete.

On the way towards this result, we provide a hardness result for a variant of Defensive Alliance which we require in the proof of Theorem 4.4.1. The problem Defensive Alliance$^{\mathrm{F}}$ generalizes Defensive Alliance where some vertices are forced to be outside the solution; these vertices are called forbidden vertices. This variant can be formalized as follows:

Defensive Alliance$^{\mathrm{F}}$
Input: An undirected graph $G = (V, E)$, a positive integer r and a set $V_{\square} \subseteq V(G)$ of forbidden vertices.
Question: Is there a defensive alliance $S \subseteq V$ such that $1 \leq |S| \leq r$, and $S \cap V_{\square} = \emptyset$?

Lemma 4.4.2. Defensive Alliance$^{\mathrm{F}}$ on circle graphs is NP-complete.

Proof. It is easy to see that the problem is in NP. To show that the problem is NP-hard we give a polynomial reduction from Dominating Set on circle graphs. Let (G, k) be an instance of Dominating Set, where G is a circle graph. Suppose we are also given the circle representation C of G. Without loss of generality, we can assume that none of the endpoints of chords overlap with each other. We create a graph G' and output the instance $(G', V_{\square}, k')$. See Figure 4.7. The steps given below describe the construction of G':

- **Step 1:** Take two distinct copies G_1 and G_2 of G and let v_i be the copy of $v \in V(G)$ in graph G_i. For each $v \in V$, make v_1 adjacent to every vertex of $N_{G_2}[v_2]$ and similarly make v_2 adjacent to every vertex of $N_{G_1}[v_1]$. Note that this operation can be easily incorporated in the circle representation by replacing the chord corresponds to v with two crossing cords correspond to v_1 and v_2 as shown in Figure 4.4.
- **Step 2:** For every $v \in V$, create two sets of vertices $X^v = \{x_1^v, \ldots, x_{2n+1}^v\}$ and $Y^v = \{y_1^v, \ldots, y_{2n+1}^v\}$ and make v_1, v_2 adjacent to every vertex of $X^v \cup Y^v$. This can be easily incorporated in circle representation by introducing $2n+1$ parallel chords for the vertices $x_1^v, \ldots, x_{2n+1}^v$ which cross the chords for v_1, v_2. Similarly, introduce $2n+1$ parallel chords for the vertices $y_1^v, \ldots, y_{2n+1}^v$ which cross the chords for v_1, v_2, as shown in Figure 4.5.
- **Step 3:** For each $x^v \in X^v$, create two 3-vertex cliques $C_{x^v}^1$ and $C_{x^v}^2$, and make x^v adjacent to every vertex of $C_{x^v}^1$ and $C_{x^v}^2$. For $1 \leq i \leq 2n$, make every vertex of $C_{x_i^v}^1$

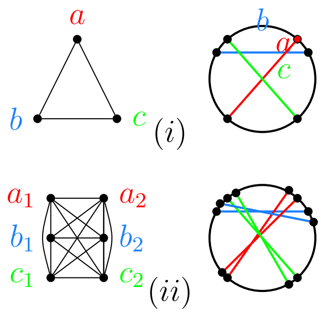


Figure 4.4: (i) Graph G and its circle representation. (ii) The graph produced after the first step of reduction and its circle representation.

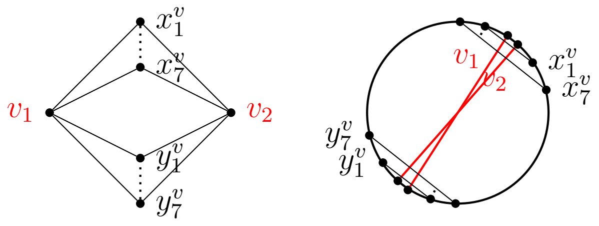


Figure 4.5: Illustration of Step 2. Here $2n + 1 = 7$.

adjacent to every vertex of $C^1_{x^v_{i+1}}$. Similarly, make every vertex of $C^2_{x^v_i}$ adjacent to every vertex of $C^2_{x^v_{i+1}}$ for $1 \leq i \leq 2n$. For each $y^v \in Y^v$, create two 3-vertex cliques $C^1_{y^v}$ and $C^2_{y^v}$, and make y^v adjacent to every vertex of $C^1_{y^v}$ and $C^2_{y^v}$. Make every vertex of $C^1_{y^v_i}$ adjacent to every vertex of $C^1_{y^v_{i+1}}$ for $1 \leq i \leq 2n$. Similarly, make every vertex of $C^2_{y^v_i}$ adjacent to every vertex of $C^2_{y^v_{i+1}}$ for $1 \leq i \leq 2n$. We start at an arbitrary vertex on the circle representation of C of G and then traverse the circle in counter clockwise direction. We record the sequence in which the chords are visited. For example, in Figure 4.4(i), if we start at the red vertex on the circle, then the sequence in which the chords are visited, is a, c, b, a, c, b. Note that every vertex appears twice in the sequence as every chord is visited twice while traversing the circle. Thus we get a sequence S of length $2n$ where n is the number of chords. We use the sequence to connect newly added cliques. For every consecutive pair (u, v) in the sequence S, make every vertex of $C^2_{x^u_{2n+1}}$ adjacent to every vertex of $C^1_{x^v_{2n+1}}$ when both u and v appear for the first time in the sequence S; make every vertex of $C^2_{y^u_{2n+1}}$ adjacent to every vertex of $C^1_{y^v_{2n+1}}$ when both u and v appear for the second time; and make every vertex of $C^2_{x^u_{2n+1}}$ adjacent to every vertex of $C^1_{y^v_{2n+1}}$ when u appears for the first time and v appear for the second

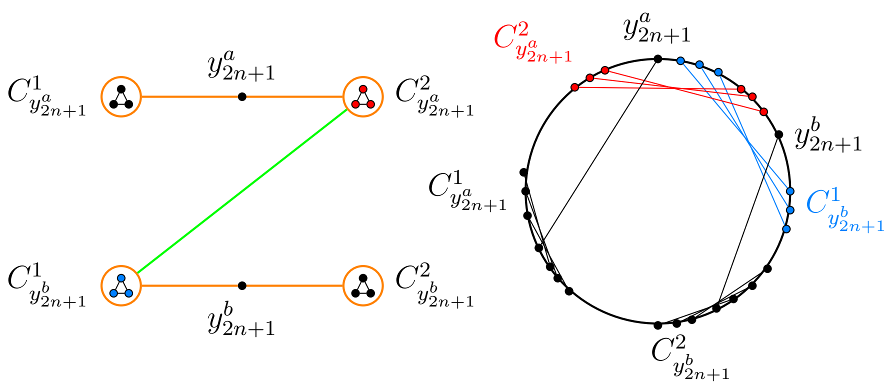


Figure 4.6: Illustration of Step 3.

time. These adjacencies are shown in green color in Figure 4.7 and 4.6.

- **Step 4:** For every vertex u in cliques, add d forbidden vertices where d is the degree of u until now in G' and make them adjacent to u. For every vertex $u \in X^v \cup Y^v$, add six forbidden vertices and make them adjacent with u. For very vertex $u \in V(G_1) \cup V(G_2)$, add $4n + 3$ forbidden vertices and make them adjacent to u. This completes the construction of G'. We set $k' = 7n(4n + 2) + n + k$ and $V_\square$ be the set of all one degree forbidden vertices.

We observed that the constructed graph G' is indeed a circle graph, and the construction can be performed in time polynomial in n. We now claim that G admits a dominating set of size at most k if and only if G' admits a defensive alliance D of size at most k' such that $D \cap V_\square = \emptyset$. Assume first that G admits a dominating set S of size at most k. Let

$$\mathcal{C} = \bigcup_{v \in V(G)} \bigcup_{i=1}^{2n+1} V(C^1_{x^v_i}) \cup V(C^2_{x^v_i}) \cup V(C^1_{y^v_i}) \cup V(C^2_{y^v_i}).$$

Consider $D = \mathcal{C} \cup \Big\{ v_1 \ : \ v \in S \Big\} \cup V(G_2) \cup \bigcup_{v \in V(G)} X^v \cup Y^v$. Clearly, $|D| \leq 7n(4n+2) + n + k$ and $D \cap V_\square = \emptyset$, so it suffices to prove that D is a defensive alliance in G'. We observe that every vertex in $\mathcal{C}$ has equally many neighbours inside and outside the solution. Every vertex $v \in \bigcup_{v \in V(G)} X^v \cup Y^v$ is protected as it has at least 7 neighbours inside the solution and at most 7 neighbours outside the solution. Each $v \in \Big\{ v_1 \ : \ v \in S \Big\} \bigcup V(G_2)$ has at least $d + 1 + 4n + 2$ neighbours inside the solution and at most $d + 1 + 4n + 2$ neighbours outside

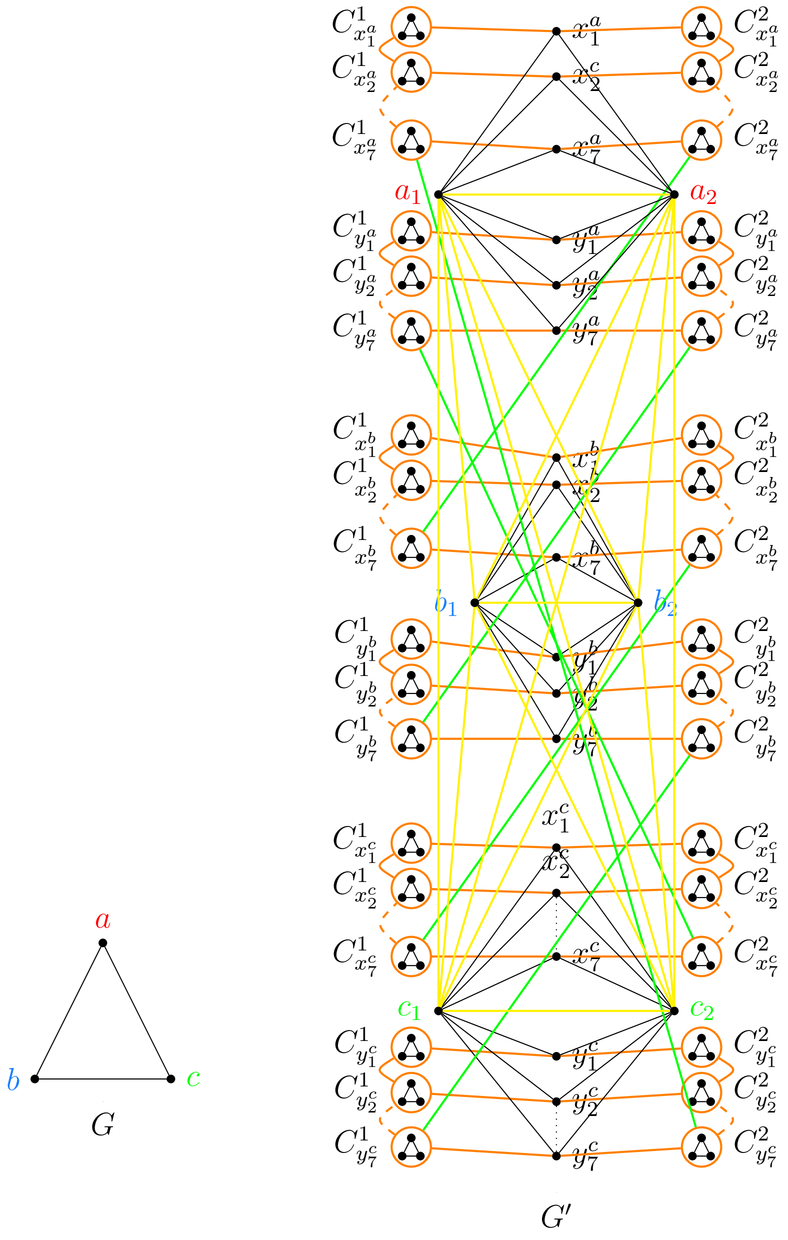


Figure 4.7: The reduction of an instance G of Dominating Set on circle graphs to an instance G' of Defensive Alliance$^{\mathrm{F}}$ in Theorem 4.4.1. Here $2n + 1 = 7$. One degree forbidden vertices introduced in Step 4 are not shown here.

the solution where $d = d_G(x)$. This shows that D is a defensive alliance of size at most k' in G'.

Conversely, suppose that G' admits a defensive alliance D of size at most k' such that $D \cap V_\square = \emptyset$. We define

$$V_\triangle = \mathcal{C} \cup \bigcup_{v \in V(G)} X^v \cup Y^v.$$

We first show that $V_\triangle \subseteq D$. Since D in non-empty, it should contain a vertex from either $V_\triangle$ or $V(G_1) \cup V(G_2)$. We consider the following cases:

Case 1: Suppose D contains a vertex from $\mathcal{C}$. Without loss of generality, we may assume that D contains a vertex u from $V(C^1_{x^v})$. It is easy to see that u is protected if and only if all its non-forbidden neighbours are inside D because the number of forbidden neighbours of u is equal to the number of non-forbidden neighbours. This implies that $x^v \in D$. It is easy to note that either $C_{x^v} \subseteq D$ or $C_{x^v} \cap D = \emptyset$. Since $d_{G'}(x^v) = 15$ and x^v has 6 forbidden neighbours, the above observation implies that $C^1_{x^v} \cup C^2_{x^v} \subseteq D$. This implies that $\bigcup_{i=1}^{2n+1} C^1_{x^v_i} \cup C^2_{x^v_i} \subseteq D$. This in turn implies that $X^v \subseteq D$. Note that every vertex of $C^2_{x^v_{2n+1}}$ is adjacent to every vertex of $C^1_{x^w_{2n+1}}$ (resp. $C^1_{y^w_{2n+1}}$) for some $w \in G$ such that u, w are consecutive elements in the sequence S and w appears for the first (resp. second) time in the sequence. Therefore $\bigcup_{i=1}^{2n+1} C^1_{x^w_i} \cup C^2_{x^w_i} \subseteq D$ and also $X^w \subseteq D$ if w appears for the first time in the sequence; whereas $\bigcup_{i=1}^{2n+1} C^1_{y^w_i} \cup C^2_{y^w_i} \subseteq D$ and also $Y^w \subseteq D$ if w appears for the second time in the sequence. Repeatedly applying the above argument, we get $V_\triangle \subseteq D$.

Case 2: Suppose D contains a vertex from $\bigcup_{v \in V(G)} X^v \cup Y^v$. Without loss of generality, we may assume that D contains x^v from X^v. We observe that the protection of x^v clearly requires at least one vertex from the set $C^1_{x^v} \cup C^2_{x^v}$. Now, Case 1 implies that $V_\triangle \subseteq D$.

Case 3: Suppose D contains a vertex v from $V(G_1) \cup V(G_2)$. The protection of v requires

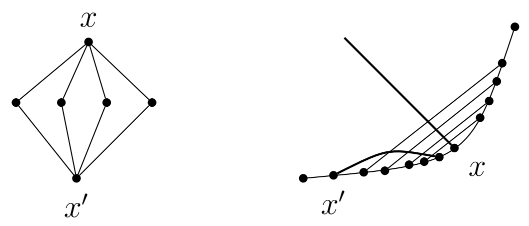


Figure 4.8: The circle representation to get rid of forbidden vertices when $k' = 2$.

at least one vertex from the set $X^v \cup Y^v$. Now, Case 2 implies that $V_\triangle \subseteq D$.

Observe that $|V_\triangle| = 7n(4n+2)$. Therefore $|D \cap (V(G_1) \cup V(G_2))| \leq n + k$. For each $v \in V(G)$, the protection of every vertex in $X^v \cup Y^v$ requires either v_1 or v_2 inside the solution. Since, v_1 and v_2 are twins, we can assume that $V(G_2) \subseteq D$. Let $v_2 \in V(G_2)$. We see that v_2 has $d + 1 + 4n + 2$ neighbours (including itself) inside the solution. The vertex v_2 has $4n + 3$ forbidden neighbours. The only unsettled neighbours of v_2 are in $V(G_1)$ and v_2 has $d + 1$ neighbours in $V(G_1)$. For protection of each $v_2 \in V(G_2)$, we require at least one neighbour from $V(G_1)$ inside the solution. We can add at most k vertices from $V(G_1)$ to the solution as we have already added $7n(4n + 2) + n$ vertices. Clearly, $S = V(G_1) \cap D$ is a dominating set of size at most k. $\square$

4.4.1 Proof of Theorem 4.4.1

It is easy to see that the problem is in NP. To show that the problem is NP-hard we give a polynomial reduction from DEFENSIVE ALLIANCE$^{\text{F}}$. Let $(G, k, V_\square)$ be an instance of DEFENSIVE ALLIANCE$^{\text{F}}$, where G is a circle graph. We construct an instance (G', k') of DEFENSIVE ALLIANCE the following way. For every $x \in V_\square$, create a vertex x' and a set of $2k'$ vertices $V_\square^x$. Make both x and x' adjacent to every vertex in $V_\square^x$. This completes the construction of G'. Set $k' = k$.

We observe in Figure 4.8 that the constructed graph G' is indeed a circle graph, and the construction can be performed in time polynomial in n. We now claim that G admits a defensive alliance D of size at most k such that $D \cap V_\square = \emptyset$ if and only if G' admits a defensive alliance D' of size at most k'. Assume first that D is a defensive alliance of size at most k in G such that $D \cap V_\square = \emptyset$. Consider $D' = D$. Clearly, D' is a defensive alliance of size at most k' in G'. Conversely, suppose that G' admits a defensive alliance D' of size at most k'. Observe that $D' \cap \bigcup_{x \in V_\square} V_\square^x \cup \{x, x'\} = \emptyset$. As x and x' are of degree $2k'$, they cannot be part of a defensive alliance of size at most k'. As x and x' are outside D', the vertices in $V_\square^x$ cannot be in D'. Consider $D = D'$. Clearly, D is a defensive alliance of size at most k in G such that $D \cap V_\square = \emptyset$. □

4.5 Closing Remarks and Future Directions

In this work we proved that the Defensive Alliance problem is W[1]-hard parameterized by a wide range of fairly restrictive structural parameters such as the feedback vertex set number, pathwidth, treewidth, treedepth, and clique width of the input graph, even when restricted to bipartite graph. We also proved that the problem parameterized by the vertex cover number of the input graph does not admit a polynomial compression unless coNP $\subseteq$ NP/poly; it cannot be solved in time $2^{o(n)}$, unless ETH fails, and the Defensive Alliance problem on circle graphs is NP-complete. By the construction of our proofs in Section 4.1, it is clear that hardness also holds for problem variants that ask for defensive alliances exactly of a given size. In the future it may be interesting to study if our ideas can be useful for different kinds of alliances from the literature such as offensive and powerful alliances. The parameterized complexity of defensive alliance problems remain unsettled when parameterized by other important structural graph parameters like twin cover and modular-width. It will also be interesting to study W[t]-membership of Defensive Alliance problem parameterized by treewidth. In [18], Hans L. Bodlaende, Gunther Cornelissen and Marieke van der Wegen put up an open question that whether Defensive Alliance is XNLP-hard when parameterized by treewidth. Note that this would imply that Defensive Alliance is W[t]-hard for all t when parameterized by treewidth of the input graph. It is also interesting to see if Defensive Alliance is W[1]-hard when parameterized by stable gonality [18].

Chapter 5

Locally Minimal Defensive Alliance in Graphs

This chapter deals with the problem Locally Minimal Defensive Alliance: Given an undirected graph $G = (V, E)$ and an integer k, the question is whether there is a vertex subset $D \subseteq V$ of at least k vertices satisfying the following two conditions: 1. Each $x \in D$ has at least as many neighbors (including itself) in S as it has neighbors not in D. 2. Removing any vertex from D destroys Condition 1 for at least one remaining vertex in D. A vertex subset of any size fulfilling the first condition is called a defensive alliance. Clearly, the vertex set of the whole graph is a defensive alliance. Any defensive alliance satisfying the second condition is called a locally minimal defensive alliance. Locally Minimal Defensive Alliance is known to be NP-hard and its parameterized complexity has been studied before, but the question whether it is FPT when parameterized by the solution size k has still remained open. This question is answered positively here. Further results are a kernel with $k^{\mathcal{O}(k)}$ vertices on C_3-free and C_4-free graphs of minimum degree at least two and a subexponential algorithm with respect to k for planar graphs of minimum degree at least 2. This chapter is based on the paper [69].

5.1 Main Results and Proof Techniques

Theorem 5.1.1. Locally Minimal Defensive Alliance *is FPT.*

The proof of Theorem 5.1.1 relies on a win/win argument. We first find the diameter of G in polynomial time. We show that if $diam(G) \geq 4k^2$, then the given instance (G, k) is a yes-instance (see Theorem 5.2.4). Hence, we assume that $diam(G) < 4k^2$. If G has a high-degree vertex v such that v has a large number of neighbours of degree ≥ 2, then we show that G has a locally minimal defensive alliance of size at least k (Lemma 5.2.9). If G has a high-degree vertex v such that v has a small number of neighbours of degree ≥ 2, then we give an FPT algorithm to check whether there is a locally minimal defensive alliance containing v (Lemma 5.2.11). If G has no high-degree vertices, then as $diam(G) < 4k^2$, the number of vertices in G is bounded by a function of k alone, and we are done. Our algorithm starts with trivial defensive alliance $V(G)$ and tries to construct a locally minimal defensive alliance of size at least k in a greedy fashion. Either we succeed in this process or derive some useful structural properties of the greedy solution. The difficult part is to keep witnesses of minimality for our solution. This is highly non-trivial and uses several interesting ideas.

The running time of our algorithm on general graphs is quite large and thus we also study the problem on special graph classes and obtain exponential kernels in polynomial time.

Theorem 5.1.2. Locally Minimal Defensive Alliance *on C_3-free or C_4-free graphs of minimum degree at least 2, admits a kernel with at most $k^{\mathcal{O}(k)}$ vertices.*

The proof of the above theorem uses some subtle structural observations on these special graph classes which allows us to bound the maximum degree of the input graph by $\mathcal{O}(k^2)$ to obtain the kernel mentioned in Theorem 5.1.2. The next special class of graphs that we consider are planar graphs. On this class we obtain a subexponential time algorithm.

Theorem 5.1.3. Locally Minimal Defensive Alliance *on the planar graphs of minimum degree at least 2, admits an FPT algorithm with running time $k^{\mathcal{O}(\sqrt{k})}$.*

The proof of this theorem relies on a win/win argument that exploits the relation between treewidth and the linear grid theorem [54]. We show that the existence of "$\Gamma_{10\sqrt{k}+4}$" as a minor obtained by only edge contractions guarantees the existence of locally minimal defensive alliance of size at least k. When this case does not arise, we use a dynamic programming algorithm on graphs of treewidth at most $\mathcal{O}(\sqrt{k})$ and maximum degree Δ with running time $\Delta^{\mathcal{O}(\sqrt{k})}n^{\mathcal{O}(1)}$. Finally, to get the required time complexity mentioned in Theorem 5.1.3, we show that if the maximum degree of the input graph is at least $\mathcal{O}(k^5)$, then there always exists a locally minimal defensive alliance of size at least k.

Finally, getting rid of the minimum degree 2 requirement in both Theorems 5.1.2 and 5.1.3, are interesting open questions.

5.2 FPT algorithm parameterized by solution size

Let D be a defensive alliance in G. A vertex $u \in D$ is said to be a *good* vertex if it has at least one marginally protected neighbor in D. A vertex $u \in D$ is said to be a *bad* vertex if it has no marginally protected neighbor in D. In this section we propose a simple greedy algorithm (Algorithm 1) that takes as input a defensive alliance and returns a locally minimal defensive alliance. At each iteration of Algorithm 1, if D contains a bad vertex, we identify it and remove from D. During the execution of Algorithm 1, a good vertex may become a bad vertex and finally gets removed from D. For an illustration, see the graph in Figure

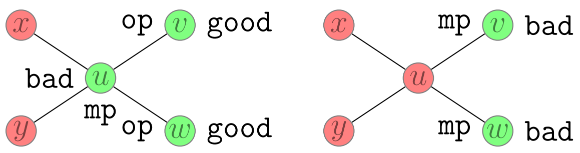


Figure 5.1: Left hand side figure shows a graph G and a defensive alliance D. The vertices of D are shown in green; u is a bad vertex where as v and w are good vertices. Right hand side figure shows the defensive alliance after the first iteration of Algorithm 1. Note that initially v and w are good vertices but after the first iteration of Algorithm 1, they become bad vertices.

5.1. The vertices of defensive alliance D are shown in green. Here u is marginally protected (mp) where as v and w are over protected (op); v and w are good vertices as they have a marginally protected neighbour u; on the other hand u is a bad vertex as it has no marginally protected neighbour in D. At the end of the first iteration, u is removed from D as u is a bad vertex. Then v and w become bad vertices as they have no marginally protected neighbour in the updated D. Finally, v and w are also removed from D. We now introduce the notion of crucial vertices in a defensive alliance.

A good vertex is called a crucial vertex of D if it is not removed from D during the execution of Algorithm 1. Consider the graph in Figure 5.2. During the execution of Algorithm 1, only u_4 will be removed from D; u_1, u_2, u_3 will not be removed from D. Thus u_1, u_2, u_2 are crucial vertices. Note that u_1 and u_2 are adjacent and marginally protected in D.

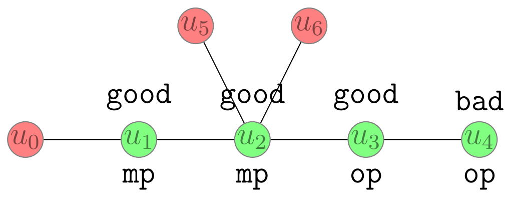


Figure 5.2: A graph G and a defensive alliance $D = \{u_1, u_2, u_3, u_4\}$. Here u_1, u_2 and u_3 are crucial vertices of D.

Now we characterize crucial vertices in a defensive alliance D. Suppose u_1 and u_2 are two marginally protected and adjacent vertices in D. Let $N_D(\{u_1, u_2\})$ be the set of neighbours of $\{u_1, u_2\}$ in D. It is easy to note that during the execution of Algorithm 1, the vertices of $\{u_1, u_2\} \cup N_D(\{u_1, u_2\})$ remain good and hence never removed from D. Thus the vertices in $\{u_1, u_2\} \cup N_D(\{u_1, u_2\})$ are crucial vertices. More formally, we have the following definition:

Definition 5.2.1. Let D be a defensive alliance in G. A vertex $u \in D$ is said to be a *crucial vertex* of D if u has a marginally protected neighbor u_1 in D and u_1 also has a marginally protected neighbor u_2 in D (u_2 and u are not necessarily distinct vertices).

Lemma 5.2.1. *A defensive alliance D of size at least two is a locally minimal defensive alliance in G if and only if every vertex of D is crucial.*

Proof. Suppose D is a locally minimal defensive alliance of size at least two in G. For any $u \in D$, $D \setminus \{u\}$ is not a defensive alliance. This implies that u must have a marginally protected neighbor u_1 in D. Due to the same reason u_1 must also have a marginally protected neighbor u_2 in D. Therefore u is a crucial vertex in D. On the other hand, suppose every vertex of D is crucial. Therefore, for any $u \in D$, $D \setminus \{u\}$ cannot form a defensive alliance as u is adjacent to a marginally protected vertex in D. □ We

now propose a simple greedy algorithm that takes as input a defensive alliance and returns a locally minimal defensive alliance.

During the execution of Algorithm 1, a good vertex may become a bad vertex and finally gets removed from D. Similarly during the execution of Algorithm 1, a good or bad vertex may become a crucial vertex, and then it never gets deleted from D. For example, see Figure 5.3.

Lemma 5.2.2. *Let u be a crucial vertex in D_{in}. Then the locally minimal defensive alliance D obtained by Algorithm 1 contains u.*

Algorithm 1

```
Require: A graph $G = (V, E)$ and a defensive alliance $D_{in}$ in $G$
Ensure: A locally minimal defensive alliance $D$
1: $D = D_{in}$
2: while $D$ contains a bad vertex do
3:     Identify a bad vertex $u \in D$
4:     $D = D \setminus \{u\}$
5: end while
6: return $D$
```

Proof. As u is a crucial vertex in D_{in}, u has a marginally protected neighbor u_1 in D_{in} and u_1 also has a marginally protected neighbor u_2 in D_{in}. As u_1 has a marginally protected neighbor u_2 in D_{in} and similarly u_2 has a marginally protected neighbor u_1 in D_{in}, u_1, u_2 will never be removed during the execution of Algorithm 1. As u has a marginally protected neighbor u_1 which will never be removed, u will also never be removed from D_{in}. Thus the locally minimal defensive alliance D obtained by Algorithm 1 contains all crucial vertices of D_{in}. □

We now modify Algorithm 1. Note that in Algorithm 1, the bad vertices of D were getting removed from D_{in} in an arbitrary order. In the following algorithm, we restrict the order in which the bad vertices will be removed from D_{in}. First, the bad vertices of $D_{in} \setminus C$ will be removed, and then the bad vertices of C will be removed. Note that due to Lemma 5.2.2, we know that if there exists a defensive alliance D_{in} that contains k crucial vertices then we have a locally minimal defensive alliance D of size at least k. The importance of introducing the set C is explained in the next lemma.

Algorithm 2

```
Require: A graph $G = (V, E)$, a defensive alliance $D \subseteq V(G)$ and a set $C \subseteq D$.
Ensure: A locally minimal defensive alliance.
1: while $D \setminus C$ contains a bad vertex of $D$ do
2:     Identify a bad vertex $u \in D \setminus C$
3:     $D = D \setminus \{u\}$
4: end while
5: run Algorithm 1 on $(G, D)$.
```

Lemma 5.2.3. *Suppose D_0 is the defensive alliance obtained at the end of the* **while** *loop in Algorithm 2. Then every vertex in $D_0 \setminus N[C]$ is a crucial vertex of D_0. Furthermore, every vertex in $D_0 \setminus N[C]$ will be there in the locally minimal defensive alliance obtained at*

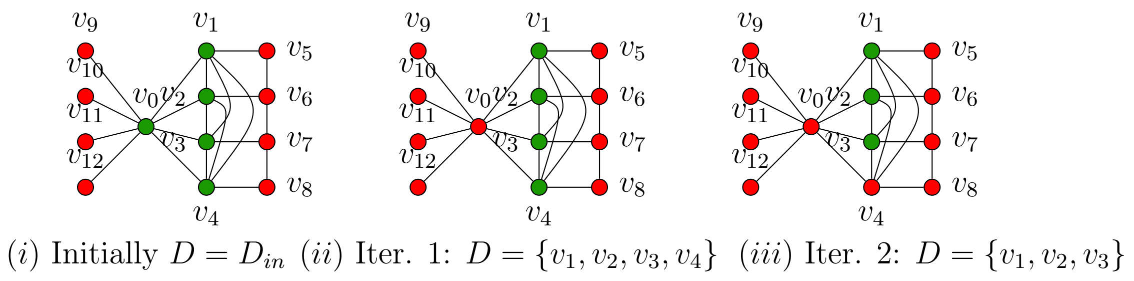

v_9
v_1
v_5
v_{10}
v_0
v_2
v_6
v_{11}
v_3
v_7
v_{12}
v_8
v_4
(i) Initially $D = D_{in}$
(ii) Iter. 1: $D = \{v_1, v_2, v_3, v_4\}$
(iii) Iter. 2: $D = \{v_1, v_2, v_3\}$

Figure 5.3: The execution of Algorithm 1 on the input graph G and $D_{in} = \{v_0, v_1, v_2, v_3, v_4\}$. In each iteration, green vertices are in D and red vertices are outside D. (i) Initially the vertices v_1, v_2, v_3, v_4 are good in D_{in} as they have a marginally protected neighbor v_0. On the other hand, v_0 is a bad vertex in D_{in} as it has no marginally protected neighbor in D_{in} (note that vertices v_1, v_2, v_3, v_4 are overprotected in D_{in}). (ii) In the first iteration, we remove the bad vertex v_0 and get $D = \{v_1, v_2, v_3, v_4\}$. All the vertices of D are bad. (iii) In the second iteration, we remove say v_4 and get $D = \{v_1, v_2, v_3\}$. Note that all the vertices of D are good as well as crucial in D now. Thus the algorithm stops and returns $D = \{v_1, v_2, v_3\}$, a locally minimal defensive alliance. It is important to note that initially, v_4 was a good vertex in $D = D_{in}$, but it has been removed from D during the execution of the algorithm. Thus during the execution of Algorithm 1, a good vertex may become bad and get deleted; but a crucial vertex will never be deleted.

the end of Algorithm 2.

Proof. Let u be an arbitrary vertex in $D_0 \setminus N[C]$. Since every vertex in $D_0 \setminus C$ has a marginally protected neighbor in D_0, u also has a marginally protected neighbor u_1 in D_0. Clearly, u_1 is in $D_0 \setminus C$. Therefore u_1 also has a marginally protected neighbor u_2 (u_2 can be u) in D_0. By Definition 5.2.1, u is a crucial vertex in D_0. Therefore every vertex in $D_0 \setminus N[C]$ is a crucial vertex. Due to Lemma 5.2.2, since every vertex in the set $D_0 \setminus N[C]$ is crucial in D_0, applying Algorithm 1 will return a locally minimal defensive alliance which contains the set $D_0 \setminus N[C]$. □

Next, we show that a correct choice of C in Algorithm 2 can generate a locally minimal defensive alliance of size at least k in G.

5.2.1 Graphs with diameter $\geq 4k^2$

Let G be a graph with $diam(G) \geq 4k^2$. In this section, we prove that G contains a locally minimal defensive alliance of size at least k. In particular, we prove the following theorem.

Theorem 5.2.4. *Let k be a positive integer and S_{in} be a defensive alliance in G. If $diam(G[S_{in}]) \geq 4k^2$, then G has a locally minimal defensive alliance of size at least k. Moreover, such a locally minimal defensive alliance can be obtained in polynomial time.*

We will begin by applying Algorithm 3 on defensive alliance S_{in}.

Algorithm 3

Require: A graph $G = (V, E)$ and a defensive alliance S_{in} in G
Ensure: A modified defensive alliance S

```
 1: $S = S_{in}$
 2: while $S$ has either a degree one bad vertex or a bad vertex with a degree one neighbor
    in $S$ do
 3:     for each $u \in S$ do
 4:         if $u$ is a bad vertex and $\deg_G(u) = 1$ then
 5:             $S = S \setminus \{u\}$
 6:         end if
 7:     end for
 8:     for each $v \in S$ do
 9:         if $v$ is a bad vertex and $v$ has a neighbor say $u$ in $S$ with $\deg_G(u) = 1$ then
10:             $S = S \setminus \{v\}$
11:         end if
12:     end for
13: end while
14: return $S$
```

Lemma 5.2.5. *Let S be the defensive alliance obtained at the end of Algorithm 3. Let $u, v \in S$ be two adjacent vertices such that $deg_G(u) = 1$ and v be the only neighbor of u in G. Then both u and v are crucial.*

Proof. As u is a degree one vertex, its only neighbor $v \in S$ must be marginally protected; otherwise, u is a bad vertex and the **for** loop of line 2-6 in Algorithm 3 would have deleted u. Note that u is overprotected. Thus v must be adjacent to a marginally protected vertex w in S; otherwise the **for** loop of line 7-11 in Algorithm 3 would have deleted v. This shows that both u and v must be crucial in S. This completes the proof of the lemma. □

Let S be the defensive alliance obtained at the end of Algorithm 3. Note that $G[S]$ may be disconnected. Suppose $S_1, S_2, \ldots, S_l$ are the connected defensive alliances such that $S = \bigcup_{i=1}^{l} S_i$.

Lemma 5.2.6. *For each $i \in [l]$, Algorithm 1 on (G, S_i) produces a locally minimal defensive alliance of size of at least two.*

Proof. For the sake of contradiction, suppose Algorithm 1 on (G, S_i) produces a locally minimal defensive alliance $\{u\}$ of size exactly one. It is easy to see that u must be a degree one vertex. Suppose the only neighbor of u is v. By Lemma 5.2.5, both u and v are crucial vertices in S_i. So the connected locally minimal defensive alliance obtained by Algorithm 1 on (G, S_i) contains both u and v, a contradiction to the assumption that it is of size exactly one. This proves the lemma. □

Lemma 5.2.7. *Suppose S_{in} is the input defensive alliance in Algorithm 3. Suppose the diameter of graph $G[S_{in}] \geq 4k^2$, that is, there exists a pair of vertices $u, v \in S_{in}$ such that $d(u, v) = 4k^2$ in $G[S_{in}]$. Let $P = (u = v_0, v_1, v_2, \ldots, v_{4k^2} = v)$ be a shortest $u - v$ path in $G[S_{in}]$. Then for each $1 \leq i \leq 4k^2 - 1$, either $v_i \in S$ or $v_{i+1} \in S$ where S is the defensive alliance obtained at the end of Algorithm 3.*

Proof. For the sake of contradiction suppose that both v_i and v_{i+1} are not in S for some $1 \leq i \leq 4k^2 - 1$. As both v_i and v_{i+1} have a degree of at least two, they must have been deleted by Algorithm 3 during the execution of second **for** loop of lines 8-12. Observe that v_i and v_{i+1} must have at least one neighbor of degree one. As these degree one neighbors are not deleted during the execution of the first **for** loop, it implies that v_i and v_{i+1} are marginally protected. As v_i is adjacent to v_{i+1}, it is clear that both v_i and v_{i+1} are crucial. Therefore Algorithm 3 cannot delete either of them. This is a contradiction. □

Due to Lemma 5.2.7, it is clear that if $diam(G[S_{in}]) \geq 4k^2$ either S contains a connected defensive alliance S_i of diameter at least $8k$ or a collection of at least $\frac{k}{2}$ connected defensive alliances $S_1, S_2, \ldots, S_l$. In the latter case, it is easy to see that one can obtain a locally minimal defensive alliance of size at least two by simply applying Algorithm 1 on (G, S_i) for each i. This is true due to Lemma 5.2.6.

Lemma 5.2.8. *If there is a connected defensive alliance S_i of diameter at least $8k$ then there exists a locally minimal defensive alliance of size at least k.*

Proof. As $diam(G[S_i]) \geq 8k$, there exists a pair of vertices w_0 and w_{8k} such that $d(w_0, w_{8k}) = 8k$ in $G[S_i]$. Let $P = (w_0, w_1, \ldots, w_{8k})$ be a shortest w_0-w_{8k} path in $G[S_i]$. Set $C = \{w_{4i} \mid 0 \leq i \leq 2k\}$. Run Algorithm 2 on (G, S_i, C). Let $S_i(C)$ be the defensive alliance obtained at the end of the **while** loop of lines 1-4 in Algorithm 2. Note that $S_i(C)$ may not be a connected defensive alliance. Suppose $G[S_i(C_1)], G[S_i(C_2)], \ldots, G[S_i(C_p)]$ are the connected components of $S_i(C)$ which generates a partition of C into p parts $C_1, C_2, \ldots, C_p$ such that $C_j \subseteq S_i(C_j)$ for all $j \in [p]$. Line 5 of Algorithm 2 executes Algorithm 1 on $(G, S_i(C))$, that is, it executes Algorithm 1 on $(G, S_i(C_j))$ for all $j \in [p]$.

Claim 5.2.1. *Line 5 of Algorithm 2 runs Algorithm 1 on $(G, S_i(C_j))$ and outputs a locally minimal defensive alliance S_{ij} such that $|S_{ij}| \geq \max\{2, |C_j| - 1\}$ for all $j \in [p]$.*

Proof. Due to Lemma 5.2.6, we always have $|S_{ij}| \geq 2$ as $S_i(C_j) \subseteq S_i$. We now prove that Algorithm 1 on input instance $(G, S_i(C_j))$ returns a locally minimal defensive alliance S_{ij} of size at least $|C_j| - 1$. Suppose $C_j = \{c_1, \ldots, c_q\}$, $q \geq 2$. We know $d(c_l, c_{l+1}) \geq 4$ and $d(w_0, c_l) < d(w_0, c_{l+1})$ in $G[S_i(C_j)]$ for all $1 \leq l \leq q-1$. As $G[S_i(C_j)]$ is connected, a shortest path between c_l and c_{l+1} in $G[S_i(C_j)]$ contains a vertex, say x, such that $x \in S_i(C_j) \setminus N[C_j]$. By Lemma 5.2.3, x is a crucial vertex in $S_i(C_j)$. As C_j has q vertices, we have at least $q - 1$ crucial vertices in $S_i(C_j)$. By Lemma 5.2.2, the locally minimal defensive alliance S_{ij} returned by line 4 of Algorithm 2 contains all $q - 1$ crucial vertices of S_{ij}. Thus we have $|S_{ij}| \geq q - 1$. This completes the proof of the claim.

As the closed neighborhood of one S_{ij} does not intersect others and $|C| \geq 2k$, we get $\bigcup_{j=1}^{p} S_{ij}$ is a locally minimal defensive alliance of size at least k. This completes the proof of the lemma. □

5.2.1.1 Proof of Theorem 5.2.4

Proof. We present a polynomial-time algorithm to obtain a locally minimal defensive alliance of size at least k. We begin by applying Algorithm 3 on (G, S_{in}). Let $S = \bigcup_{i=1}^{l} S_i$ be the defensive alliance obtained at the end of Algorithm 3 where $S_1, S_2, \ldots, S_l$ are connected defensive alliances such that $N[S_i] \cap S_j = \emptyset$ for all $i, j \in [l]$ and $i \neq j$. Due to Lemma 5.2.6, we know that applying Algorithm 1 on each S_i returns a locally minimal defensive alliance of size at least 2. Therefore we can assume that $l < \frac{k}{2}$, otherwise it is a yes instance. As diameter of $G[S] \geq 4k^2$ and $l < \frac{k}{2}$, due to Lemma 5.2.7 we observe that there exists $i \in [l]$ such that $diam(G[S_i]) \geq 8k$. Due to Lemma 5.2.8, one can obtain a locally minimal defensive alliance of size at least k by applying Algorithm 2 on (G, S_i). As Algorithm 1, 2 and 3 run in polynomial time, the above algorithm runs in polynomial time as well. This completes the proof of Theorem 5.2.4. □

5.2.2 Graphs with diameter $< 4k^2$

Let G be a graph with $diam(G) < 4k^2$. We consider two cases: $\Delta(G)$ is large and $\Delta(G)$ is small.

5.2.2.1 $\Delta(G)$ is large:

We partition the set of vertices of G into two parts H and L. We define

$$H = \{v \in V(G) \mid \deg(v) > 2f_0(k)\} \text{ and } L = \{v \in V(G) \mid \deg(v) \leq 2f_0(k)\}$$

where $f_0(k) = k^{k^{c_0 k}}$ for some large constant c_0. The vertices of H are called *high-degree* vertices. For each $v \in V$, we define $N_2(v) = \{u \in N(v) \mid \deg(u) \geq 2\}$ and $N_1(v) = \{u \in N(v) \mid \deg(u) = 1\}$. We further partition H into two parts

$$H_1 = \{v \in H \mid |N_2(v)| < f_0(k)\} \text{ and } H_2 = \{v \in H \mid |N_2(v)| \geq f_0(k)\}.$$

Lemma 5.2.9. *If $H_2 \neq \emptyset$ then there exists a locally minimal defensive alliance of size at least k in G.*

Proof. Let us assume that $u_0 \in H_2$. We will show that there exists a locally minimal defensive alliance of size at least k in G.

Case 1. Let us assume that $|N_1(u_0)| \leq |N_2(u_0)|$. This implies that $V(G) \setminus N_1(u_0)$ is a defensive alliance. This is true because the number of neighbors of u_0 in $V(G) \setminus N_1(u_0)$ is greater than or equal to the number of neighbors of u_0 in $N_1(u_0)$. So u_0 is protected; the other vertices are clearly protected. Now run Algorithm 2 on $(G, V(G) \setminus N_1(u_0), \{u_0\})$ and suppose it returns D_0.

Subcase 1.1 If $u_0 \in D_0$ then at least $\frac{\deg(u_0)}{2} > k$ neighbors of u_0 are in D_0 for its protection. That means, $|D_0| \geq k$.

Subcase 1.2 Suppose u_0 is not in D_0. Let D_0' be the defensive alliance obtained at the end of the **while** loop of lines 1-4 in Algorithm 2 and all neighbors of u_0 in D_0' are overprotected. This is why u_0 is deleted from D_0' in line 5. We partition the vertices of defensive alliance $D_0' \setminus \{u_0\}$ into two parts: part $P_1^{u_0} \subseteq N_2(u_0)$ contains vertices adjacent to u_0 and part $P_2^{u_0}$ contains vertices not adjacent to u_0. Note that all the vertices in $P_1^{u_0}$ have a degree of at least two. Due to Lemma 5.2.3, the vertices of $P_2^{u_0}$ will be there in a final locally minimal defensive alliance as they are crucial in D_0'. Thus we have $|P_2^{u_0}| \leq k-1$, otherwise we can obtain a solution of size at least k by applying Algorithm 1 on D_0'. Furthermore, every vertex in $P_2^{u_0}$ has a degree at most $2k$, or else we have a solution of size at least k. As every vertex in $P_2^{u_0}$ has degree at most $2k$, we prove that $P_1^{u_0}$ contains at least one vertex u_1 with degree at least $f_1(k)$ where $f_1(k) = k^{k^{c_1 k}}$ for some large constant c_1. Note that $G[D_0' \setminus \{u_0\}]$ contains at most $\frac{k}{2} - 1$ connected components of size at least two, otherwise by applying Algorithm 1 on each of these defensive alliances, we will obtain a locally minimal defensive alliance of size at least k. Moreover, every component of $G[D_0' \setminus \{u_0\}]$ has diameter at most $4k^2$, otherwise by Theorem 5.2.4, it is a yes instance. Therefore, if the maximum degree of $G[D_0' \setminus \{u_0\}]$ is $f_1(k)$ then we know that it contains at most $\mathcal{O}(k \cdot f_1(k)^{4k^2})$ vertices. We also know that $G[D_0' \setminus \{u_0\}]$ contains at least $\mathcal{O}(f_0(k))$ vertices. Therefore, we get $cf_0(k) \leq c'k \cdot f_1(k)^{4k^2}$ where c and c' are two positive real numbers. This implies

$$f_1(k) \geq \left(\frac{c \cdot f_0(k)}{c' \cdot k}\right)^{\frac{1}{4k^2}}.$$

Since we have $f_0(k) = k^{k^{c_0 k}}$, we can assume that $f_1(k) = k^{k^{c_1 k}}$ for some large constant $c_1 \leq c_0$. As we have proved that the maximum degree of $G[D_0' \setminus \{u_0\}]$ is $f_1(k)$, there must exist a vertex in $P_1^{u_0}$ with degree at least $f_1(k)$. Let us call that vertex u_1.

We again run Algorithm 2 on $(G, D_0' \setminus \{u_0\}, \{u_1\})$. Let D_1' is a defensive alliance obtained at the end of the **while** loop of lines 1-4 in Algorithm 2 on $(G, D_0' \setminus \{u_0\}, \{u_1\})$. We can apply the same argument as before and either obtain a locally minimal defensive alliance of size at least k containing u_1 or a defensive alliance $D_1' \setminus \{u_1\}$ containing a vertex u_2 of degree

$f_2(k) = k^{k^{c_2 k}}$ for some large constant c_2.

We repeat the same procedure k times. Let u_k be the vertex obtained by the same procedure with a degree at least $f_k(k) = k^{k^{c_k k}}$ for some large constant c_k. Call Algorithm 2 on $(G, D'_{k-1} \setminus \{u_{k-1}\}, u_k)$ and suppose it returns D_k. If u_k is in D_k then at least $\frac{\deg(u_k)}{2} > k$ neighbors of u_k are in D_k for its protection. That means we have a locally minimal defensive alliance D_k of size at least k; so we are done. Suppose u_k is not in D_k. Let D'_k be the defensive alliance obtained at the end of the **while** loop of Algorithm 2. We assume that all neighbors of u_k in D'_k are overprotected and this is why u_k is deleted. We partition the vertices of defensive alliance $D'_k \setminus \{u_k\}$ into two parts: $P_1^{u_k}$ and $P_2^{u_k}$. A vertex x is in part $P_1^{u_k}$ if it is adjacent to all u_i's for $i \in [k]$ and a vertex x is in part $P_2^{u_k}$ if it is not adjacent to u_i for any $i \in [k]$. By Lemma 5.2.3, the vertices of $P_2^{u_k}$ are crucial, and therefore these crucial vertices will be there in the final locally minimal defensive alliance. If $|P_2^{u_k}| \geq k$, then we have a locally minimal defensive alliance of size at least k, and we are done. Thus we assume $|P_2^{u_k}| < k$. Consider a vertex $x \in P_1^{u_k}$. By the definition of $P_1^{u_k}$, x is adjacent to $k+1$ vertices $u_0, u_1, \ldots, u_k$ which are outside the defensive alliance $D'_k \setminus \{u_k\}$. As x is protected in $D'_k \setminus \{u_k\}$ it must have at least k neighbors in $D'_k \setminus \{u_k\}$. Thus $\deg_G(x) \geq 2k+1$. Line 6 of Algorithm 2 executes Algorithm 1 on $(G, D'_k \setminus \{u_k\})$; suppose it outputs S. We prove that at least one vertex of $P_1^{u_k}$ will survive in S. For the sake of contradiction, assume that this is false, that is, no vertex survive in S. Suppose vertex x in $P_1^{u_k}$ is deleted last. It has at most k neighbors, including itself, in the current defensive alliance as $P_2^{u_k}$ contains at most $k-1$ vertices and $k+1$ neighbors outside the current defensive alliance. That means, x is not protected, a contradiction to the fact that x is protected in $D'_k \setminus \{u_k\}$. Therefore, the locally minimal defensive alliance S obtained by Algorithm 2 contains a vertex of degree at least $2k+1$; hence S is of size at least k.

Case 2. Let us assume that $|N_1(u_0)| \geq |N_2(u_0)|$. Suppose $N(u_0) = \{u_1, \ldots, u_l\}$. As we know that $|N_1(u_0)| \geq |N_2(u_0)|$, without loss of generality, we can assume that $\deg(u_i) = 1$ for $i \in \left[\lceil \frac{l-1}{2} \rceil\right]$. Let us denote $N_1^*(u_0) = \{u_1, u_2, \ldots, u_{\lfloor \frac{l-1}{2} \rfloor}\} \subseteq N_1(u_0)$. Clearly, $V(G) \setminus N_1^*(u_0)$ is a defensive alliance. Now run Algorithm 2 on $(G, V(G) \setminus N_1^*(u_0), \{u_0\})$. Suppose it returns D_0.

Subcase 2.1 If $u_0 \in D_0$, then there exists a locally minimal defensive alliance of size at least k.

Subcase 2.2 Suppose $u_0 \notin D_0$. Let D'_0 be the defensive alliance obtained at the end of lines 1-4 in Algorithm 2 and all neighbors of u_0 in D'_0 are overprotected. This is why we deleted u_0 from D'_{u_0}. As u_0 is marginally protected in $V(G) \setminus N_1^*(u_0)$ and $N_2(u_0) \subseteq V(G) \setminus N_1^*(u_0)$ implies that $N_2(u_0) \subseteq D'_0$. This is true because we cannot remove the neighbors of u_0 anymore in $V(G) \setminus N_1^*(u_0)$ as u_0 is marginally protected in $V(G) \setminus N_1^*(u_0)$. Next, we consider the defensive alliance $D''_0 = D'_0 \setminus (N_1(u_0) \cup \{u_0\})$ where we get rid of degree one neighbors of u_0 from D'_0. Just as in Subcase 1.2, we can partition the vertices of D''_0 into two parts $P_1^{u_0} = \{w \in N(u_0) \cap D''_0\}$ and $P_2^{u_0} = \{w \notin N(u_0) \mid w \in D''_0)\}$. Clearly, all the vertices in $P_1^{u_0}$ are of degree at least 2, in fact, $P_1^{u_0} = N_2(u_0)$. Also, we know that $|N_2(u_0)| = |P_1^{u_0}| \geq f_0(k)$. From here on, we can apply the same argument as in Subcase 1.2 and obtain a locally minimal defensive alliance of size at least k. This proves Lemma 5.2.9. $\square$ Now let us focus on vertices in the set H_1. First, we make some simple observations about vertices in H_1.

Observation 5.2.1. Suppose $u \in H_1$ is contained in a locally minimal defensive alliance S and u is overprotected. Then u can be made marginally protected in S by moving some degree one neighbors of u from S to $V \setminus S$.

Observation 5.2.2. A graph G has a defensive alliance D with a pair of adjacent marginally protected vertices $u, v \in D$ if and only if G has a locally minimal defensive alliance S in which u is marginally protected.

Lemma 5.2.10. *Given a partition of $V(G)$ into three parts A, B, C, we can check in polynomial time if there is a defensive alliance D in G such that $A \subseteq D$ and $B \cap D = \emptyset$. Also if such a defensive alliance exists then it can be obtained in polynomial time.*

Proof. We begin with the set $D = A \cup C$. Note that D may not be a defensive alliance. We keep removing the unprotected vertices from D until all the vertices are protected. Let D' be the defensive alliance obtained by this method. If $A \subseteq D'$ then we return D' as the required defensive alliance. If $A \nsubseteq D'$ then one can see that there does not exist a defensive alliance satisfying the given constraints. $\square$

Lemma 5.2.11. *Given a vertex $v \in H_1$ and $H_2 = \emptyset$, we can determine if there exists a locally minimal defensive alliance containing v in FPT time.*

Proof. We know that there exists a locally minimal defensive alliance containing v if and only if there exists a defensive alliance S where v is a crucial vertex. Thus our goal here is to generate a defensive alliance where v is a crucial vertex. Due to Observation 5.2.1, we can assume that v is marginally protected in S. Also due to Lemma 5.2.10, it is enough to guess $N(v) \cap S$ and $N(N(v) \cap S) \cap S$ such that v will be marginally protected in S and also have a marginally protected neighbor. Note that as v is marginally protected in S, guessing $S \cap N_2(v)$ already determines $|S \cap N_1(v)|$. As $N_1(v)$ is a set of degree one vertices, it also determines $S \cap N_1(v)$ as any subset of $N_1(v)$ of size $|S \cap N_1(v)|$ will work. As $|N_2(v)| \leq f_0(k)$, it implies that there are at most $2^{f_0(k)}$ guesses for $S \cap N_2(v)$ and therefore for $N(v) \cap S$ as well. We partition the vertices in $N(v) \cap S \subseteq N_2(v)$ into two types. Type 1 contains vertices of degree at most $2f_0(k)$. Type 2 contains vertices of degree more than $2f_0(k)$. Since we know that $H_2 = \emptyset$, all the vertices of type 2 are contained in H_1. For every vertex w of type 1, we can guess $N(w) \cap S$ as the number of guesses required are $2^{\deg(w)}$ where $\deg(w) \leq 2f_0(k)$. We also know that type 1 and type 2 vertices are bounded by $f_0(k)$. Since there are at most $f_0(k)$ vertices of type 1, guessing $N(\text{type1}) \cap S$ will require at most $2^{(2f_0(k))^2}$ guesses. Now for every type 2 vertex w, since we know they are marginally protected in S, we need to only guess $N_2(w) \cap S$. Since $|N_2(w)| \leq f_0(k)$, we need to make $2^{f_0(k)}$ guesses. Determining $N_2(w) \cap S$ will automatically determine $|N_1(w) \cap S|$ and therefore $N_1(w) \cap S$ as well. We have made at most $2^{\mathcal{O}(2f_0(k)^2)}$ guesses. Due to Lemma 5.2.10, we can check in polynomial time whether there exists such a defensive alliance S, which satisfies the guesses above. If the above conditions are satisfied by S, that is, $v \in S$, v is marginally protected, and v is crucial then one can apply Algorithm 1 on S to get a locally minimal defensive alliance containing v. Therefore we have an FPT algorithm to determine whether there exists a locally minimal defensive alliance S containing v. Note that the running time of the algorithm is bounded by $2^{(f_0(k))^2} n^{\mathcal{O}(1)}$. By substituting the value of $f_0(k)$, we get the running time of the algorithm to be $2^{k^{k^{\mathcal{O}(k)}}} n^{\mathcal{O}(1)}$. This proves Lemma 5.2.11. $\square$

Given a graph G, we first check whether $H_2 \neq \emptyset$. If yes, then due to Lemma 5.2.9 conclude that we are dealing with a yes-instance. Therefore, we can assume that $H_2 = \emptyset$. Next, for each $v \in H_1$, we check in FPT time whether there exists a locally minimal defensive alliance containing v. If there exists a locally minimal defensive alliance containing v then conclude that we are dealing with a yes-instance as $\deg(v) > 2k$. Therefore, we can assume that there does not exist any locally minimal defensive alliance containing a vertex from H_1. So we remove the vertices of H_1 from graph G. To do this, we create a weighted graph G' from G in the following way: $G' = G - H_1$ and $w : V(G') \to \mathbb{Z}$ is a weight function where $w(v)$ is equal to the number of neighbors of v in H_1. Note that G' can be a disconnected graph. Suppose

$G'_1, G'_2, \ldots, G'_l$ are the connected components of G'. We observe that $D \subseteq V(G'_i)$ is a defensive alliance in G if for all $v \in D$, we have $\deg_D(v) + 1 \geq \deg_{V(G'_i)\setminus D}(v) + w(v)$. Given a connected component G'_i, if there exists a vertex $v \in V(G'_i)$ such that $\deg_{G'_i}(v) + 1 < w(v)$ then we update G'_i to $G'_i - \{v\}$ and update the weight of the vertices in $G'_i - \{v\}$ to $w(u) = w(u) + 1$ for all $u \in N(v)$, otherwise we keep the same weight. This is true because such a vertex v cannot be part of any locally minimal defensive alliance in G. Now let us assume that after this "reduction" procedure, we get $G''_1, G''_2, \ldots, G''_{l'}$, $l' \leq l$, as nonempty connected components.

Claim 5.2.2. *For each $i \in [l']$, $V(G''_i)$ forms a defensive alliance in G.*

Proof. Let $v \in V(G''_i)$. We know that $\deg_{V(G''_i)}(v) + 1 \geq w(v)$, otherwise v must have been deleted from G''_i. Note that $w(v) = \deg_{V(G)\setminus V(G''_i)}(v)$ by construction. Therefore, we have $\deg_{V(G''_i)}(v) + 1 \geq \deg_{V(G)\setminus V(G''_i)}(v)$ for all $v \in V(G''_i)$. This completes the proof of the claim. $\square$

Note that we have deleted the vertices v for which we know that there is no locally minimal defensive alliance containing v. Now, one can check if $diam(G[V(G''_i)]) \geq 4k^2$ for some i. If yes then conclude that we are dealing with a yes-instance due to Theorem 5.2.4. Therefore we can assume that $diam(G[V(G''_i)]) < 4k^2$ for all i. We also know that $\Delta(G''_i) \leq 2f_0(k)$. This means that we have a weighted graph where a function of k bounds the size of each connected component. From here, one can do brute force to find the largest locally minimal defensive alliance in each G''_i. Finally, we can add the sizes of the largest locally minimal defensive alliances from different connected components, and if the sum is at least k then conclude that we are dealing with a yes-instance.

5.2.2.2 $\Delta(G)$ is small:

Given a graph G, if we know that $\Delta(G)$ is at most $k^{k^{c_0 k}}$ for some constant c_0 and also that the diameter of each connected component is at most $4k^2$ then clearly each connected component size is bounded by a function of k only. As explained in previous paragraph, we can easily get a locally minimal defensive alliance of size at least k in FPT time.

5.2.3 Proof of Theorem 5.1.1

Given a graph G, we first check if the diameter of G is at least $4k^2$. This can be done in polynomial time. If yes, then due to Theorem 5.2.4 it is a yes instance. Hence, we assume that the diameter of G is less than $4k^2$. We find a vertex v in G with maximum degree $\Delta(G)$. As explained in Section 5.2.2.1, if $\Delta(G) > k^{k^{c_0 k}}$ for some constant c_0 then we can solve the problem in FPT time. Finally, if $\Delta(G) \leq k^{k^{c_0 k}}$ then due to Section 5.2.2.2, we know how to find a locally minimal defensive alliance of size at least k in FPT time. Let us calculate the total time required to solve the problems in Section 5.2.2.1 and Section 5.2.2.2. One can observe that the worst case time complexity is obtained when we solve the problem by brute force technique. Therefore the worst case time complexity is essentially the number of subsets of connected components whose maximum degree is bounded by $2f_0(k)$ and diameter is bounded by $4k^2$. That is, we need to consider at most $2^{f_0(k)^{4k^2}}$ many subsets. Therefore, we get the worst case time complexity to be $2^{k^{k^{\mathcal{O}(k^3)}}}.n^{\mathcal{O}(1)}$. This proves Theorem 5.1.1.

5.3 Kernels for Locally Minimal Defensive Alliance restricted to C_3-free and C_4-free graphs

In this section we give improved kernels for Locally Minimal Defensive Alliance restricted to C_3-free and C_4-free graphs, when the parameter is the solution size k. The following lemma is the basis for our kernelization algorithm.

Lemma 5.3.1. Let G be a C_3-free graph with a minimum degree at least 2. If $\Delta(G) \geq 4k^2$ then G has a locally minimal defensive alliance of size at least k.

Proof. Suppose $d(u) = 4k^2$. Run BFS(G, u) to obtain the BFS tree T rooted at u and to get level for each vertex in G. Let L_i be the set of vertices at level i of T. Run Algorithm 2 on $(G, V(G), u)$ and suppose it outputs D. If $u \in D$ then clearly D is a locally minimal defensive alliance of size at least $2k^2$ as the degree of u is $4k^2$ and u is protected in D. Consider the case where u is not in D. Let D' be the defensive alliance obtained at line 5 of Algorithm 2. As u is protected in D' at least $2k^2$ of its neighbours are in $L_1 \cap D'$. All neighbors of u in $L_1 \cap D'$ are overprotected, this is why u is deleted from D' in line 6. As G is triangle-free and $\delta(G) \geq 2$, every $x \in L_1 \cap D'$ has least one neighbour in $L_2 \cap D'$. We know every vertex from level 2 or higher that survives in the defensive alliance obtained by Algorithm 2 is crucial and they remain in the final locally minimal defensive alliance. This implies that there are at most $k-1$ vertices in $D' \cap L_2$, otherwise we have a solution of size at least k. Note that there are at least $2k^2$ vertices in $D' \cap L_1$, at most $k-1$ vertices in $L_2 \cap D'$ and each vertex in $D' \cap L_1$ has at least one neighbor in $L_2 \cap D'$. By the Pigeonhole principle there exists a vertex $w \in L_2 \cap D'$ of degree at least $2k$. As w is crucial it is in the final solution. Moreover, as $d(w) \geq 2k$, the final locally minimal defensive alliance is of size at least k.

Lemma 5.3.2. Let G be a C_4-free graph with a minimum degree of at least 2. If $\Delta(G) \geq 4k^2 - 2k$ then G has a locally minimal defensive alliance of size at least k.

Proof. Let $d(u) \geq 4k^2 - 2k$. Run BFS(G, u) to obtain the BFS tree T rooted at u and to get level for each vertex in G. Run Algorithm 2 on $(G, V(G), u)$ and suppose it outputs D. If $u \in D$ then clearly D is a locally minimal defensive alliance of size at least $2k^2 - k$ as the degree of u is $4k^2 - 2k$ and u is protected in D. Consider the case where u is not in D. Let D' be the defensive alliance obtained at line 5 of Algorithm 2. As u is protected in D' at least $2k^2 - k$ of its neighbours are in $L_1 \cap D'$. All neighbors of u in $L_1 \cap D'$ are overprotected, this is why u is deleted from D' in line 6. Now let us focus on $G[L_1 \cap D']$. It may be noted that the maximum degree of a vertex in $G[L_1 \cap D']$ is at most 1 as G is C_4-free. As $\delta(G) \geq 2$, all isolated vertices in $G[L_1 \cap D']$ must have at least one neighbour in $L_2 \cap D'$. Every vertex from level 2 or higher that survives in the defensive alliance obtained by Algorithm 2 is crucial and it remains in the final locally minimal defensive alliance. Thus we assume there are at most $k-1$ vertices in $D' \cap L_2$, otherwise we have a solution of size at least k. Furthermore, every vertex in $D' \cap L_2$ has a degree at most $2k$, or else we have a solution of size at least k. This implies that there are at most $2k^2 - 2k$ vertices in $G[N(u) \cap D']$ which have neighbour in $L_2 \cap D'$. Note that there are still k vertices that have no neighbor in $L_2 \cap D'$; so they must have at least one neighbor in $L_1 \cap D'$. As $G[N(u) \cap D']$ has maximum degree one, we must have at least $\frac{k}{2}$ isolated edges in $G[D' \setminus \{u\}]$. As these $\frac{k}{2}$ isolated edges in $G[D' \setminus \{u\}]$ are connected defensive alliances such that the closed neighborhood of one defensive alliance does not intersect the other, we get a locally minimal defensive alliance of size at least k. □

5.3.1 Proof of Theorem 5.1.2

Proof. If $\Delta(G) \geq 4k^2$ or $diam(G) \geq \mathcal{O}(k)$ then we get a yes instance due to Lemma 5.3.1, Lemma 5.3.2 and the Lemma 5.4.4 . Therefore, we assume G has $\Delta(G) < 4k^2$ and $diam(G) < \mathcal{O}(k)$ which produces a kernel of size $k^{\mathcal{O}(k)}$. □

5.4 FPT algorithm for Locally Minimal Defensive Alliance on planar graphs

It is proved in [71] that the Locally Minimal Defensive Alliance is NP-complete in planar graphs via a reduction from Minimum Maximal Matching in a cubic planar graph. In this section, we design an FPT algorithm for Locally Minimal Defensive Alliance on planar graphs with $\delta(G) \geq 2$. We use a win/win approach to design an FPT algorithm for Locally Minimal Defensive Alliance on planar graphs. For $n \in \mathbb{N}$, by $[n]$ we denote the set $\{1, 2, \ldots, n\}$. For a positive integer t, a $t \times t$ *grid* $\boxplus_t$ is a graph with vertex set $\{(x, y) \;:\; x, y \in [t]\}$ and two different vertices (x, y) and (x', y') are adjacent if and only if $|x - x'| + |y - y'| = 1$. The triangulated grid Γ_t is obtained from the grid $\boxplus_t$ by adding the edges $(x + 1, y), (x, y + 1)$ for all $1 \leq x, y \leq t - 1$ and additionally making vertex (t, t) adjacent to the whole border of $\boxplus_t$. Theorem 5.4.1 gives the relationship between the treewidth and the size of a triangulated grid as a contraction.

Theorem 5.4.1. *[54]* **(Planar excluded grid theorem for edge contractions)**. *For every connected planar graph G and integer $t > 0$, if $tw(G) > 9t + 5$ then G contains Γ_t as a contraction. Furthermore, for every $\epsilon > 0$ there exists an $\mathcal{O}(n^2)$ algorithm that, given a connected planar n-vertex graph G and an integer t, either outputs a tree decomposition of G of width $(9 + \epsilon)t + 5$ or a set of edges whose contraction in G results in Γ_t.*

We now obtain the following result.

Lemma 5.4.2. Let G be a planar graph of minimum degree at least 2 which contains a triangulated grid $\Gamma_{10\sqrt{k}+4}$ as a minor obtained by only edge contraction operations. Then there always exists a locally minimal defensive alliance of size at least k.

Proof. The proof of Lemma 5.4.2 is similar to that of Lemma 5.2.8. Given that G can be transformed to Γ_t after a sequence of edge contractions. Suppose that the vertices of Γ_t are labelled (i, j) where $i, j \in [t]$. Consider the set $C' = \{(4x + 2, 4y + 2) \mid 0 \leq x, y \leq \frac{t-4}{5}\} \subset V(\Gamma_t)$. See Figure 5.4 for an illustration; the vertices of C' are shown in red. Every (red) vertex in Γ_t is either an original vertex of G or obtained by contracting some edges of G. We obtain a set $C \subseteq V(G)$ from C' as follows. For each (red) vertex (i, j) in C', if the red vertex is an original vertex, then include it in C; if the red vertex is obtained by contracting some edges of G, then arbitrarily include in C an endpoint of one of the contracted edges. Run Algorithm 2 on $(G, V(G), C)$. Let D_C be a defensive alliance obtained at the end of

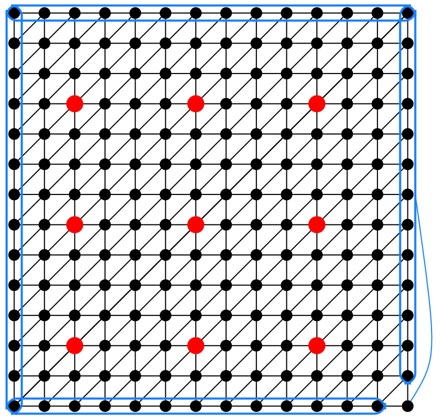


Figure 5.4: Example of a triangulated grid Γ_{14}. Note that a blue edge denotes that the vertex is adjacent to all the vertices inside the blue boundary. The set of vertices colored red forms C'.

the **while** loop of lines 1-4 in Algorithm 2. Note that D_C may not be a connected defensive alliance. Let $S_1, S_2, \ldots, S_p$ be connected components of D_C such that $\bigcup_{i=1}^{p} S_i = D_C$. This implies that D_C will generate a partition of C into $C_1, C_2, \ldots, C_p$ such that $C_i \subseteq S_i$.

Claim 5.4.1. *Line 6 of Algorithm 2 runs Algorithm 1 on (G, S_i) and outputs a locally minimal defensive alliance S'_i such that $|S'_i| \geq \max\{2, |C_i| - 1\}$ for all $1 \leq i \leq p$.*

Proof of Claim: As we have $\delta(G) \geq 2$, we have $|S'_i| \geq 2$. Therefore let us focus on C_i's such that $|C_i| \geq 4$. Let $C_i = \{c_1, c_2, \ldots, c_q\}$. Then $d_G(c_s, c_t) \geq 4$ for all $1 \leq s, t \leq q - 1$ and $s \neq t$ due to choice of C. Note that as $G[S_i]$ is connected, it implies that $G[S_i]$ contains at least $q - 1$ crucial vertices in the defensive alliance S_i. This is because any path between c_s and c_t in $G[S_i]$ contains a vertex u such that $d(u, C_i) \geq 2$ for all $1 \leq i \leq q - 1$. Due to Lemma 5.2.2, we have $|S'_i| \geq q - 1$. This completes the proof of the lemma.

If we set $t = 10\sqrt{k} + 4$ then $|C| \geq 2k$. It implies that $\bigcup_{i=1}^{p} S_i'$ is a locally minimal defensive alliance of size at least k. This completes the proof of the lemma. $\square$

To design a dynamic programming algorithm on a given tree decomposition of the input graph, we use the following theorem.

Theorem 5.4.3. *[71] Given an n-vertex graph G and its nice tree decomposition T of width at most w, the size of a maximum locally minimal defensive alliance of G can be computed in time $\mathcal{O}^*(18^w \Delta^{\mathcal{O}(w)})$.*

In the next lemma, we improve the bound given in Theorem 5.2.4, for graphs with minimum degree at least two. Note that the proof is similar to Lemma 5.2.8.

Lemma 5.4.4. *Let G be a graph such that $\delta(G) \geq 2$ and S be a connected defensive alliance of diameter at least $8k'$ where k' is any positive integer then there exists a locally minimal defensive alliance of size at least k'.*

Proof. As $diam(G[S]) \geq 8k'$, there exists a pair of vertices w_0 and $w_{8k'}$ such that $d(w_0, w_{8k'}) = 8k'$ in $G[S]$. Let $P =< w_0, w_1, \ldots, w_{8k'} >$ be a shortest w_0-$w_{8k'}$ path in $G[S]$. Set $C = \{w_{4i} \mid 0 \leq i \leq 2k'\}$. Run Algorithm 2 on (G, S, C). Let $S(C)$ be the defensive alliance obtained at the end of the **while** loop of lines 1-4 in Algorithm 2. Note that $S(C)$ may not be a connected defensive alliance. Suppose $S(C_1), S(C_2), \ldots, S(C_p)$ are the connected components of $S(C)$ which generates a partition of C into p parts $C_1, C_2, \ldots, C_p$ such that $C_j \subseteq S(C_j)$ for all $j \in [p]$. Line 5 of Algorithm 2 executes Algorithm 1 on $(G, S(C))$, that is, it executes Algorithm 1 on $(G, S(C_j))$ for all $j \in [p]$.

Claim 5.4.2. *Line 5 of Algorithm 2 runs Algorithm 1 on $(G, S(C_j))$ and outputs a connected locally minimal defensive alliance S_j such that $|S_j| \geq \max\{2, |C_j| - 1\}$ for all $j \in [p]$.*

Proof. As we have $\delta(G) \geq 2$, any (locally minimal) defensive alliance in G must be of size at least 2. Therefore, we get $|S_j| \geq 2$. We now prove that Algorithm 1 on input instance $(G, S(C_j))$ returns a connected locally minimal defensive alliance S_j of size at least $|C_j| - 1$. Suppose $C_j = \{c_1, \ldots, c_q\}$, $q \geq 2$. We know $d(c_l, c_{l+1}) \geq 4$ and $d(w_0, c_l) < d(w_0, c_{l+1})$ in $G[S(C_j)]$ for all $1 \leq l \leq q-1$. As $G[S(C_j)]$ is connected, a shortest path between c_l and c_{l+1} in $G[S(C_j)]$ contain a vertex, say x, such that $x \in S(C_j) \setminus N[C_j]$. By Lemma 5.2.3, x is a crucial vertex in $S(C_j)$. As C_j has q vertices, we have at least $q-1$ crucial vertices in $S(C_j)$.

By Lemma 5.2.2, the locally minimal defensive alliance S_j returned by line 4 of Algorithm 2 contains all $q-1$ crucial vertices of S_j. Thus we have $|S_j| \geq q-1$. This completes the proof of the claim.

As the closed neighbourhood of one S_j does not intersect others and $|C| \geq 2k'$, we get $\bigcup_{j=1}^{p} S_j$ is a locally minimal defensive alliance of size at least k'. This completes the proof of the lemma. □

Corollary 5.4.5. Let l and k be positive integers and $S_1, S_2, \ldots, S_l$ be a set of connected defensive alliance in G with $\delta(G) \geq 2$ such that $N[S_i] \cap S_j = \emptyset$ for all $i \neq j$ and $i, j \in [l]$. If $8.(d_i+1) > diam(G[S_i]) \geq 8.d_i$ for some integer d_i such that $\sum_{i=1}^{l} d_i \geq k$ then G has a locally minimal defensive alliance of size at least k. Also, such a locally minimal defensive alliance can be obtained in polynomial time.

Proof. Due to Lemma 5.4.4, given a defensive alliance S_i one can obtain a locally minimal defensive alliance of size at least d_i for $d_i \geq 1$. Note that when $d_i = 0$, one can simply apply algorithm 1 on S_i to get a locally minimal defensive alliance of size at least 2. It is easy to see that the union of such locally minimal defensive alliances is also a locally minimal defensive alliance as $N[S_i] \cap S_j = \emptyset$ for all $i \neq j$ and $i, j \in [l]$. Since $\sum_{i=1}^{l} d_i \geq k$, we get a locally minimal defensive alliance of size at least k. □

Finally, to design an FPT algorithm when parameterized by the solution size k, we prove the following lemma.

Lemma 5.4.6. Let G be a planar graph with a minimum degree of at least 2. If $\Delta(G) \geq ck^5$ where c is a sufficiently large constant then there exists a locally minimal defensive alliance of size at least k.

Proof. Let v be any vertex such that $d(v) \geq ck^5$. Run BFS(G, v) to obtain the BFS tree T rooted at v and to get the level for each vertex in G. We will first run the Algorithm 2 on $(G, V(G), v)$ and suppose it outputs D_0. If $v \in D_0$ then at least $\frac{d(v)}{2} > k$ neighbours of v are in D_0 for its protection. Therefore D_0 is a locally minimal defensive alliance of size at least k. Consider the case v is not in D_0. Let D_1 be the defensive alliance obtained at the end of

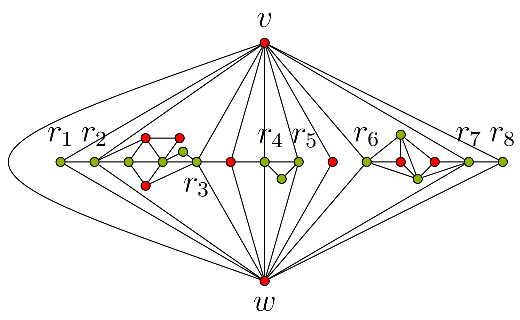


Figure 5.5: A planar drawing of the vertices in $(D \cap L_1) \cup \{v\}$. The vertices colored green are inside the defensive alliance, and the vertices colored red are outside the defensive alliance.

the **while** loop of lines 1-4 in Algorithm 2 and all neighbors of v in D_1 are overprotected. This is why v is deleted from D_0. As we have discussed before, the vertices in $D_1 \cap L_{\geq 2}$ are crucial. Therefore if $D_1 \cap L_{\geq 2}$ contains more than $k-1$ vertices or any of them has a degree more than $2k$, then we have a locally minimal defensive alliance of size at least k. So we assume $|D_1 \cap L_{\geq 2}| \leq k-1$ and every vertex in $D_1 \cap L_{\geq 2}$ has degree less than $2k$. Now we consider the defensive alliance $D = D_1 \setminus \{v\}$. As v was protected in D_1, $D \cap L_1$ contains at least $\lfloor \frac{ck^5}{2} \rfloor$ vertices.

Case 1. Let us assume that there exists a vertex w in $D \cap L_1$ such that it has at least $34k$ neighbours in $D \cap L_1$. In this case, we run Algorithm 2 on $(G, D, \{w\})$ and suppose it outputs D_2. If $w \in D_2$ then at least $\frac{d(w)}{2} > k$ neighbours of w are in D_2 for its protection. Therefore D_2 is a locally minimal defensive alliance of size at least k. Consider the case w is not in D_2. Let D_3 be the defensive alliance obtained at the end of the **while** loop of lines 1-4 in Algorithm 2, and all neighbors of w are overprotected in D_3. This is why w is deleted from D_3. Let us denote $D_4 = D_3 \setminus \{w\}$. Since w was protected in D_3 and it had at most $k-1$ neighbours in $D \cap L_{\geq 2}$, D_4 must contain at least $16k$ vertices from set $N(w) \cap N(v)$. As D_4 may not be a connected defensive alliance, let us assume that $D_4 = \bigcup\limits_{i=1}^{l} D_{4i}$ where $D_{41}, D_{42}, \ldots, D_{4l}$ be a set of connected defensive alliance in G such that $N[D_{4i}] \cap D_{4j} = \emptyset$ for all $i \neq j$ and $i, j \in [l]$. Due to planarity of the graph, one can observe that $diam(G[D_{4i}]) \geq \max\{1, |D_{4i} \cap N(w) \cap N(v)| - 1\}$. See Figure 5.5 for an illustration. This shows that $\sum\limits_{i=1}^{l} diam(G[D_{4i}]) \geq 8k$. Due to Corollary 5.4.5, a locally minimal defensive alliance of size at least k exists.

Case 2. We assume that every vertex in $D \cap L_1$ has at most $34k$ neighbors in the set $D \cap L_1$. As $|D \cap L_{\geq 2}| \leq k-1$, the maximum degree of graph $G[D]$ is at most $35k$. As D contains at least $\frac{ck^5}{2}$ vertices and $\Delta(G[D]) \leq 35k$, one can greedily construct a 4-scattered set C in a graph $G[D]$ of size at least $c'k^2$ for a large constant c', assuming that c is sufficiently large. Next run Algorithm 2 on $(G, V(G), C)$. Let D_C be a defensive alliance obtained at the end of the **while** loop of lines 1-4 in Algorithm 2. Note that D_C be a defensive alliance. Let $S_1, S_2, \ldots, S_p$ be connected component of D_C such that $\bigcup\limits_{i=1}^{p} S_i = D_C$. This implies that D_C will generate a partition of C into $C_1, C_2, \ldots, C_p$ such that $C_i \subseteq S_i$.

Claim 5.4.3. *Line 6 of Algorithm 2 runs Algorithm 1 on (G, S_i) and outputs a locally minimal defensive alliance S'_i such that $|S'_i| \geq \max\left\{2, \frac{|C_i|}{35k}\right\}$ for all $1 \leq i \leq p$.*

Proof. Clearly, as minimum degree of G is at least 2 implies that $|S'_i| \geq 2$. Therefore let us focus on C_i's such that $|C_i| \geq 70k$. Let $C_i = \{c_1, c_2, \ldots, c_q\}$. We have $d(c_{i'}, c_{j'}) \geq 4$ for all $1 \geq i', j' \geq q$ and $i' \neq j'$ in $G[S_i]$ as C_i is a 4-scattered set in $G[S_i]$. All the vertices in $S_i \setminus N[C_i]$ are crucial in S_i due to Lemma 5.2.3. To get a lower bound on the number of crucial vertices, we make the following observation. Let us consider the subgraphs $G[N[C_i]]$ as an induced subgraph of graph $G[S_i]$. Clearly the induced subgraph $G[N[C_i]]$ have exactly q many connected components. For any vertex subset $\{x_1, x_2, \ldots, x_l\} \subseteq S_i \setminus N[C_i]$, the induced subgraph $G[N[C_i] \cup \{x_1, x_2, \ldots, x_l\}]$ will contain at least $\max\{1, q - (35k)l\}$ connected components as $\Delta(G[S_i]) \leq 35k$. As we know that $G[S_i]$ is connected, we must have $|S_i \setminus N[C_i]| \geq \frac{q}{35k}$ which is essentially the number of crucial vertices in the defensive alliance S_i. Therefore we get that $|S'_i| \geq \frac{|C_i|}{35k}$. This proves the claim.

Due to the claim above, we see that $\sum\limits_{i=1}^{p} |S'_i| \geq \sum\limits_{i=1}^{p} \frac{c'k^2}{35k} \geq k$ as c' is a large constant. Also, it is easy to see that $\bigcup\limits_{i=1}^{p} S'_i$ is a locally minimal defensive alliance. □

5.5 Locally Minimal Defensive Alliance Parameterized by Treewidth

This section presents an XP-algorithm for Locally Minimal Defensive Alliance problem parameterized by treewidth. We prove the following theorem:

Theorem 5.5.1. *Given an n-vertex graph G and its nice tree decomposition T of width at most k, the size of a maximum locally minimal defensive alliance of G can be computed in $O(18^k n^{4k+10})$ time.*

Let $(T, \{X_t\}_{t \in V(T)})$ be a nice tree decomposition rooted at node r of the input graph G. For a node t of T, let V_t be the union of all bags present in the subtree of T rooted at t, including X_t. We denote by G_t the subgraph of G induced by V_t. Here we distinguish not only if a vertex is in the solution or not, but if it is in the solution we also distinguish if it is marginally protected or not. A *coloring* of bag X_t is a mapping $f : X_t \to \{b, w, r\}$ assigning three different colours to vertices of the bag. We give intuition behind the three colours.

- **White**, represented by w. The meaning is that all white vertices have to be contained in the partial solution in G_t.
- **Black**, represented by b. The meaning is that all black vertices have to be contained in the partial solution in G_t; additionally, all black vertices must be marginally protected in the final solution.
- **Red**, represented by r. The meaning is that all red vertices are not contained in the partial solution in G_t.

For a node t, there are $3^{|X_t|}$ colourings X_t. Now, for each node t in T, we construct a table $dp_t(f, \mathbf{p}, \mathbf{a}, \mathbf{v}, \alpha, \pi, \beta, \beta^*, \gamma) \in \{\text{true, false}\}$ where f is a colouring of X_t, $\mathbf{p}$ is a vector of length n such that

$$\mathbf{p}(i) = \begin{cases} 0 \text{ or } 1 & \text{if } v_i \in X_t \text{ and } f(v_i) \in \{b, w\} \\ \star & \text{otherwise;} \end{cases}$$

$\mathbf{a}$ and $\mathbf{v}$ are vectors of length n, and their ith coordinates are positive only if v_i is in X_t and it is coloured b or w; $\alpha, \pi, \beta, \beta^*$ and γ are integers between 0 to n. We set $dp_t(f, \mathbf{p}, \mathbf{a}, \mathbf{v}, \alpha, \pi, \beta, \beta^*, \gamma) = $ true if and only if there exists a set $A_t \subseteq V_t$ such that

1. $\alpha = |A_t| = |\{v \in V_t : f(v) \in \{b, w\}\}|$

2. $f^{-1}\{b, w\} = A_t \cap X_t = A$, which is the set of vertices of X_t colored black or white.

3. the ith coordinate of vector $\mathbf{p}$ is

$$\mathbf{p}(i) = \begin{cases} 1 & \text{if } v_i \in X_t,\ f(v_i) \in \{b, w\} \text{ and } v_i \text{ has a black neighbour in } A_t \\ 0 & \text{if } v_i \in X_t,\ f(v_i) \in \{b, w\} \text{ and } v_i \text{ has no black neighbours in } A_t \\ \star & \text{otherwise} \end{cases}$$

4. the ith coordinate of vector $\mathbf{a}$ is

$$\mathbf{a}(i) = \begin{cases} d_{A_t}(v_i) & \text{if } v_i \in X_t \text{ and } f(v_i) \in \{b, w\} \\ 0 & \text{otherwise} \end{cases}$$

   That is, $\mathbf{a}(i)$ denotes the number of neighbours of vertex v_i in A_t if $v_i \in X_t$ and $f(v_i) \in \{b, w\}$.

5. the ith coordinate of vector $\mathbf{v}$ is

$$\mathbf{v}(i) = \begin{cases} d_{V_t}(v_i) & \text{if } v_i \in X_t \text{ and } f(v_i) \in \{b, w\} \\ 0 & \text{otherwise.} \end{cases}$$

   That is, $\mathbf{v}(i)$ denotes the number of neighbours of vertex v_i in V_t if $v_i \in X_t$ and $f(v_i) \in \{b, w\}$.

6. π is the number of vertices $v \in A_t$ that are protected, that is, $d_{A_t}(v) \geq \frac{d_G(v)-1}{2}$.

7. β is the number of black vertices in A_t.

8. β^* is the number of black vertices v in A_t such that $N(v) \subseteq V_t$ and $d_{A_t}(v) = \lceil \frac{d_G(v)-1}{2} \rceil$. Thus β^* is the number of black vertices v in A_t that are marginally protected when all its neighbours are introduced in G_t. The intuition here is that we want every back vertex to be marginally protected when all its neighbours are introduced.

9. γ is the number of vertices in A_t who has a black neighbour. In other words, γ is the number of *good* vertices in A_t.

We compute all entries $dp_t(f, \mathbf{p}, \mathbf{a}, \mathbf{v}, \alpha, \pi, \beta, \beta^*, \gamma)$ in a bottom-up manner. Since $tw(T) \leq k$, there are $O(3^k \cdot 2^k \cdot n^k \cdot n^k \cdot (n+1)^5) = O(6^k n^{2k+5})$ possible tuples $(f, \mathbf{p}, \mathbf{a}, \mathbf{v}, \alpha, \pi, \beta, \beta^*, \gamma)$. Thus, to prove Theorem 5.5.1, it suffices to show that each entry $dp_t(f, \mathbf{p}, \mathbf{a}, \mathbf{v}, \alpha, \pi, \beta, \beta^*, \gamma)$ can be computed in $O(3^k n^{2k+5})$ time, assuming that the entries for the children of t are already computed.

Lemma 5.5.2. *For a leaf node t, $dp_t(f, \mathbf{p}, \mathbf{a}, \mathbf{v}, \alpha, \pi, \beta, \beta^*, \gamma)$ can be computed in O(1) time.*

Proof. For leaf node t we have that $X_t = \emptyset$. Thus $dp_t(f, \mathbf{p}, \mathbf{a}, \mathbf{v}, \alpha, \pi, \beta, \beta^*, \gamma)$ is true if and only if $f = \emptyset$, $\mathbf{p} = \mathbf{0}$, $\mathbf{a} = \mathbf{0}$, $\mathbf{v} = \mathbf{0}$, $\alpha = 0$, $\pi = 0$, $\beta = 0$, $\beta^* = 0$ and $\gamma = 0$. These conditions can be checked in $O(1)$ time. □

Lemma 5.5.3. *For an introduce node t, $dp_t(f, \mathbf{p}, \mathbf{a}, \mathbf{v}, \alpha, \pi, \beta, \beta^*, \gamma)$ can be computed in O(1) time.*

Proof. Suppose t is an introduce node with child t' such that $X_t = X_{t'} \cup \{v_i\}$ for some $v_i \notin X_{t'}$. Let f be any coloring of X_t. We consider three cases:

Case (i): Let $f(v_i) = r$. In this case $dp_t(f, \mathbf{p}, \mathbf{a}, \mathbf{v}, \alpha, \pi, \beta, \beta^*, \gamma)$ is true if and only if $dp_{t'}(f|_{X_{t'}}, \mathbf{p}, \mathbf{a}, \mathbf{v}', \alpha, \pi, \beta, \beta^*, \gamma)$ is true where

$$\mathbf{v}(j) = \begin{cases} \mathbf{v}'(j) + 1 & \text{if } j \neq i,\ v_j \in X_t,\ f(v_j) \in \{b, w\} \text{ and } v_j \in N_{X_t}(v_i) \\ \mathbf{v}'(j) & \text{otherwise} \end{cases}$$

Case (ii): Let $f(v_i) = b$. Here $dp_t(f, \mathbf{p}, \mathbf{a}, \mathbf{v}, \alpha, \pi, \beta, \beta^*, \gamma)$ is true if and only if there exist a tuple $(f', \mathbf{p}', \mathbf{a}', \mathbf{v}', \alpha', \pi', \beta', \beta^{*'}, \gamma')$ such that $dp_{t'}(f', \mathbf{p}', \mathbf{a}', \mathbf{v}', \alpha', \pi', \beta', \beta^{*'}, \gamma')$=true, where

1. $f|_{X_t \setminus \{v_i\}} = f'|_{X_{t'}}$;

2. $$\mathbf{p}(j) = \begin{cases} 1 & \text{if } v_j \in X_t,\ f(v_j) \in \{b, w\} \text{ and } v_j \in N(v_i) \\ \mathbf{p}'(j) & \text{otherwise} \end{cases}$$

3.

$$\mathbf{a}(j) = \begin{cases} \mathbf{a}'(j)+1 & \text{if } j \neq i,\ v_j \in X_t,\ f(v_j) \in \{b,w\} \text{ and } v_j \in N_{X_t}(v_i) \\ |N_A(v_i)| & \text{if } j = i \\ \mathbf{a}'(j) & \text{otherwise} \end{cases}$$

where $A = A_t \cap X_t$.

4.

$$\mathbf{v}(j) = \begin{cases} \mathbf{v}'(j)+1 & \text{if } j \neq i,\ v_j \in X_t,\ f(v_j) \in \{b,w\} \text{ and } v_j \in N_{X_t}(v_i) \\ |N_{X_t}(v_i)| & \text{if } j = i \\ \mathbf{v}'(j) & \text{otherwise} \end{cases}$$

5. $\alpha = \alpha' + 1$;

6. $\pi = \pi' + l$; here l is the cardinality of the set

$$\Big\{v_j \in X_t \mid f(v_j) \in \{b,w\}, \mathbf{a}'(j) < \frac{d_G(v_j)-1}{2}; \mathbf{a}(j) \geq \frac{d_G(v_j)-1}{2}\Big\}.$$

That is, to compute π from π' we need to add the number l of vertices $v_j \in X_t$ which are not protected in $X_{t'}$ but protected in X_t.

7. $\beta = \beta' + 1$;

8. $\beta^* = \beta^{*'} + \delta$
where δ is the number of black vertices $v_j \in X_t$ such that $\mathbf{a}'(j) \neq \lceil \frac{d_G(v_j)-1}{2} \rceil$ or $\mathbf{v}'(j) \neq d(v_j)$ but it satisfies the conditions $\mathbf{a}(j) = \lceil \frac{d_G(v_j)-1}{2} \rceil$ and $\mathbf{v}(j) = d(v_j)$.

9. $\gamma = \gamma' + |\{v_j \in A \mid \mathbf{p}(j) = 1 \text{ but } \mathbf{p}'(j) = 0\}|$.

Case (iii): Let $f(v_i) = w$. Here $dp_t(f, \mathbf{p}, \mathbf{a}, \mathbf{v}, \alpha, \pi, \beta, \beta^*, \gamma)$ is true if and only if there exist a tuple $(f', \mathbf{p}', \mathbf{a}', \mathbf{v}', \alpha', \pi', \beta', \beta^{*'}, \gamma')$ such that $dp_{t'}(f', \mathbf{p}', \mathbf{a}', \mathbf{v}', \alpha', \pi', \beta', \beta^{*'}, \gamma')$=true, where

1. $f|_{X_t \setminus \{v_i\}} = f'$;

2.

$$\mathbf{p}(j) = \begin{cases} \mathbf{p}'(j) & \text{if } j \neq i \\ 1 & \text{if } j = i \text{ and } v_i \text{ has a black neighbour in } X_t \\ 0 & \text{if } j = i \text{ and } v_i \text{ has no black neighbours in } X_t \end{cases}$$

3. $$\mathbf{a}(j) = \begin{cases} \mathbf{a}'(j)+1 & \text{if } j \neq i,\ v_j \in X_t,\ f(v_j) \in \{b,w\} \text{ and } v_j \in N_{X_t}(v_i) \\ |N_A(v_i)| & \text{if } j = i \\ \mathbf{a}'(j) & \text{otherwise} \end{cases}$$
   where $A = A_t \cap X_t$.

4. $$\mathbf{v}(j) = \begin{cases} \mathbf{v}'(j)+1 & \text{if } j \neq i,\ v_j \in X_t,\ f(v_j) \in \{b,w\} \text{ and } v_j \in N_{X_t}(v_i) \\ |N_{X_t}(v_i)| & \text{if } j = i \\ \mathbf{v}'(j) & \text{otherwise} \end{cases}$$

5. $\alpha = \alpha' + 1$;

6. $\pi = \pi' + l$; here l is the cardinality of the set
   $$\Big\{v_j \in X_t \mid f(v_j) \in \{b,w\}, \mathbf{a}'(j) < \frac{d_G(v_j)-1}{2}; \mathbf{a}(j) \geq \frac{d_G(v_j)-1}{2}\Big\}.$$
   That is, to compute π from π' we need to add the number l of vertices $v_j \in X_t$ which are not protected in $X_{t'}$ but protected in X_t.

7. $\beta = \beta'$;

8. $\beta^* = \beta^{*'} + \delta$
   where δ is the number of black vertices $v_j \in X_t$ such that $\mathbf{a}'(j) \neq \lceil\frac{d_G(v_j)-1}{2}\rceil$ or $\mathbf{v}'(j) \neq d(v_j)$ but it satisfies the conditions $\mathbf{a}(j) = \lceil\frac{d_G(v_j)-1}{2}\rceil$ and $\mathbf{v}(j) = d(v_j)$.

9. $\gamma = \gamma' + 1$ if v_i is adjacent to a vertex in X_t which is coloured black; otherwise $\gamma = \gamma'$.

For introduce node t, $dp_t(f, \mathbf{p}, \mathbf{a}, \mathbf{v}, \alpha, \pi, \beta, \beta^*, \gamma)$ can be computed in $O(1)$ time as there is only one candidate of such tuple $(f', \mathbf{p}', \mathbf{a}', \mathbf{v}', \alpha', \pi', \beta', \beta^{*'}, \gamma')$. □

Lemma 5.5.4. *For a forget node t, $dp_t(f, \mathbf{p}, \mathbf{a}, \mathbf{v}, \alpha, \pi, \beta, \beta^*, \gamma)$ can be computed in $O(n)$ time.*

Proof. Suppose t is a forget node with child t' such that $X_t = X_{t'} \setminus \{v_i\}$ for some $v_i \in X_{t'}$. Here $dp_t(f, \mathbf{p}, \mathbf{a}, \mathbf{v}, \alpha, \pi, \beta, \beta^*, \gamma)$ is true if and only if $dp_{t'}(f', \mathbf{p}', \mathbf{a}', \mathbf{v}', \alpha', \pi', \beta', \beta^{*'}, \gamma')$ is true, where

1. $f' = f_{v_i \to b}, f_{v_i \to w}$ or $f_{v_i \to r}$.

2. $$\mathbf{p}(j) = \begin{cases} \mathbf{p}'(j) & \text{if } j \neq i \\ \star & \text{if } j = i \end{cases}$$

3. $\mathbf{a}(j) = \mathbf{a}'(j)$ for all $j \neq i$ and $\mathbf{a}(i) = 0$;

4. $\mathbf{v}(j) = \mathbf{v}'(j)$ for all $j \neq i$ and $\mathbf{v}(i) = 0$;

5. $\alpha = \alpha'$;

6. $\pi = \pi'$;

7. $\beta = \beta'$;

8. $\beta^* = {\beta^*}'$;

9. $\gamma = \gamma'$.

There are $n+1$ choices for $\mathbf{a}'(i)$ and $\mathbf{v}'(i)$ each. Thus the lemma follows as there are $O(n)$ candidates of such tuples $(f', \mathbf{p}', \mathbf{a}', \mathbf{v}', \alpha', \pi', \beta', {\beta^*}', \gamma')$. This completes the proof of the lemma. □

Lemma 5.5.5. *For a join node t, $dp_t(f, \mathbf{p}, \mathbf{a}, \mathbf{v}, \alpha, \pi, \beta, \beta^*, \gamma)$ can be computed in $O(3^k n^{3k+5})$ time.*

Proof. Suppose t is a join node with children t_1 and t_2 such that $X_t = X_{t_1} = X_{t_2}$. Then $dp_t(f, \mathbf{p}, \mathbf{a}, \mathbf{v}, \alpha, \pi, \beta, \beta^*, \gamma)$ is true if and only if there exist $(f_1, \mathbf{p}_1, \mathbf{a}_1, \mathbf{v}_1, \alpha_1, \pi_1, \beta_1, \beta_1^*, \gamma_1)$ and $(f_2, \mathbf{p}_2, \mathbf{a}_2, \mathbf{v}_2, \alpha_2, \pi_2, \beta_2, \beta_2^*, \gamma_2)$ such that $dp_{t_1}(f_1, \mathbf{p}_1, \mathbf{a}_1, \mathbf{v}_1, \alpha_1, \pi_1, \beta_1, \beta_1^*, \gamma_1)$ and $dp_{t_2}(f_2, \mathbf{p}_2, \mathbf{a}_2, \mathbf{v}_2, \alpha_2, \pi_2, \beta_2$
are true, where

1. $f = f_1 = f_2$;

2. $\mathbf{p}(i) = 1$ if $\mathbf{p}_1(i) = 1$ or $\mathbf{p}_2(i) = 1$;

3. $\mathbf{a}(i) = \mathbf{a}_1(i) + \mathbf{a}_2(i) - d_A(v_i)$ for all $v_i \in A$, and $\mathbf{a}(i) = 0$ if $v_i \notin A$ where $A = \{v \in X_t \mid f(v) \in \{b, w\}\}$;

4. $\mathbf{v}(i) = \mathbf{v}_1(i) + \mathbf{v}_2(i) - d_{X_t}(v_i)$ for all $v_i \in A$, and $\mathbf{v}(i) = 0$ if $v_i \notin A$;

5. $\alpha = \alpha_1 + \alpha_2 - |A|$;

6. $\pi = \pi_1 + \pi_2 - l_1 + l_2$;
   where l_1 is the cardinality of the set
   $$\Big\{v_j \in A \mid \mathbf{a}_1(j) \geq \frac{d_G(v_i) - 1}{2};\ \mathbf{a}_2(j) \geq \frac{d_G(v_i) - 1}{2}\Big\}$$
   and l_2 is the cardinality of the set
   $$\Big\{v_j \in A \mid \mathbf{a}_1(j) < \frac{d_G(v_i) - 1}{2};\ \mathbf{a}_2(j) < \frac{d_G(v_i) - 1}{2};\ \mathbf{a}(j) \geq \frac{d_G(v_i) - 1}{2}\Big\}.$$
   To compute π from $\pi_1 + \pi_2$, we need to subtract the number of those v_j which are protected in both the branches and add the number of vertices v_j which are not protected in either of the branches t_1 and t_2 but protected in t.

7. $\beta = \beta_1 + \beta_2 - |\{v \in A \mid f(v) = b\}|$;

8. $\beta^* = \beta_1^* + \beta_2^* + \delta_1 + \delta_2 - \delta_{12}$.
   Here δ_1 is the number of black vertices v_j in X_t such that $\mathbf{a}_1(j) \neq \lceil \frac{d_G(v_j)-1}{2} \rceil$ or $\mathbf{v}_1(j) \neq d(v_j)$ but it satisfies the conditions $\mathbf{a}(j) = \lceil \frac{d_G(v_j)-1}{2} \rceil$ and $\mathbf{v}(j) = d(v_j)$. Similarly, δ_2 is the number of black vertices v_j in X_t such that either $\mathbf{a}_2(j) \neq \lceil \frac{d_G(v_j)-1}{2} \rceil$ or $\mathbf{v}_2(j) \neq d(v_j)$ but it satisfies the conditions $\mathbf{a}(j) = \lceil \frac{d_G(v_j)-1}{2} \rceil$ and $\mathbf{v}(j) = d(v_j)$. Finally δ_{12} is the number of black vertices v_j in X_t such that $\mathbf{a}_1(j) \neq \lceil \frac{d_G(v_j)-1}{2} \rceil$ or $\mathbf{v}_1(j) \neq d(v_j)$, and $\mathbf{a}_2(j) \neq \lceil \frac{d_G(v_j)-1}{2} \rceil$ or $\mathbf{v}_2(j) \neq d(v_j)$ but it satisfies the conditions $\mathbf{a}(j) = \lceil \frac{d_G(v_j)-1}{2} \rceil$ and $\mathbf{v}(j) = d(v_j)$.

9. $\gamma = \gamma_1 + \gamma_2 - |\{v \in A \mid \mathbf{p}_1(v) = \mathbf{p}_2(v) = 1\}|$.

For join node t, there are at most 3^k possible pairs for $(\mathbf{p_1}, \mathbf{p_2})$ as $(\mathbf{p_1}(i), \mathbf{p_2}(i)) \in \{(1,0), (0,1), (1,1)\}$ when $\mathbf{p}(i) = 1$ and $(\mathbf{p_1}(i), \mathbf{p_2}(i)) = (0,0)$ when $\mathbf{p}(i) = 0$; there are n^k possible pairs for $(\mathbf{a_1}, \mathbf{a_2})$ as $\mathbf{a_2}$ is uniquely determined by $\mathbf{a_1}$; there are n^k possible pairs for $(\mathbf{v_1}, \mathbf{v_2})$ as $\mathbf{v_2}$ is uniquely determined by $\mathbf{v_1}$; $n+1$ possible pairs for (α_1, α_2); $n+1$ possible pairs for (π_1, π_2); $n+1$ possible pairs for (β_1, β_2); $n+1$ possible pairs for (β_1^*, β_2^*); and $n+1$ possible pairs for (γ_1, γ_2). In total, there are $O(3^k n^{2k+5})$ candidates, and each of them can be checked in $O(1)$ time. Thus, for join node t, $dp_t(f, \mathbf{p}, \mathbf{a}, \mathbf{v}, \alpha, \pi, \beta, \beta^*, \gamma)$ can be computed in $O(3^k n^{2k+5})$ time. $\square$

At the root node r, we look at all records such that $dp_r(\emptyset, \emptyset, \emptyset, \emptyset, \alpha, \pi, \beta, \beta^*, \gamma)=$ true, $\beta = \beta^*$ (that is, all black vertices in the solution are marginally protected) and $\alpha = \pi = \gamma$ (that is, every vertex in the solution is protected and has a black or marginally protected neighbour). The size of a maximum locally minimal defensive alliance is the maximum α satisfying $dp_r(\emptyset, \emptyset, \emptyset, \emptyset, \alpha, \pi, \beta, \beta^*, \gamma)=$ true, $\alpha = \pi = \gamma$ and $\beta = \beta^*$.

Remark. The above algorithm implies that Locally Minimal Defensive Alliance can be solved in polynomial time on trees.

5.6 Closing Remarks and Future Directions

We showed that Locally Minimal Defensive Alliance is FPT when parameterized by solution size. On the C_3-free and C_4-free graphs with minimum degree at least 2, we have provided a kernel of size $k^{\mathcal{O}(k)}$. When restricted to planar graphs with minimum degree at least 2, we have designed an algorithm with running time $k^{\mathcal{O}(\sqrt{k})} n^{\mathcal{O}(1)}$. It remains open whether Locally Minimal Defensive Alliance admits a kernel of polynomial size. Also it would be interesting to know if our ideas can be used to decide if Globally Minimal Defensive Alliance is FPT when parameterized by solution size.

Chapter 6

Globally Minimal Defensive Alliance

A *defensive alliance* in an undirected graph $G = (V, E)$ is a non-empty set S of vertices satisfying the condition that every vertex $v \in S$ has at least as many neighbours (including itself) in S as it has in $V \setminus S$. We consider the notion of global minimality in this chapter. A defensive alliance S is called a *globally minimal defensive alliance* if no proper subset is a defensive alliance. Given an undirected graph G and a positive integer k, we study Globally Minimal Defensive Alliance, where the goal is to check whether G has a globally minimal defensive alliance of size at least k. This problem is NP-hard but its parameterized complexity remains open until now. The goal of this chapter is to provide new insights into the complexity of Globally Minimal Defensive Alliance parameterized by the structure of the input graph. We show that the problem is FPT parameterized by the neighbourhood diversity of the input graph. The result for neighborhood diversity implies that the problem is FPT parameterized by vertex cover number also. We prove that the problem parameterized by the vertex cover number of the input graph does not admit a polynomial compression unless coNP $\subseteq$ NP/poly. We show that the problem is W[1]-hard parameterized by a wide range of fairly restrictive structural parameters such as the feedback vertex set number, pathwidth, treewidth and treedepth. We also prove that, given a vertex $r \in V(G)$, deciding if G has a globally minimal defensive alliance of any size containing vertex r is NP-complete. This chapter is based on the paper [63].

6.1 FPT algorithm parameterized by neighbourhood diversity

In this section, we show that when parameterized by neighbouhood diversity, the problem of finding a largest globally minimal defensive alliance is fixed-parameter tractable. In this section, we prove the following theorem:

Theorem 6.1.1. Globally Minimal Defensive Alliance *can be solved in time* $k^{O(k^2)} \cdot n^{O(1)}$ *where* k *is the neighbourhood diversity of the input graph.*

We reduce the problem of finding a largest globally minimal defensive alliance to an integer linear programming optimization with k variables. Since integer linear programming is fixed parameter tractable when parameterized by the number of variables [99], we conclude that our problem is FPT when parameterized by the neighbourhood diversity k.

6.1.1 ILP formulation

Let G be a connected graph such that $\mathtt{nd}(G) = k$. In this section we assume that we have the partition of $V(G)$ into sets of type classes $C_1, \ldots, C_k$. We assume $k \geq 2$ since otherwise the problem becomes trivial. We prove the following lemma.

Lemma 6.1.2. *Suppose* $S_1, S_2 \subseteq V(G)$ *are such that* $|S_1 \cap C_i| = |S_2 \cap C_i|$ *for all* $i \in [k]$. *If* S_1 *is a GMDA in* G *then* S_2 *is also a GMDA in* G.

Proof. Suppose S_1 is a globally minimal defensive alliance in G. For each $i \in [k]$, the vertices in $S_1 \cap C_i$ and $S_2 \cap C_i$ have the same neighbourhood in G as the vertices in C_i have the same neighbourhood in G. Therefore S_2 is also a globally minimal defensive alliance in G. Similarly, given S_2 is a globally minimal defensive alliance in G, it is easy to prove that S_1 is also a globally minimal defensive alliance in G. □

Let $x_i = |C_i \cap S|$ where S is a globally minimal defensive alliance. We define the following sets:

$$\mathcal{C}_0 = \{C_i \mid x_i = 0\};\ I_0 = \{i \mid C_i \in \mathcal{C}_0\}$$

$$\mathcal{C}_1 = \{C_i \mid x_i = 1\};\ I_1 = \{i \mid C_i \in \mathcal{C}_1\}$$

$$\mathcal{C}_{\geq 2} = \{C_i \mid x_i \geq 2\};\quad I_{\geq 2} = \{i \mid C_i \in \mathcal{C}_{\geq 2}\}$$

Our goal here is to find a largest globally minimal defensive alliance S of G, with $C_i \cap S = \emptyset$ when $C_i \in \mathcal{C}_0$, $|C_i \cap S| = 1$ when $C_i \in \mathcal{C}_1$ and $|C_i \cap S| \geq 2$ when $C_i \in \mathcal{C}_{\geq 2}$ where $\mathcal{C}_0$, $\mathcal{C}_1$ and $\mathcal{C}_{\geq 2}$ are given. By Lemma 6.1.2, the variables x_i determine S uniquely. The objective is to maximize the sum

$$|I_1| + \sum_{i \in [k]} x_i [C_i \in \mathcal{C}_{\geq 2}]$$

under the condition $2 \leq x_i \leq |C_i| = n_i$ for all $i \in [k]$ and the additional conditions (Type 1 and Type 2) described below. Here $[C_i \in \mathcal{C}_{\geq 2}]$ denotes the Iverson bracket.

Type 1 Condition: For each $i \in [k]$ we add the condition given in Equation 1. This is called type 1 condition. Type 1 conditions ensure that the vertices in C_i for all $i \in [k]$ are protected. Define

$$\mathcal{K} = \text{the collection of all clique type classes.}$$

A vertex v from C_i is protected if and only if $d_S(v) \geq \frac{d_G(v)-1}{2}$, that is,

$$(x_i - 1)[C_i \in \mathcal{K}] + \sum_{j \in [k]} 1 \times [C_j \in N_H(C_i) \cap \mathcal{C}_1] + \sum_{j \in [k]} x_j [C_j \in N_H(C_i) \cap \mathcal{C}_{\geq 2}] \geq \frac{d_G(v) - 1}{2} \tag{6.1}$$

Type 2 Conditions: Let $\mathbf{x} = (x_1, \ldots, x_k)$ be the vector corresponds to $S \subseteq V(G)$. We want to make sure that the vector $\mathbf{x} = (x_1, \ldots, x_k)$ or S forms a globally minimal defensive alliance, that is, no proper subset of S forms a defensive alliance. Type 2 conditions ensure that no proper subset of S is a defensive alliance. To do this, we use the idea of Proposition 5 in [12] which is a polynomial time algorithm to check whether a given defensive alliance is globally minimal or not. Note that given a defensive alliance S, we will first remove an arbitrary vertex v from it. If S is globally minimal, clearly $S \setminus \{v\}$ cannot be a defensive alliance. To get a defensive alliance we must delete all the unprotected vertices of $S \setminus \{v\}$. In this way, if we keep on removing all the unprotected vertices then at the end we must be left with an empty set otherwise the non-empty set remaining at the end will form a defensive alliance which is impossible if S is a globally minimal defensive alliance. We repeat this for all vertices v in S. Note that the vertices in a type class have the same number of defenders in S and the same number of attackers outside S. This implies if one vertex of a type class becomes unprotected then all the vertices of that type class become unprotected. As the neighbourhood diversity of G is bounded by k, we guess an ordering of type classes in which we want them to become unprotected. We now summarize the above discussion in the following lemma.

Lemma 6.1.3. Let $I_1 \cup I_{\geq 2} = \{i_1, i_2, \ldots, i_r\}$, $r \leq k$. Then, S is a globally minimal defensive alliance in G if and only if for every $i_l \in I_1 \cup I_{\geq 2}$ there exists

$$\pi^{i_l} = \begin{cases} \text{a permutation of elements of } I_1 \cup I_{\geq 2}, & \text{if } i_l \in I_{\geq 2} \\ \text{a permutation of elements of } (I_1 \setminus \{i_l\}) \cup I_{\geq 2}, & \text{if } i_l \in I_1 \end{cases}$$

such that

1. Vertices in $C_{\pi^{i_l}(i_1)}$ are unprotected in $S \setminus \{v\}$ for any $v \in C_{i_l}$.

2. Define $S_{\pi^{i_l}(i_1)} = (S \setminus \{v\}) \setminus C_{\pi^{i_l}(i_1)}$ and $S_{\pi^{i_l}(i_j)} = S_{\pi^{i_l}(i_{j-1})} \setminus C_{\pi^{i_l}(i_j)}$ for all $j > 1$. Then, the vertices in $C_{\pi^{i_l}(i_{j+1})}$ are unprotected in $S_{\pi^{i_l}(i_j)}$ for all $j \geq 1$ and finally $S_{\pi^{i_l}(i_r)} = \emptyset$ if $i_l \in I_{\geq 2}$, otherwise $S_{\pi^{i_l}(i_{r-1})} = \emptyset$ if $i_l \in I_1$.

Proof. The forward direction follows from the idea of Proposition 5 in [12]. Suppose S is globally minimal in G. Clearly $S \setminus \{v\}$ cannot be a defensive alliance for any $v \in S$; suppose v is in the type class C_{i_l}. To get a defensive alliance we must delete all the unprotected vertices of $S \setminus \{v\}$. The unprotected vertices of $S \setminus \{v\}$ are in some type class. Suppose the unprotected vertices are in the type class $C_{\pi^{i_l}(i_1)}$. Note that if one vertex of $C_{\pi^{i_l}(i_1)}$ is unprotected in $S \setminus \{v\}$ then all the vertices of $C_{\pi^{i_l}(i_1)}$ are unprotected in $S \setminus \{v\}$. Thus we remove all the elements of $C_{\pi^{i_l}(i_1)}$ from $S \setminus \{v\}$ and set $S_{\pi^{i_l}(i_1)} = (S \setminus \{v\}) \setminus C_{\pi^{i_l}(i_1)}$. Clearly, $S_{\pi^{i_l}(i_1)}$ cannot be a defensive alliance in G as S is a globally minimal defensive alliance. Suppose the unprotected vertices of $S_{\pi^{i_l}(i_1)}$ are in the type class $C_{\pi^{i_l}(i_2)}$. Thus we remove all the elements of $C_{\pi^{i_l}(i_2)}$ from $S_{\pi^{i_l}(i_1)}$ and set $S_{\pi^{i_l}(i_2)} = S_{\pi^{i_l}(i_1)} \setminus C_{\pi^{i_l}(i_2)}$. Again, using the same argument, $S_{\pi^{i_l}(i_2)}$ cannot be a defensive alliance in G, so remove all the unprotected elements of $S_{\pi^{i_l}(i_2)}$ and so on. In this way, if we keep on removing all the unprotected vertices then at the end we must be left with an empty set otherwise the non-empty set remaining at the end will form a defensive alliance which is impossible as S is a globally minimal defensive alliance.

The reverse direction follows from the definition of a globally minimal defensive alliance. $\square$

Definition 6.1.1. Given i_l and π^{i_l}, we say $S \subseteq V(G)$ is *reducible* by (i_l, π^{i_l}) if S satisfies the conditions of Lemma 6.1.3.

For given i_l and π^{i_l}, we want S to satisfy the conditions in inequality 2 and 3. These are called type 2 conditions. By Lemma 6.1.3, the vertices in $C_{\pi^{i_l}(i_1)}$ must be unprotected in $S \setminus \{v\}$ for any $v \in C_{i_l}$. Let $u \in C_{\pi^{i_l}(i_1)}$. Then u is unprotected in $S \setminus \{v\}$ if and only if the number of neighbours of u in $S \setminus \{v\}$ is strictly less than $\frac{d_G(u)-1}{2}$, that is,

$$(x_{i_l}-1)[C_{i_l} \in N_H(C_{\pi^{i_l}(i_1)}) \cap \mathcal{C}_{\geq 2}] + \sum_{j \in [k]} 1 \times [C_j \in N_H(C_{\pi^{i_l}(i_1)}) \cap \mathcal{C}_1][j \neq l] +$$
$$\sum_{j \in [k]} x_j [C_j \in N_H(C_{\pi^{i_l}(i_1)}) \cap \mathcal{C}_{\geq 2}][j \neq l] + (x_{\pi^{i_l}(i_1)}-1)[C_{\pi^{i_l}(i_1)} \in \mathcal{K}] < \frac{d_G(u)-1}{2} \tag{6.2}$$

Let i_j be the jth element of $I_1 \cup I_{\geq 2}$ and $P^{i_l}(i_j) = \{\pi^{i_l}(i_1), \pi^{i_l}(i_2), \ldots, \pi^{i_l}(i_j)\}$ for $j \geq 1$. Let $u \in C_{\pi^{i_l}(i_{j+1})}$. Then u is unprotected in $S_{\pi^{i_l}(i_j)}$ if and only if the number of neighbours of u in $S_{\pi^{i_l}(i_j)}$ is strictly less than $\frac{d_G(u)-1}{2}$, that is,

$$\begin{aligned}
&(x_{i_l}-1)[i_l \notin P^{i_l}(i_j)][C_{i_l} \in N_H(C_{\pi^{i_l}(i_{j+1})}) \cap \mathcal{C}_{\geq 2}] + \\
&\sum_{i \in [k]} 1 \times [C_i \in N_H(C_{\pi^{i_l}(i_{j+1})}) \cap \mathcal{C}_1][i \notin P^{i_l}(i_j) \cup \{i_l\}] + \\
&\sum_{i \in [k]} x_i [C_i \in N_H(C_{\pi^{i_l}(i_{j+1})}) \cap \mathcal{C}_{\geq 2}][i \notin P^{i_l}(i_j) \cup \{i_l\}] \\
&+ (x_{\pi^{i_l}(i_{j+1})}-1)[C_{\pi^{i_l}(i_{j+1})} \in \mathcal{K}] < \frac{d_G(u)-1}{2}.
\end{aligned} \tag{6.3}$$

Our algorithm for GMDA will use the following annotated problem as subroutine. In the ANNOTATED GMDA problem, we are given a graph G, type classes $C_1, C_2, \ldots, C_k$ of G, two disjoint subsets $I_1, I_{\geq 2}$ of $\{1, 2, \ldots, k\}$, a permutation π^{i_l} for each $i_l \in I_1 \cup I_{\geq 2}$ where

$$\pi^{i_l} = \begin{cases} \text{a permutation of elements of } I_1 \cup I_{\geq 2}, & \text{if } i_l \in I_{\geq 2} \\ \text{a permutation of elements of } (I_1 \setminus \{i_l\}) \cup I_{\geq 2}, & \text{if } i_l \in I_1 \end{cases}$$

and the goal is to find a largest set $S \subseteq V(G)$ such that $S \cap C_i \neq \emptyset$ if $i \in I_1 \cup I_{\geq 2}$ and S is reducible by (i_l, π^{i_l}) for all $i_l \in I_1 \cup I_{\geq 2}$.

An algorithm for Annotated GMDA: Let $(G, (C_1, \ldots, C_k), I_1, I_{\geq 2}, \ \pi^{i_l} \ \forall \ i_l \in I_1 \cup I_{\geq 2})$ be an instance of Annotated GMDA. We reduce the problem of solving Annotated GMDA to an integer linear programming optimization with at most k variables as follows:

$$
\begin{aligned}
&\text{Maximize } |I_1| + \sum_{i \in [k]} x_i [C_i \in \mathcal{C}_{\geq 2}] \\
&\text{Subject to} \\
&x_i = 1 \ \ \forall \ \ i \in I_1 \\
&2 \leq x_i \leq |C_i| = n_i \quad i \in I_{\geq 2} \\
&\text{Equation (1)} \ \ \forall \ \ C_i \\
&\text{Equation (2)} \ \ \forall \ \ i_l \in I_1 \cup I_{\geq 2} \\
&\text{Equation (3)} \ \ \forall \ i_l \in I_1 \ \ \forall \ j \in [r-1] \\
&\text{Equation (3)} \ \ \forall \ i_l \in I_{\geq 2} \ \ \forall \ j \in [r]
\end{aligned}
$$

6.1.2 Running time

In our formulation for Annotated GMDA, we have at most k variables. The value of the objective function is bounded by n and the value of any variable in the integer linear programming is also bounded by n. The constraints can be represented using $O(k^2 \log n)$ bits. Therefore, Lemma 2.3.2 implies that we can solve the problem in $k^{O(k)} \cdot n^{O(1)}$ time.

Lemma 6.1.4. *If there exists an FPT algorithm for* Annotated GMDA *then there exists an FPT algorithm for* GMDA.

Proof. Suppose there exists an FPT algorithm for Annotated GMDA parameterized by neighbourhood diversity k. Note that there are at most $3^k(k!)^k$ candidates for Annotated GMDA instances. The reason is this. There are at most 3^k candidates for $I_1 \cup I_{\geq 2}$ as each C_i has three options: either in I_0, I_1 or $I_{\geq 2}$; there are at most $(k!)^k$ candidates for π^l for all $l \in I_1 \cup I_{\geq 2}$. In order to obtain a largest globally minimal defensive alliance, we first solve all Annotated GMDA instances, then consider a largest solution over all Annotated GMDA instances. Therefore, GMDA can be solved in FPT time parameterized by neighbourhood diversity k. $\square$

We have already proved that the ILP formula for an Annotated GMDA can be solved in $k^{O(k)} \cdot n^{O(1)}$ time. As there are at most $3^k(k!)^k$ many instances of Annotated GMDA, we can solve the GMDA in $k^{O(k^2)} \cdot n^{O(1)}$ time. Thus Theorem 6.1.1 holds.

6.2 No polynomial kernel parameterized by vertex cover number

A set $C \subseteq V$ is a *vertex cover* of $G = (V, E)$ if each edge $e \in E$ has at least one endpoint in X. The minimum size of a vertex cover in G is the *vertex cover number* of G, denoted by $vc(G)$. The problem is FPT parameterized by neighbourhood diversity implies that it is FPT parameterized by vertex cover number vc. In this section we prove the following kernelization hardness of Globally Minimal Defensive Alliance.

Theorem 6.2.1. Globally Minimal Defensive Alliance parameterized by the vertex cover number of the input graph does not admit a polynomial compression unless coNP $\subseteq$ NP/poly.

To prove Theorem 6.2.1, we give a polynomial parameter transformation (PPT) from the well-known Red Blue Dominating Set problem (RBDS) to Globally Minimal Defensive Alliance parameterized by vertex cover number. Recall that in RBDS we are given a bipartite graph $G = (T \cup S, E)$ and an integer k, and we are asked whether there exists a vertex set $X \subseteq S$ of size at most k such that every vertex in T has at least one neighbour in X. We also refer to the vertices of T as *terminals* and to the vertices of S as *sources* or *nonterminals*. The following theorem is known:

Theorem 6.2.2. [55] RBDS parameterized by $|T|$ does not admit a polynomial compression unless coNP $\subseteq$ NP/poly.

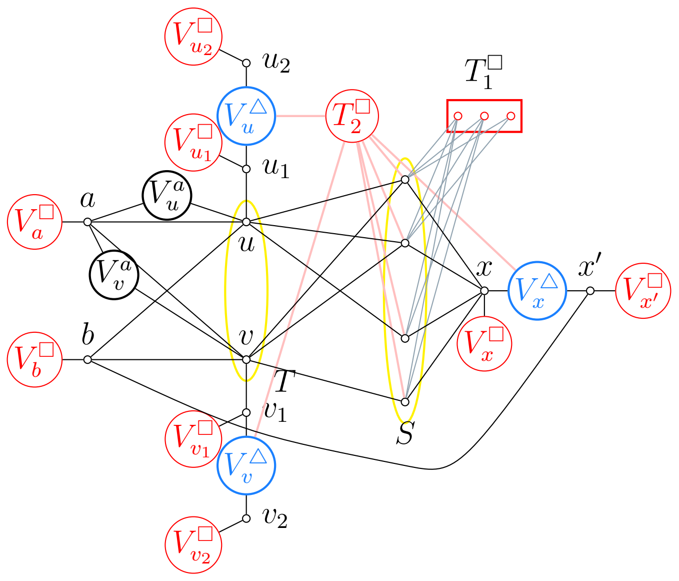


Figure 6.1: PPT from RBDS to Globally Minimal Defensive Alliance

6.2.1 Proof of Theorem 6.2.1

By Theorem 6.2.2, RBDS parameterized by $|T|$ does not admit a polynomial compression unless coNP $\subseteq$ NP/poly. To prove Theorem 6.2.1, we give a PPT from RBDS parameterized by $|T|$ to Globally Minimal Defensive Alliance parameterized by the vertex cover number. Given an instance $I = (G = (T \cup S, E), k)$ of RBDS, we construct an instance $I' = (G', k')$ of Globally Minimal Defensive Alliance as follows. See Fig. 6.1 for an illustration.

- We introduce two new vertices x and x', a set $V_x^{\triangle}$ of $4n$ vertices, a set $V_x^{\square}$ of $|S| - k + 4n + 1$ vertices, and a set $V_{x'}^{\square}$ of $4n + 1$ vertices. Make x adjacent to every vertex of $V_x^{\triangle} \cup S \cup V_x^{\square}$ and make x' adjacent to every vertex of $V_x^{\triangle} \cup V_{x'}^{\square}$.

- For every vertex $u \in T$, we introduce two vertices u_1 and u_2, a set $V_u^{\triangle}$ of $4n$ vertices, a set $V_{u_1}^{\square}$ of $4n + 1$ vertices, and a set $V_{u_2}^{\square}$ of $4n$ vertices. Make u_1 adjacent to every vertex of $\{u\} \cup V_u^{\triangle} \cup V_{u_1}^{\square}$, and make u_2 adjacent to every vertex of $V_u^{\triangle} \cup V_{u_2}^{\square}$.

- We introduce two new vertices a and b into G', a set $V_a^{\square}$ of $|T| + \sum\limits_{u \in T} (d_S(u) - 1) + 2$ vertices, and a set $V_b^{\square}$ of $|T|+1$ vertices. Make a adjacent to every vertex of $T \cup V_a^{\square}$, and make b adjacent to very vertex of $T \cup V_b^{\square} \cup \{x'\}$. Let $d_S(u)$, $u \in T$, denote the number of neighbours of u in S. For each $u \in T$, we add a set $V_u^a = \{u_1^a, u_2^a, \dots, u_{d_S(u)-1}^a\}$ of $d_S(u) - 1$ vertices and make a and u adjacent to every vertex of V_u^a.

- Finally, we add a set $T^\square = T_1^\square \cup T_2^\square$ of vertices where $|T_1^\square| = |T| + 1$ and $|T_2^\square| = 2$. We make every vertex of $V_x^\triangle \cup \bigcup_{u \in T} V_u^\triangle$ adjacent to every vertex of $T_2^\square$. For every $s \in S$, we make s adjacent to $d_T(s) + 1$ many arbitrary vertices of $T_1^\square$. For every $t \in T^\square$, we introduce $d + 2$ vertices and make them adjacent to t where d is the degree of t until this point of the construction; *these vertices are not shown in the figure.* Finally, we set $k' = (|S| - k) + 4n + 2 + |T|(4n + 3) + 1$.

Now we claim that there exists a vertex set $X \subseteq S$ in G of size at most k such that every vertex in T has at least one neighbour in X if and only if G' has a globally minimal defensive alliance of size at least k'. Suppose there is a vertex set $X \subseteq S$ in G of size at most k such that every vertex in T has at least one neighbour in X. We show that the set

$$H = (S \setminus X) \cup \{b, x, x'\} \cup V_x^\triangle \cup \bigcup_{u \in T} (\{u, u_1, u_2\} \cup V_u^\triangle) \cup \bigcup_{u \in T} \{u_1^a, \dots, u_{d_X(u)-1}^a\}$$

is a globally minimal defensive alliance. Clearly $|H| \geq k'$ and observe that every vertex in H is marginally protected. Let u be an arbitrary element of H. If u is an element of T, then $N_H(u) = \{b, u_1\} \cup \{u_1^a, \dots, u_{d_X(u)-1}^a\} \cup N_{S \setminus X}(u)$ and $N_{H^c}(u) = \{b\} \cup \{u_{d_X(u)}, \dots, u_{d_S(u)-1}\} \cup N_X(u)$. That is, $d_H(u) = d_S(u) + 1$ and $d_{H^c}(u) = d_S(u) + 1$; thus u is marginally protected. Similarly, it is easy to check that other vertices of H are also marginally protected. We also see that $G'[H]$ is connected. Using Observation 2, H is a globally minimal defensive alliance.

To prove the reverse direction of the equivalence, we assume that there exists a globally minimal defensive alliance H of size at least k'. By Observation 1.3.1, H can never contain a vertex of degree one. As one degree vertices cannot be part of the solution, we observe that no vertex from $T^\square$ is part of the solution. This is true because we will not be able to protect any vertex from $T^\square$ as more than half of the neighbours are one degree vertices. We claim that $T \subseteq H$. For the sake of contradiction suppose that there exists some $u \in T$ such that $u \notin H$. Then vertices in $V_u^\triangle \cup \{u_1, u_2\}$ cannot be inside H as a globally minimal defensive alliance must be connected. If we do not include $V_u^\triangle$ inside H then H cannot achieve the size k'. Therefore, we must include u in H. From the above argument, we also see that we must include u_1 and u_2 inside H as otherwise vertices in $V_u^\triangle$ cannot be protected. Therefore, we have $\bigcup_{u \in T} (V_u^\triangle \cup \{u, u_1, u_2\}) \subseteq H$. Similarly, we can argue that $V_x^\triangle \cup \{x, x', b\} \subseteq H$. Observe that $u \in T$ must be marginally protected in H as otherwise $H \setminus (V_u^{\triangle} \cup \{u_1, u_2\})$ forms a defensive alliance, which is not possible. For $u \in T$, we have $N(u) = N_S(u) \cup V_u^a \cup \{a, b, u_1\}$ and hence $d(u) = 2d_S(u) + 2$. Since $a \notin H$ and $b, u_1 \in H$, we must have added at most $d_S(u) - 1$ nodes from $N_S(u)$ in H for each $u \in T$. Consider $X = H^c \cap S$. Clearly, every vertex $u \in T$ has at least one neighbour in X. Next, we see that $|X| \leq k$. As otherwise, $|S \cap H| < |S| - k$ and then x cannot be protected. This implies that I is a yes instance. $\square$

6.3 Hardness Results

In this section, we show that Globally Minimal Defensive Alliance is W[1]-hard when parameterized by the size of a vertex deletion set into trees of height at most three, i.e., a subset R of the vertices of the graph such that every component in the graph, after removing R, is a tree of height at most three. We give a reduction from Multi-Colored Clique. The input of Multi-Colored Clique consists of a graph G, an integer k, and a partition $(V_1, \dots, V_k)$ of the vertices of G; the task is to decide if there is a k-clique containing exactly one vertex from each set V_i.

Theorem 6.3.1. Globally Minimal Defensive Alliance is W[1]-hard when parameterized by the size of a vertex deletion set into trees of height at most 3.

Proof. The approach for using Multi-Colored Clique in reductions is described in [46], and has been proven to be very useful in showing hardness results in the parameterized complexity setting. We use G to denote a graph colored with k colors given in an instance of Multi-Colored Clique, and G' to denote the graph in the reduced instance of Globally Minimal Defensive Alliance. For a color $i \in [k]$, let V_i denote the subset of vertices in G colored with color i and for a pair of distinct colors $i, j \in [k]$, let E_{ij} denote the subset of edges in G with endpoints colored i and j.

We construct G' using two types of gadgets. Our goal is to guarantee that any globally minimal defensive alliance in G' with a specific size encodes a multi-colored clique in G. These gadgets are the selection and validation gadgets. The selection gadgets encode the selection of k vertices and $\binom{k}{2}$ edges that together encode a vertex and edge set of some multi-colored clique in G. The selection gadgets also ensure that in fact k distinct vertices are chosen from k distinct color classes, and that distinct $\binom{k}{2}$ edges are chosen from $\binom{k}{2}$ distinct edge color classes. The validation gadgets validate the selection done in the selection gadgets in the sense that they make sure that the edges chosen are in fact incident to the selected vertices. In the following we sketch the construction of selection and validation gadgets as given in [14]:

Selection: For each color-class $i \in [k]$, and each pair of distinct colors $i, j \in [k]$, we construct an i-selection gadget and an $\{i, j\}$-selection gadget which respectively encode the selection of a vertex colored i and an edge colored $\{i, j\}$ in G. The i-selection gadget consists of a vertex x_v for every vertex $v \in V_i$, and likewise, the $\{i, j\}$-selection gadget consists of a vertex $x_{\{u,v\}}$ for every edge $\{u, v\} \in E_{\{i,j\}}$. There are no edges between the vertices of the selection gadgets, that is, the union of all vertices in these gadgets is an independent set in G'.

Validation: We assign to every vertex v in G two unique identification numbers, low(v) and high(v), with low$(v) \in [n]$ and high$(v) = 2n - \text{low}(v)$ where n is the order of the graph G. For every pair of distinct colors $i, j \in [k]$, we construct validation gadgets between the $\{i, j\}$-selection gadget and the i- and j-selection gadget. Let i and j be any pair of distinct colors. We describe the validation gadget between the i- and $\{i, j\}$-selection gadgets. It consists of two validation vertices α_{ij} and β_{ij}, the *validation-pair* of this gadget. The first vertex α_{ij} of this pair is adjacent to each vertex $x_u, u \in V_i$, by high(u) parallel edges, and to each edge-selection vertex $x_{\{u,v\}}, \{u, v\} \in E_{\{ij\}}$ and $v \in V_j$, by low(u) parallel edges. The other vertex β_{ij} is adjacent to each $x_u, u \in V_i$, by low(u) parallel edges, and to each $x_{\{u,v\}}, \{u, v\} \in E_{\{i,j\}}$ and $v \in V_j$, by high(u) parallel edges. We next subdivide the edges between the selection and validation gadgets to obtain a simple graph, where all new vertices introduced by the subdivision are referred to as the *connection vertices.* The set of connection vertices adjacent to one of the validation vertices $\{\alpha_{ij}, \beta_{ij}\}$ and x_u is denoted by A^u_{ij} and the set of connection vertices adjacent to one of the validation vertices $\{\alpha_{ij}, \beta_{ij}\}$ and $x_{\{u,v\}}$ is denoted by B^{uv}_{ij}.

For each connection vertex we add two new vertices and make them adjacent to it. For each x_u in the i-selection gadget, we introduce $d(x_u)$ new vertices and make them adjacent to x_u, and likewise for each $x_{\{u,v\}}$ in $\{i, j\}$-selection gadget we introduce $d(x_{\{u,v\}})$ new vertices and make them adjacent to $x_{\{u,v\}}$. Let L be the set of all validation vertices in G'. Set $N = 100n^2$.

For every validation vertex $\alpha \in L$, we add the following *protection gadget.* We introduce

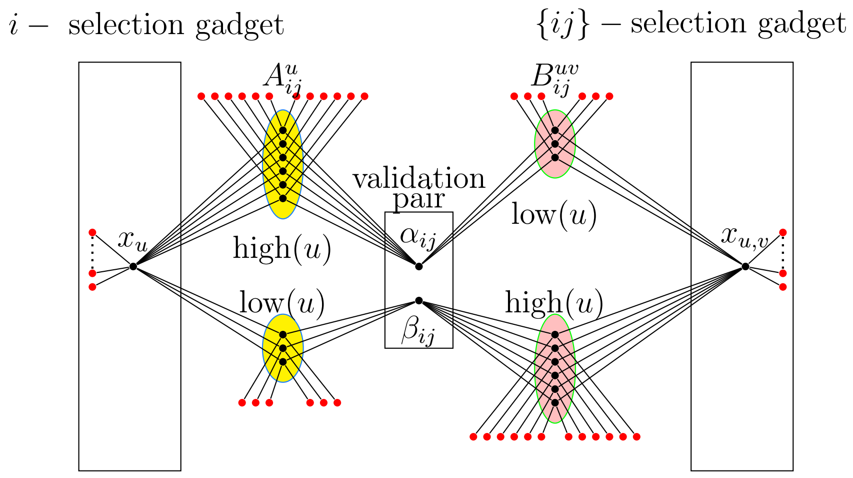


Figure 6.2: A graphical depiction of the validation gadget. In the example, $n = 5$ and low(u) $= 3$. Note that A^u_{ij} contains the connection vertices in the yellow region and B^{uv}_{ij} contains the connection vertices in the pink region. The red vertices are one-degree vertices, and hence they are not part of any globally minimal defensive alliance.

a new vertex $\alpha^{\triangle}$, a set $V^{\triangle}_{\alpha}$ of N vertices, and a set of $V^{\square}_{\alpha^{\triangle}}$ of N vertices. We make α adjacent to every vertex of $V^{\triangle}_{\alpha}$; make $\alpha^{\triangle}$ adjacent to every vertex of $V^{\triangle}_{\alpha} \cup V^{\square}_{\alpha^{\triangle}}$. For each vertex $x \in V^{\triangle}_{\alpha}$, we introduce two new vertices and make them adjacent to x.

Finally for every validation vertex $\alpha \in L$, we add the following *marginal protection gadget*. This gadget ensures that α is marginally protected in a globally minimal defensive alliance. We introduce two new vertices α' and $\alpha'^{\triangle}$, a set $V^{\triangle}_{\alpha'}$ of N vertices, a set $V^{\square}_{\alpha'}$ of N vertices, and a set $V^{\square}_{\alpha'^{\triangle}}$ of $N+1$ vertices. We make α' adjacent to every vertex of $\{\alpha\} \cup V^{\triangle}_{\alpha'} \cup V^{\square}_{\alpha'}$; make $\alpha'^{\triangle}$ adjacent to every vertex of $V^{\triangle}_{\alpha'} \cup V^{\square}_{\alpha'^{\triangle}}$. For each vertex $x \in V^{\triangle}_{\alpha'}$, we introduce two new vertices and make them adjacent to x. Let d be the degree of α up to this point of construction. Finally for each $\alpha \in L$, we introduce a set $V^{\square}_{\alpha}$ of $N + 4n - d + 1$ vertices and make them adjacent to α. This completes the construction of graph G'.
We set $k' = k + \binom{k}{2} + 8n\binom{k}{2} + 2(4N+8)\binom{k}{2}$. We observe that on removing

$$R = \{\alpha_{ij}, \alpha_{ji}, \alpha^{\triangle}_{ij}, \alpha^{\triangle}_{ji}, \alpha'_{ij}, \alpha'_{ji}, \alpha'^{\triangle}_{ij}, \alpha'^{\triangle}_{ji} \ : \ i, j \in [k], i \neq j\}$$
$$\cup \{\beta_{ij}, \beta_{ji}, \beta^{\triangle}_{ij}, \beta^{\triangle}_{ji}\beta'_{ij}, \beta'_{ji}, \beta'^{\triangle}_{ij}, \beta'^{\triangle}_{ji} \ : \ i, j \in [k], i \neq j\}$$

from graph G', we are left with trees of height at most 3. Note that $|R| = 16\binom{k}{2}$, a function of k only. We claim that G has a k-clique with exactly one vertex from each V_i if and only

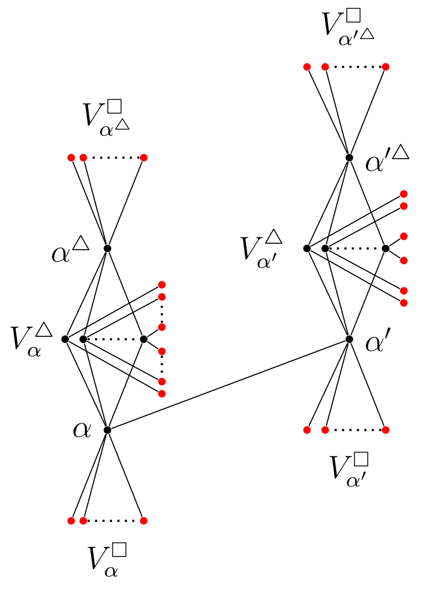


Figure 6.3: A graphical depiction of protection and marginal protection gadgets associated with validation vertex α. Note that the red vertices are one-degree vertices, and hence they are not part of any globally minimal defensive alliance.

if G' has a globally minimal defensive alliance of size at least k'. Suppose that $v_1 \in V_1$, $v_2 \in V_2, \ldots, v_k \in V_k$ is a k-clique in G. We show that

$$\begin{aligned} S = \big\{x_{v_i} \ : \ i \in [k]\big\} \cup \big\{x_{\{v_i,v_j\}} \ : \ i,j \in [k], i \neq j\big\} \cup \bigcup_{i,j\in[k],i\neq j} A^{v_i}_{ij} \cup B^{v_iv_j}_{ij} \cup A^{v_j}_{ji} \cup B^{v_jv_i}_{ji} \\ \cup \bigcup_{i,j\in[k],i\neq j} \{\alpha_{ij}, \alpha^{\triangle}_{ij}, \alpha'_{ij}, \alpha'^{\triangle}_{ij}, \beta_{ij}, \beta^{\triangle}_{ij}, \beta'_{ij}, \beta'^{\triangle}_{ij}\} \cup V^{\triangle}_{\alpha_{ij}} \cup V^{\triangle}_{\alpha'_{ij}} \cup V^{\triangle}_{\beta_{ij}} \cup V^{\triangle}_{\beta'_{ij}} \\ \cup \bigcup_{i,j\in[k],i\neq j} \{\alpha_{ji}, \alpha^{\triangle}_{ji}, \alpha'_{ji}, \alpha'^{\triangle}_{ji}, \beta_{ji}, \beta^{\triangle}_{ji}, \beta'_{ji}, \beta'^{\triangle}_{ji}\} \cup V^{\triangle}_{\alpha_{ji}} \cup V^{\triangle}_{\alpha'_{ji}} \cup V^{\triangle}_{\beta_{ji}} \cup V^{\triangle}_{\beta'_{ji}} \end{aligned}$$

is a globally minimal defensive alliance. Clearly $|S| = k'$. To prove that S is a globally minimal defensive alliance, we prove that S is connected and every vertex in S is marginally protected. It is easy to see that each vertex in

$$\big\{x_{v_i} \ : \ i \in [k]\big\} \cup \big\{x_{\{v_i,v_j\}} \ : \ i,j \in [k], i \neq j\big\}$$

is marginally protected as all the connection vertices adjacent to it are inside the solution and the same number of one degree neighbours are outside the solution. It is also easy to see that every connection vertex in $\bigcup_{i,j\in[k],i\neq j} A_{ij}^{v_i} \cup B_{ij}^{v_iv_j} \cup A_{ji}^{v_j} \cup B_{ji}^{v_jv_i}$ is marginally protected as it has two neighbours inside the solution and two neighbours outside the solution. Similarly, we observe that all the vertices in the set

$$\bigcup_{i,j\in[k],i\neq j} \left\{\alpha_{ij}^{\triangle}, \alpha'_{ij}, \alpha'^{\triangle}_{ij}, \beta_{ij}^{\triangle}, \beta'_{ij}, \beta'^{\triangle}_{ij}\right\} \cup V_{\alpha_{ij}}^{\triangle} \cup V_{\alpha'_{ij}}^{\triangle} \cup V_{\beta_{ij}}^{\triangle} \cup V_{\beta'_{ij}}^{\triangle}$$
$$\cup \bigcup_{i,j\in[k],i\neq j} \left\{\alpha_{ji}^{\triangle}, \alpha'_{ji}, \alpha'^{\triangle}_{ji}, \beta_{ji}^{\triangle}, \beta'_{ji}, \beta'^{\triangle}_{ji}\right\} \cup V_{\alpha_{ji}}^{\triangle} \cup V_{\alpha'_{ji}}^{\triangle} \cup V_{\beta_{ji}}^{\triangle} \cup V_{\beta'_{ji}}^{\triangle}$$

are also marginally protected. Lastly, we prove that the vertices in the set $\bigcup_{i,j\in[k],i\neq j} \{\alpha_{ij}, \beta_{ij}, \alpha_{ji}, \beta_{ji}\}$ are marginally protected. Consider α_{ij}; it has total $2n$ connection vertices neighbours inside the solution as $\text{high}(u) + \text{low}(u) = 2n$. Since $V_{\alpha_{ij}}^{\triangle} \subseteq S$ and $V_{\alpha_{ij}}^{\square} \cap S = \emptyset$, note that, including itself, α_{ij} has $N+2n+1$ neighbours in S and $d_{S^c}(\alpha_{ij}) = (d-2n)+(N+4n-d+2) = N+2n+1$. Therefore α_{ij} is marginally protected. It is easy to observe that S is connected. This shows that S is a globally minimal defensive alliance.

In the reverse direction, we assume that G' admits a globally minimal defensive alliance S of size at least k'. By Observation 1, no vertex of degree one can be part of a globally minimal defensive alliance of size ≥ 2 as a one-degree vertex itself forms a defensive alliance. We claim that the vertices in

$$\bigcup_{i,j\in[k],i\neq j} \left\{\alpha_{ij}, \alpha_{ij}^{\triangle}, \alpha'_{ij}, \alpha'^{\triangle}_{ij}, \beta_{ij}, \beta_{ij}^{\triangle}, \beta'_{ij}, \beta'^{\triangle}_{ij}\right\} \cup V_{\alpha_{ij}}^{\triangle} \cup V_{\alpha'_{ij}}^{\triangle} \cup V_{\beta_{ij}}^{\triangle} \cup V_{\beta'_{ij}}^{\triangle}$$
$$\cup \bigcup_{i,j\in[k],i\neq j} \left\{\alpha_{ji}, \alpha_{ji}^{\triangle}, \alpha'_{ji}, \alpha'^{\triangle}_{ji}, \beta_{ji}, \beta_{ji}^{\triangle}, \beta'_{ji}, \beta'^{\triangle}_{ji}\right\} \cup V_{\alpha_{ji}}^{\triangle} \cup V_{\alpha'_{ji}}^{\triangle} \cup V_{\beta_{ji}}^{\triangle} \cup V_{\beta'_{ji}}^{\triangle}$$

are always in S. Assume, for the sake of contradiction, that $\alpha_{ij} \notin S$. Then we cannot include any vertex from $V_{\alpha_{ij}}^{\triangle}$ in the solution. This is true because every globally minimal defensive alliance must be connected. If we cannot include the set $V_{\alpha_{ij}}^{\triangle}$ then clearly $|S| \leq k'$, a contradiction. Next, we observe that the protection of α_{ij} requires at least one vertex from the set $V_{\alpha_{ij}}^{\triangle}$ inside the solution. As every vertex in $V_{\alpha_{ij}}^{\triangle}$ has two one degree neighbours, it implies that the protection of that vertex requires $\alpha_{ij}^{\triangle}$ inside the solution. Now, the protection of $\alpha_{ij}^{\triangle}$ forces the full set $V_{\alpha_{ij}}^{\triangle}$ inside the solution as its one-degree neighbours are always outside S. Similarly, we argue that α'_{ij} and $V_{\alpha'_{ij}}^{\triangle}$ will be inside the solution. This proves the claim.

Observe that the above set has size exactly equal to $2(4N+8)\binom{k}{2}$. We need to add at least $k+\binom{k}{2}+8n\binom{k}{2}$ more vertices in S. We claim that

$$\bigcup_{i,j\in[k],i\neq j} A_{ij}^{v_i}\cup B_{ij}^{v_iv_j}\cup A_{ji}^{v_j}\cup B_{ji}^{v_jv_i}\subseteq S.$$

Observe that α_{ij} must be marginally protected inside the solution as otherwise $S\backslash(\{\alpha'_{ij},\alpha'^{\triangle}_{ij}\}\cup V^{\triangle}_{\alpha'_{ij}})$ forms a defensive alliance. This is equivalent to say that the marginal protection of α_{ij} requires $2n$ neighbours from connection vertices inside the solution. Similarly, the marginal protection of β_{ij} requires exactly $2n$ neighbours from connection vertices inside the solution. Therefore, for each $i,j\in[k]$, $i\neq j$, we have $A_{ij}^{v_i}\cup B_{ij}^{v_iv_j}\cup A_{ji}^{v_j}\cup B_{ji}^{v_jv_i}\subseteq S$.

We now show that each vertex (or edge, respectively) selection gadget contributes exactly one vertex in S. We first show that every vertex selection gadget contributes at most one vertex inside the solution. Suppose there exists a vertex selection gadget which contributes at least two vertices inside the solution. Without loss of generality, suppose x_{u_1} and x_{u_2} from an i-selection gadget are inside the solution. It implies that the protection of x_{u_1} and x_{u_2} requires $A_{ij}^{u_1}$ and $A_{ij}^{u_2}$, respectively, inside the solution. Note that either α_{ij} or β_{ij} will have more than $2n$ connection vertex neighbours inside the solution as either $\text{high}(u_1)+\text{high}(u_2)>2n$ or $\text{low}(u_1)+\text{low}(u_2)>2n$. This is a contradiction as either α_{ij} or β_{ij} will not be marginally protected inside S. We can argue similarly for edge selection gadgets and other color classes as well. Next, we show that every vertex (or edge, respectively) selection gadget contributes at least one vertex to the solution. For the sake of contradiction, assume that some i-selection gadget does not contribute any vertex to the solution. In this case, it will not be possible to protect the validation vertices α_{ij} and β_{ij} for $j\neq i$, because no connection vertex between i-selection gadget and the validation pair $\{\alpha_{ij},\beta_{ij}\}$ can be added to the solution. This is true as the protection of connection vertices require their neighbour in the i-selection gadget to be part of the solution. As the edge selection gadget can contribute at most one vertex due to above argument, the vertices in the set $\{\alpha_{ij},\beta_{ij}\}$ will have $<2n$ neighbours from the set of connection vertices. This makes the protection of the validation vertices α_{ij} and β_{ij} impossible. Therefore each selection gadget contributes exactly one vertex inside the solution.

Next, we claim that if the i-selection gadget contributes x_{v_i} and if the j-selection gadget contributes x_{v_j} then the $\{i,j\}$-selection gadget must contribute $x_{\{v_i,v_j\}}$. Assume, for the sake of contradiction, that the $\{i,j\}$-selection gadget contributes $x_{\{v_k,v_j\}}$ where $v_k \neq v_i$. In this case, we have $A_{ij}^{v_i} \cup A_{ji}^{v_j} \cup B_{ij}^{v_i v_k} \cup B_{ji}^{v_j v_k} \subseteq S$. Note that one of the vertices from the set $\{\alpha_{ij}, \beta_{ij}\}$ is not protected because when $v_i \neq v_k$ either high(v_i)+low(v_k) $< 2n$ or low(v_i)+high(v_k) $< 2n$. This is a contradiction. Therefore, we proved that if the i-selection gadget contributes x_{v_i} and if the j-selection gadget contributes x_{v_j}, then the $\{i,j\}$-selection gadget must contribute $x_{\{v_i,v_j\}}$. It implies that the set $\{v_i \in G \mid x_{v_i} \in i\text{-selection gadget}, i \in [k]\}$ forms a multicolored clique in G. □

Clearly trees of height at most three are trivially acyclic. Moreover, it is easy to verify that such trees have pathwidth [93] and treedepth [110] at most three, which implies:

Theorem 6.3.2. Globally Minimal Defensive Alliance is W[1]-hard when parameterized by any of the following parameters:

- the feedback vertex set number,
- the pathwidth and hence also treewidth of the input graph,
- the treedepth of the input graph.

6.4 NP-completeness

Rooted Globally Minimal Defensive Alliance asks for a globally minimal defensive alliance S that contains a specified vertex r in a graph G. The vertex r is called the root of S. A globally minimal defensive alliance of G will not be of use for this problem unless the set contains r. We define the problem as follows:

Rooted Globally Minimal Defensive Alliance
Input: An undirected graph $G = (V, E)$, a vertex $r \in V$.
Question: Does there exist a globally minimal defensive alliance $S \subseteq V$, such that $r \in S$?

In this section, we prove the following theorem:

Theorem 6.4.1. Rooted Globally Minimal Defensive Alliance is NP-hard.

Proof. We prove it is NP-hard by giving a polynomial time reduction from Clique on regular graphs to Rooted Globally Minimal Defensive Alliance. See Figure 6.4 for an illustration. Let $I = (G, k)$ be an instance of Clique, where G is an s-regular graph. We construct an instance $I' = (G', r)$ of Rooted Globally Minimal Defensive Alliance as follows. First, we introduce one vertex r and a set of vertices $\{z_1, z_1, \dots, z_{n-2k}\}$ into G'.

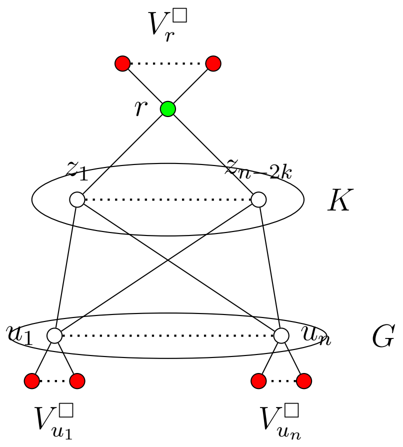


Figure 6.4: Reduction from Clique to Rooted Globally Minimal Defensive Alliance

We make the set $\{z_1, z_1, \dots, z_{n-2k}\}$ a clique K in G'. We make every vertex of K adjacent to every vertex u of G and the vertex r. Next, we introduce a set $V_r^\square$ of $n - 2k$ vertices and make them adjacent to r. For every $u \in V(G)$, we introduce a set $V_u^\square$ of $n-s-2$ vertices and make them adjacent to u. This completes the construction of G'. To prove the correctness of the reduction, we claim that G has a k-clique if and only if G' admits a globally minimal defensive alliance containing r. Assume first that G has a clique C of size k. We claim that $S = C \cup K \cup \{r\}$ is a globally minimal defensive alliance of G'. We observe that all the vertices in S are marginally protected and $G[S]$ is connected. Using Observation 1.3.2, S is a globally minimal defensive alliance containing r in G'.

To prove the reverse direction of the equivalence, suppose G' has a globally minimal defensive alliance S containing vertex r. By Observation 1.3.1, one degree vertices cannot be part of S. This implies that the protection of r requires all the vertices of K in the solution. Therefore, we can assume that $K \subseteq S$. We observe that every vertex of K needs at least k vertices from $V(G)$ for its protection. We also see that, if we take more than k vertices from $V(G)$ inside S then all the vertices in K are overprotected. Then $S \setminus \{r\}$ is also a defensive alliance. This is a contradiction to the assumption that S is globally minimal defensive alliance. This proves that $V(G)$ contributes exactly k vertices in the solution. Let us denote this set by C. Since G is s-regular, note that each $u \in C$ requires exactly $k-1$ neighbours from C for its protection. Then u has $n-2k+k-1 = n-k-1$ neighbours in S and $(n-s-2)+(s-k+1) = n-k-1$ neighbours outside S. This implies that C is a clique of size exactly k. □

6.5 Closing Remarks and Future Directions

The main contributions in this work are that GLOBALLY MINIMAL DEFENSIVE ALLIANCE is FPT when parameterized by neighborhood diversity; no polynomial kernel parameterized by vertex cover number; and the problem is W[1]-hard parameterized by a wide range of fairly restrictive structural parameters such as the feedback vertex set number, pathwidth, treewidth, and treedepth of the input graph. We also proved that given a vertex $v \in V(G)$, deciding if G has a globally minimal defensive alliance containing v, is NP-complete. The parameterized complexity of GLOBALLY MINIMAL DEFENSIVE ALLIANCE remains unsettled when parameterized by the solution size. It would be interesting to consider the parameterized complexity with respect to structural parameters twin-cover and vertex integrity. The modular width parameter also appears to be a natural parameter to consider here; since there are graphs with bounded modular-width and unbounded neighborhood diversity, we believe this is also an interesting open problem.

Chapter 7

Offensive Alliances in Graphs

7.1 Introduction

The Offensive Alliance problem has been studied extensively during the last twenty years. A set $S \subseteq V$ of vertices is an *offensive alliance* in an undirected graph $G = (V, E)$ if each $v \in N(S)$ has at least as many neighbours in S as it has neighbours (including itself) not in S. We study the classical and parameterized complexity of the Offensive Alliance problem, where the aim is to find a minimum size offensive alliance. We enhance our understanding of the problem from the viewpoint of parameterized complexity by showing that (1) the problem is W[1]-hard parameterized by a wide range of fairly restrictive structural parameters such as the feedback vertex set number, treewidth, pathwidth, and treedepth of the input graph; this puts it among the few problems that are FPT when parameterized by solution size but not when parameterized by treewidth (unless FPT=W[1]), (2) the problem cannot be solved in time $\mathcal{O}^*(2^{o(k \log k)})$ where k is the solution size, unless ETH fails, (3) it does not admit a polynomial kernel parameterized by solution size and vertex cover of the input graph. On the positive side we prove that (4) it can be solved in time $\mathcal{O}^*(\texttt{vc(G)}^{\mathcal{O}(\texttt{vc(G)})})$ where $\texttt{vc(G)}$ is the vertex cover number of the input graph. (5) it admits an FPT algorithm when parameterized by vertex integrity of input graph. In terms of classical complexity, we prove that (6) the problem cannot be solved in time $2^{o(n)}$ even when restricted to bipartite graphs, unless ETH fails, (7) it cannot be solved in time $2^{o(\sqrt{n})}$ even when restricted to apex graphs, unless ETH fails. We also prove that (8) it is NP-complete even when restricted to bipartite, chordal, split and circle graphs. This chapter is based on the papers [67, 60].

7.2 W[1]-Hardness Parameterized by Structural Parameters

In this section, we show that Offensive Alliance is W[1]-hard parameterized by the size of a vertex deletion set into trees of height at most seven, that is, a subset D of the vertices of the graph such that every component in the graph, after removing D, is a tree of height at most seven. On the way towards this result, we provide hardness results for several interesting versions of Offensive Alliance which we require in our proofs.

The Offensive Alliance$^{\mathrm{F}}$ problem generalizes Offensive Alliance where some vertices are forced to be outside the solution; these vertices are called *forbidden vertices*. This variant can be formalized as follows:

Offensive Alliance$^{\mathrm{F}}$
Input: An undirected graph $G = (V, E)$, an integer r and a set $V_\square \subseteq V$ of forbidden vertices such that each degree one forbidden vertex is adjacent to another forbidden vertex and each forbidden vertex of degree greater than one is adjacent to a degree one forbidden vertex.
Question: Is there an offensive alliance $S \subseteq V$ such that (i) $1 \leq |S| \leq r$, and (ii) $S \cap V_\square = \emptyset$?

The Strong Offensive Alliance$^{\mathrm{FN}}$ problem is a generalization of Strong Offensive Alliance$^{\mathrm{F}}$ that, in addition, requires some "necessary" vertices to be in S. This variant can be formalized as follows:

Strong Offensive Alliance$^{\mathrm{FN}}$
Input: An undirected graph $G = (V, E)$, an integer r, a set $V_\triangle \subseteq V$ of necessary vertices, and a set $V_\square \subseteq V$ of forbidden vertices such that each degree one forbidden vertex is adjacent to another forbidden vertex and each forbidden vertex of degree greater than one is adjacent to a degree one forbidden vertex.
Question: Is there a strong offensive alliance $S \subseteq V$ such that (i) $1 \leq |S| \leq r$, (ii) $S \cap V_\square = \emptyset$, and (iii) $V_\triangle \subseteq S$?

While Offensive Alliance asks for offensive alliance of size at most r, we also consider Exact Offensive Alliance that concerns offensive alliance of size exactly r. Analogously, we also define exact versions of Strong Offensive Alliance presented above. To prove Lemma 7.2.2, we consider the following problem:

Multidimensional Relaxed Subset Sum (MRSS)
Input: Two integers k and k', a set $S = \{s_1, \ldots, s_n\}$ of vectors with $s_i \in \mathbb{N}^k$ for every i with $1 \leq i \leq n$ and a target vector $t \in \mathbb{N}^k$.
Parameter: $k + k'$
Question: Is there a subset $S' \subseteq S$ with $|S'| \leq k'$ such that $\sum\limits_{s \in S'} s \geq t$?

Lemma 7.2.1. [74] MRSS is W[1]-hard when parameterized by the combined parameter $k + k'$, even if all integers in the input are given in unary.

We now show that Strong Offensive Alliance$^{\text{FN}}$ is W[1]-hard parameterized by the size of a vertex deletion set into trees of height at most 5, via a reduction from MRSS.

Lemma 7.2.2. Strong Offensive Alliance$^{\text{FN}}$ is W[1]-hard when parameterized by the size of a vertex deletion set into trees of height at most 5.

Proof. To prove this we reduce from MRSS, which is known to be W[1]-hard when parameterized by the combined parameter $k+k'$, even if all integers in the input are given in unary [74]. Let $I = (k, k', S, t)$ be an instance of MRSS. We construct an instance $I' = (G, r, V_\triangle, V_\square)$ of Strong Offensive Alliance$^{\text{FN}}$ the following way. See Figure 7.1 for an illustration.

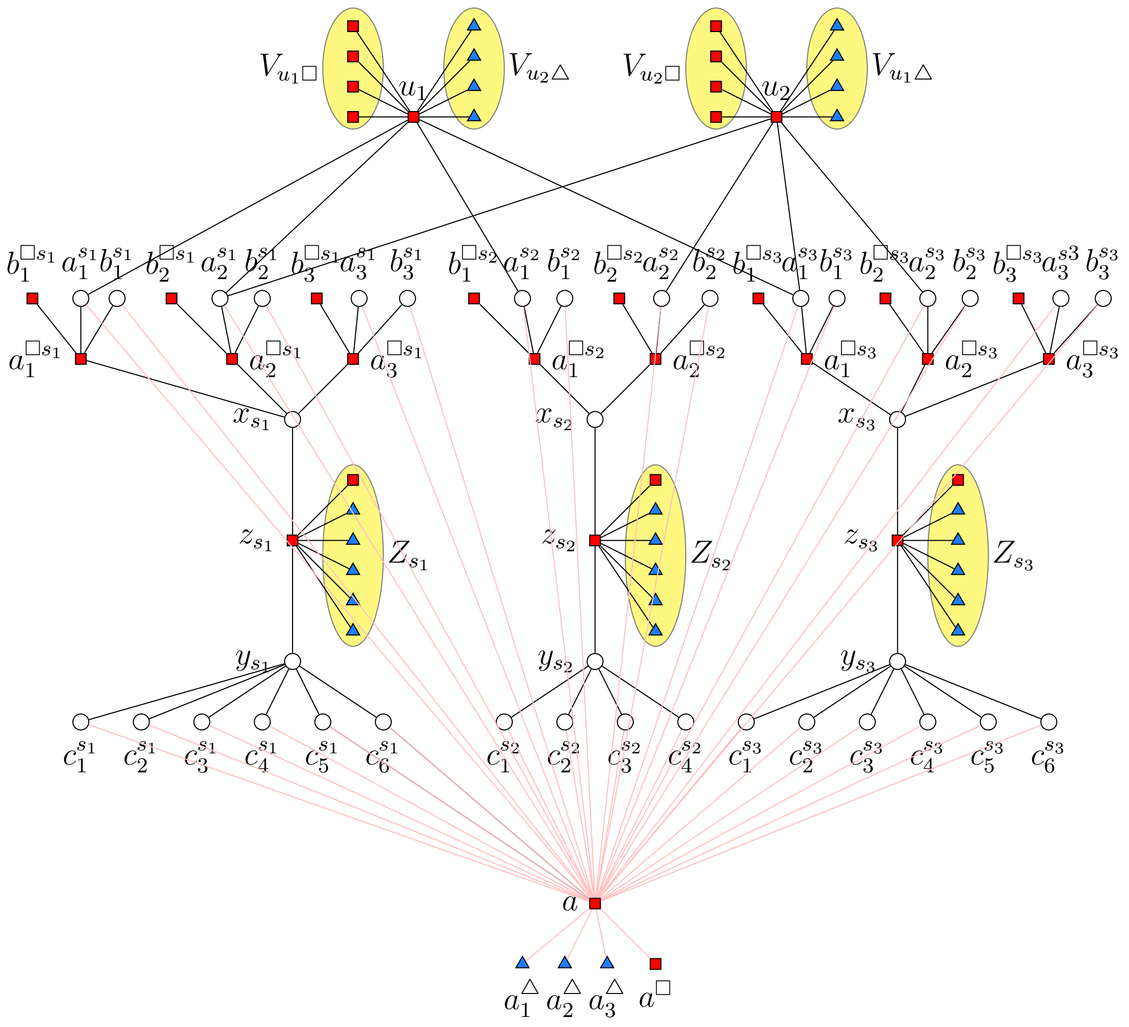

$V_{u_1\square}$
u_1
$V_{u_2\triangle}$
$V_{u_2\square}$
u_2
$V_{u_1\triangle}$
$b_1^{\square s_1}$ $a_1^{s_1}$ $b_1^{s_1}$ $b_2^{\square s_1}$ $a_2^{s_1}$ $b_2^{s_1}$ $b_3^{\square s_1}$ $a_3^{s_1}$ $b_3^{s_1}$
$b_1^{\square s_2}$ $a_1^{s_2}$ $b_1^{s_2}$ $b_2^{\square s_2}$ $a_2^{s_2}$ $b_2^{s_2}$
$b_1^{\square s_3}$ $a_1^{s_3}$ $b_1^{s_3}$ $b_2^{\square s_3}$ $a_2^{s_3}$ $b_2^{s_3}$ $b_3^{\square s_3}$ $a_3^{s_3}$ $b_3^{s_3}$
$a_1^{\square s_1}$ $a_2^{\square s_1}$ $a_3^{\square s_1}$
$a_1^{\square s_2}$ $a_2^{\square s_2}$
$a_1^{\square s_3}$ $a_2^{\square s_3}$ $a_3^{\square s_3}$
x_{s_1} x_{s_2} x_{s_3}
z_{s_1} Z_{s_1} z_{s_2} Z_{s_2} z_{s_3} Z_{s_3}
y_{s_1} y_{s_2} y_{s_3}
$c_1^{s_1}$ $c_2^{s_1}$ $c_3^{s_1}$ $c_4^{s_1}$ $c_5^{s_1}$ $c_6^{s_1}$
$c_1^{s_2}$ $c_2^{s_2}$ $c_3^{s_2}$ $c_4^{s_2}$
$c_1^{s_3}$ $c_2^{s_3}$ $c_3^{s_3}$ $c_4^{s_3}$ $c_5^{s_3}$ $c_6^{s_3}$
a
$a_1^{\triangle}$ $a_2^{\triangle}$ $a_3^{\triangle}$ $a^{\square}$

Figure 7.1: A schematic view of the construction of G in the proof of Lemma 7.2.2 for a MRSS instance $S = \{(2,1),(1,1),(1,2)\}$, $t = (3,3)$, $k = 2$ and $k' = 2$. We introduce $k = 2$ new vertices $\{u_1, u_2\}$ in G. For u_1, we introduce a set $V_{u_1\square}$ of $\sum\limits_{s\in S} s(1) = 2+1+1 = 4$ forbidden vertices and a set $V_{u_1\triangle}$ of $\sum\limits_{s\in S} s(1)-2t(1)+2 = 2\times4-2\times3+2 = 4$ necessary vertices. Similarly we introduce four forbidden and four necessary vertices for u_2. For the vector $s_1 = (2,1)$, we introduce a tree structure T_{s_1} with vertex set $A_{s_1} \cup B_{s_1} \cup A^{\square}_{s_1} \cup B^{\square}_{s_1} \cup C_{s_1} \cup Z_{s_1} \cup \left\{x_{s_1}, y_{s_1}, z_{s_1}\right\}$. As $\max(s_1) = 2$, $A_{s_1} = \{a_1^{s_1}, a_2^{s_1}, a_3^{s_1}\}$, $B_{s_1} = \{b_1^{s_1}, b_2^{s_1}, b_3^{s_1}\}$, $A^{\square}_{s_1} = \{a_1^{\square s_1}, a_2^{\square s_1}, a_3^{\square s_1}\}$, $B^{\square}_{s_1} = \{b_1^{\square s_1}, b_2^{\square s_1}, b_3^{\square s_1}\}$ have three vertices each; C_{s_1} has $2\max(s_1) + 2 = 6$ vertices. Similarly, for the vectors $s_2 = (1,1)$ and $s_3 = (1,2)$, we create tree structures T_{s_2} and T_{s_3} respectively. The edges incident to a are depicted in pink colour.

Let $s = (s(1), s(2), \ldots, s(k)) \in S$ and let $\max(s)$ denote the value of the largest coordinate of s. Set $N = \sum\limits_{s \in S}(2\max(s) + 2)$. We introduce three types of vertices: necessary vertices, forbidden vertices and normal vertices. We often indicate necessary vertices by means of a triangular node shape, forbidden vertices by means of a square node shape, and normal vertices by means of a circular node shape. We want to make sure that *necessary vertices* are always inside every solution, *forbidden vertices* are always outside every solution, and a *normal vertex* could be inside or outside the solution. The vertex set of the constructed graph G is defined as follows:

1. We introduce a set of k necessary vertices $U = \{u_1, u_2, \ldots, u_k\}$. For every $u_i \in U$, we create a set $V_{u_i\square}$ of $\sum\limits_{s \in S} s(i)$ forbidden vertices and a set $V_{u_i\triangle}$ of $2\sum\limits_{s \in S} s(i) - 2t(i) + 2$ necessary vertices.

2. For each vector $s = (s(1), s(2), \ldots, s(k)) \in S$, we introduce a tree T_s into G. The vertex set of tree T_s is defined as follows:

$$V(T_s) = A_s \cup B_s \cup A_s^\square \cup B_s^\square \cup C_s \cup Z_s \cup \Big\{x_s, y_s, z_s\Big\}$$

   where $A_s = \{a_1^s, \ldots, a_{\max(s)+1}^s\}$, $B_s = \{b_1^s, \ldots, b_{\max(s)+1}^s\}$, $A_s^\square = \{a_1^{\square s}, \ldots, a_{\max(s)+1}^{\square s}\}$, $B_s^\square = \{b_1^{\square s}, \ldots, b_{\max(s)+1}^{\square s}\}$ and $C_s = \{c_1^s, \ldots, c_{2\max(s)+2}^s\}$ are five sets of vertices, and the set $Z_s = \{z_1^{\triangle s}, z_2^{\triangle s}, z_3^{\triangle s}, z_4^{\triangle s}, z_5^{\triangle s}, z^{\square s}\}$ contains five necessary vertices and one forbidden vertex.

3. We introduce a vertex a and a set of four vertices $A = \{a_1^\triangle, a_2^\triangle, a_3^\triangle, a^\square\}$ containing three necessary vertices and one forbidden vertex.

We now create the edge set of G.

1. For every $u_i \in U$, make u_i adjacent to every vertex of $V_{u_i\square} \cup V_{u_i\triangle}$. For each $i \in \{1, 2, \ldots, k\}$ and for each $s \in S$, we make u_i adjacent to exactly $s(i)$ many vertices of A_s in arbitrary manner.

2. We now create the edge set of T_s.

$$E(T_s) = \{(x_s, z_s), (z_s, y_s)\} \cup \bigcup_{i=1}^{\max(s)+1} \Big\{(a_i^{\square s}, b_i^{\square s}), (a_i^{\square s}, a_i^{s}), (a_i^{\square s}, b_i^{s}), (x_s, a_i^{\square s})\Big\}$$
$$\cup \bigcup_{i=1}^{5}\{(z_s, z_i^{\triangle s}), (z_s, z^{\square s})\} \cup \bigcup_{i=1}^{2\max(s)+2} (y_s, c_i^{s})$$

3. Make a adjacent to every vertex of A. For each $s \in S$, we make a adjacent to every vertex of $A_s \cup B_s \cup C_s$.

We now define

$$V_\triangle = A \setminus \{a^\square\} \cup \bigcup_{i=1}^{k} V_{u_i\triangle} \cup \bigcup_{s\in S} Z_s \setminus \{z^{\square s}\}$$

and

$$V_\square = U \cup \{a, a^\square\} \cup \bigcup_{i=1}^{k} V_{u_i\square} \cup \bigcup_{s\in S} A_s^\square \cup B_s^\square \cup \{z_s, z^{\square s}\}.$$

We set $r = \sum\limits_{i=1}^{k} 2\Big(\sum\limits_{s\in S} s(i) - t(i) + 1\Big) + \sum\limits_{s\in S} 2(\max(s)+1) + 5n + 3 + k'$. Observe that if we remove the set $U \cup \{a\}$ of $k+1$ vertices from G, each connected component of the resulting graph is a tree with height at most 5. Note that I' can be constructed in polynomial time.

It remains to show that I is a yes instance if and only if I' is a yes instance. Towards showing the forward direction, let S' be a subset of S such that $|S'| \leq k'$ and $\sum\limits_{s\in S'} s \geq t$. We claim

$$R = V_\triangle \cup \bigcup_{s\in S'} \big(A_s \cup B_s \cup \{x_s\}\big) \cup \bigcup_{s\in S\setminus S'} C_s$$

is a strong offensive alliance of G such that $|R| \leq r$, $V_\triangle \subseteq R$, and $V_\square \cap R = \emptyset$. Observe that

$$N(R) = U \cup \{a\} \cup \bigcup_{s\in S'} \big(A_s^\square \cup \{z_s\}\big) \cup \bigcup_{s\in S\setminus S'} \{y_s\}.$$

Let $u_i \in U$, then we show that $d_R(u_i) \geq d_{R^c}(u_i) + 2$. The neighbours of u_i in R are the vertices of $V_{u_i\triangle}$ and $s(i)$ vertices of A_s for each $s \in S'$. As $\sum\limits_{s\in S'} s(i) - t(i) \geq 0$, we get

$$\begin{aligned}
d_R(u_i) &= \sum_{s\in S'} s(i) + |V_{u_i\triangle}| \\
&= \sum_{s\in S'} s(i) + \underbrace{2\sum_{s\in S} s(i) - 2t(i) + 2}_{\text{size of } V_{u_i\triangle}} \\
&= \underbrace{\Big(\sum_{s\in S'} s(i) - t(i)\Big)}_{\geq 0 \text{ by assumption}} + \sum_{s\in S} s(i) - t(i) + \sum_{s\in S} s(i) + 2 \\
&\geq \sum_{s\in S} s(i) - t(i) + \sum_{s\in S} s(i) + 2 \\
&= \sum_{s\in S\setminus S'} s(i) + \underbrace{\Big(\sum_{s\in S'} s(i) - t(i)\Big)}_{\geq 0 \text{ by assumption}} + \sum_{s\in S} s(i) + 2 \\
&\geq \sum_{s\in S\setminus S'} s(i) + \underbrace{\sum_{s\in S} s(i)}_{\text{size of } V_{u_i\square}} + 2 = \underbrace{\sum_{s\in S\setminus S'} s(i) + |V_{u_i\square}|}_{\text{the number of neighbours of } u_i \text{in } R^c} + 2 \\
&= d_{R^c}(u_i) + 2.
\end{aligned}$$

For the remaining vertices x in $N(R)$, it is easy to see that $d_R(x) \geq d_{R^c}(x) + 2$. Therefore, R is a strong offensive alliance.

Towards showing the reverse direction of the equivalence, suppose G has a strong offensive alliance R of size at most r such that $V_\triangle \subseteq R$ and $V_\square \cap R = \emptyset$. As $\bigcup\limits_{i=1}^{k} V_{u_i\triangle} \subseteq V_\triangle$, the vertices of $\bigcup\limits_{i=1}^{k} V_{u_i\triangle}$ are in the solution R. On the other hand, as $U \subseteq V_\square$, the vertices of U are not in R. Therefore it may be noted that $U \subseteq N(R)$. We know $V_\triangle$ contains $\sum\limits_{i=1}^{k}\Big(\sum\limits_{s\in S} 2s(i) - 2t(i)\Big) + 5n + 3$ vertices; thus besides the vertices of $V_\triangle$, there are at most $\sum\limits_{s\in S} 2(\max(s)+1) + k'$ vertices in R. As the necessary vertices $a_1^\triangle, a_2^\triangle, a_3^\triangle$ are in R and $a \in V_\square$ is not in R, we have $a \in N(R)$. Note that $N(a) = \{a_1^\triangle, a_2^\triangle, a_3^\triangle, a^\square\} \bigcup\limits_{s\in S}(A_s \cup B_s \cup C_s)$ and $d_G(a) = \sum\limits_{s\in S} 4(\max(s) + 1) + 4$. As a is adjacent to three necessary vertices, it must have at least $\sum\limits_{s\in S} 2(\max(s)+1)$ many neighbours in R from the set $\bigcup\limits_{s\in S}(A_s \cup B_s \cup C_s)$. It is to be noted that if a vertex from the set $A_s \cup B_s$ is in the solution then the whole set $A_s \cup B_s \cup \{x_s\}$

lie in the solution. Otherwise $v \in A_s^{\square} \subseteq N(R)$ will have $d_R(v) < d_{R^c}(v) + 2$ which is a contradiction as R is a strong offensive alliance. This shows that at most k' many sets of the form $A_s \cup B_s \cup \{x_s\}$ contribute to the solution as otherwise the size of solution exceeds r. Therefore, any strong offensive alliance R of size at most r can be transformed to another strong offensive alliance R_0 of size at most r as follows:

$$R_0 = V_{\triangle} \cup \bigcup_{x_s \in R} \Big(A_s \cup B_s \cup \{x_s\}\Big) \cup \bigcup_{x_s \in V(G)\setminus R} C_s$$

We define a subset $S' = \Big\{s \in S \mid x_s \in R_0\Big\}$. Clearly, $|S'| \leq k'$. We claim that $\sum\limits_{s\in S'} s(i) \geq t(i)$ for all $1 \leq i \leq k$. Assume for the sake of contradiction that $\sum\limits_{s \in S'} s(i) < t(i)$ for some $i \in \{1, 2, \ldots, k\}$. Then, we have

$$\begin{aligned}
d_{R_0}(u_i) &= \sum_{s\in S'} s(i) + |V_{u_i\triangle}| = \sum_{s\in S'} s(i) + \underbrace{2\sum_{s\in S} s(i) - 2t(i) + 2}_{\text{size of } V_{u_i\triangle}} \\
&= \underbrace{\sum_{s\in S'} s(i) - t(i)}_{<0 \text{ by assumption}} + \sum_{s\in S} s(i) - t(i) + \sum_{s\in S} s(i) + 2 \\
&< \sum_{s\in S} s(i) - t(i) + \sum_{s\in S} s(i) + 2 \\
&= \sum_{s\in S\setminus S'} s(i) + \underbrace{\Big(\sum_{s\in S'} s(i) - t(i)\Big)}_{<0 \text{ by assumption}} + \sum_{s\in S} s(i) + 2 \\
&< \sum_{s\in S\setminus S'} s(i) + \underbrace{\sum_{s\in S} s(i)}_{\text{size of } V_{u_i\square}} + 2 = \sum_{s\in S\setminus S'} s(i) + |V_{u_i\square}| + 2 \\
&= d_{R_0^c}(u_i) + 2
\end{aligned}$$

and we also know $u_i \in N(R_0)$, which is a contradiction to the fact that R_0 is a strong offensive alliance. This shows that I is a yes instance. $\square$

We have the following corollaries from Lemma 7.2.2.

Corollary 7.2.3. Strong Offensive Alliance$^{\text{FN}}$ is W[1]-hard when parameterized by the size of a vertex deletion set into trees of height at most 5, even when $|V_{\triangle}| = 1$.

Proof. Given an instance $I = (G, r, V_\triangle, V_\Box)$ of Strong Offensive Alliance$^{\text{FN}}$, we construct an equivalent instance $I' = (G', r', V'_\triangle, V'_\Box)$ with $|V'_\triangle| = 1$. See Figure 7.2 for an illustration.

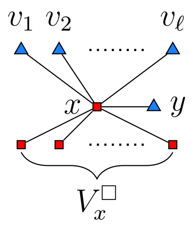


Figure 7.2: An illustration of the gadget used in the proof of Corollary 7.2.3.

The construction of G' starts with $G' := G$ and then add the following new vertices and edges. Let $v_1, v_2, \ldots, v_\ell$ be vertices of $V_\triangle$ where we assume that $\ell > 1$. We introduce two vertices x and y where x is a forbidden vertex and y is a necessary vertex; and make x and y adjacent. We make x adjacent to all the vertices in $V_\triangle$. We also introduce a set $V_{x\Box}$ of $\ell - 1$ forbidden vertices and make them adjacent to x. Set $r' = r + 1$. Define $V'_\triangle = \{y\}$ and $V'_\Box = \{x\} \cup V_{x\Box} \cup V_\Box$. We define G' as follows

$$V(G') = V(G) \cup \{x, y\} \cup V_{x\Box}$$

and

$$E(G') = E(G) \cup \Big\{(x, y), (x, \alpha), (x, \beta) \mid \alpha \in V_{x\Box}, \beta \in V_\triangle\Big\}.$$

Let H be a vertex deletion set of G into trees of height at most 5. Clearly, if H has at most k vertices then the set $H \cup \{x\}$ has at most $k + 1$ vertices and it is a vertex deletion set of G' into trees of height at most 5. We now claim that I and I' are equivalent instances.

Suppose G has a strong offensive alliance R of size at most r such that $V_\triangle \subseteq R$ and $V_\Box \cap R = \emptyset$. Clearly $R' = R \cup \{y\}$ is a strong offensive alliance of size at most $r + 1$ in G' such that $V'_\triangle \subseteq R'$ and $V'_\Box \cap R' = \emptyset$. Note that $x \in N_{G'}(R')$ and it satisfies $d_{R'}(x) = l + 1 \geq d_{R'^c}(x) + 2 = (l - 1) + 2$.

Conversely suppose G' has a strong offensive alliance R' of size at most $r + 1$ such that $y \in R'$ and $V'_\Box \cap R' = \emptyset$. We claim $\{v_1, v_2, \ldots, v_l\} \subseteq R'$. As y is in R' but its neighbour x is not in R', we have $x \in N(R')$ and it satisfies the condition $d_{R'}(x) \geq d_{R'^c}(x) + 2$. Note that $N(x) = \{v_1, v_2, \dots, v_l\} \cup \{y\} \cup V_x^{\square}$ and $N_{R'^c}(x) = V_x^{\square}$. As x has $l-1$ neighbours outside R' and x satisfies the condition $d_{R'}(x) \geq d_{R'^c}(x) + 2 = (l-1) + 2 = l + 1$, R' includes $\{v_1, v_2, \dots, v_l\} \cup \{y\}$. Define $R = R' \setminus \{y\}$. Clearly R is a strong offensive alliance of size at most r such that $V_{\triangle} = \{v_1, v_2, \dots, v_l\} \subseteq R$ and $R \cap V_{\square} = \emptyset$. □

We can get an analogous result for the exact variant.

Corollary 7.2.4. Exact Strong Offensive Alliance$^{\text{FN}}$ is W[1]-hard when parameterized by the size of a vertex deletion set into trees of height at most 5 even when $|V_{\triangle}| = 1$.

Next, we present an FPT reduction that eliminates necessary vertices.

Lemma 7.2.5. Offensive Alliance$^{\text{F}}$ is W[1]-hard when parameterized by the size of a vertex deletion set into trees of height at most 5.

Proof. To prove this we reduce from the Strong Offensive Alliance$^{\text{FN}}$ problem, which is W[1]-hard when parameterized by the size of a vertex deletion set into trees of height at most 5, even when $|V_{\triangle}| = 1$. See Corollary 7.2.3. Given an instance $I = (G, r, V_{\triangle} = \{x\}, V_{\square})$ of Strong Offensive Alliance$^{\text{FN}}$, we construct an instance $I' = (G', r', V'_{\square})$ of Offensive Alliance$^{\text{F}}$ the following way. See Figure 7.3 for an illustration. The construction of G' starts with $G' := G$ and then add some new vertices and edges. Let

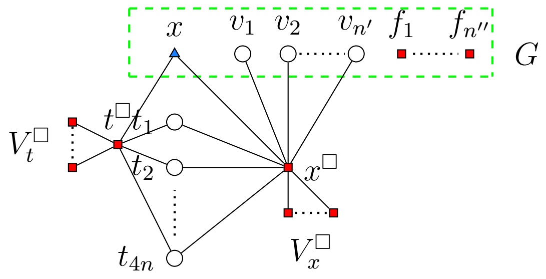


Figure 7.3: The reduction from Strong Offensive Alliance$^{\text{FN}}$ to Offensive Alliance$^{\text{F}}$ in Lemma 7.2.5. Note that the set $\{v_1, \dots, v_{n'}\}$ may contain forbidden vertices of degree greater than one.

n be the number of vertices in G. Suppose $V(G) = \{x, v_1, v_2, \dots, v_{n'}, f_1, \dots, f_{n''}\}$ where $F = \{f_1, \dots, f_{n''}\}$ is the set of degree one forbidden vertices in $V(G)$. Note that, we give special consideration to degree one forbidden vertices $f_1, \dots, f_{n''}$ in G. While constructing

G' from G, we refrain from introducing any new neighbors for these vertices. In other words, these vertices will retain their status as degree one forbidden vertices in G' as well. This is because, in each instance, we would like to maintain the property that each degree one forbidden vertex is adjacent to another forbidden vertex and each forbidden vertex of degree greater than one is adjacent to a degree one forbidden vertex. This structural property is exploited in the proof of Theorem 8.2.1 to get rid of forbidden vertices. We introduce two forbidden vertices $t^\square, x^\square$ into G'. We create a set $V_t^\square$ of $4n$ forbidden vertices and a set $V_x^\square$ of n forbidden vertices into G'. Finally we create a set $T = \{t_1, \ldots, t_{4n}\}$ of $4n$ vertices into G'. Make $t^\square$ adjacent to every vertex of $T \cup V_t^\square \cup \{x\}$ and make $x^\square$ adjacent to every vertex of $T \cup V_x^\square \cup V(G) \setminus F$. Set $r' = r + 4n$. We define G' as follows:

$$V(G') = V(G) \cup T \cup V_t^\square \cup V_x^\square \cup \{t^\square, x^\square\}$$

and

$$\begin{aligned} E(G') = &E(G) \cup \Big\{(t^\square, \alpha) \ : \ \alpha \in T \cup V_t^\square \cup \{x\}\Big\} \\ &\cup \Big\{(x^\square, \beta) \ : \ \beta \in T \cup V_x^\square \cup V(G) \setminus F\Big\} \end{aligned}$$

We define $V'_\square = V_\square \cup V_t^\square \cup V_x^\square \cup \{t^\square, x^\square\}$. Observe that there exists a set of at most $k+2$ vertices for G' whose deletion makes the resulting graph a forest containing trees of height at most 5. We can find such a set because there exists a vertex deletion set D of size at most k for G into trees of height at most 5. We just add $\{x^\square, t^\square\}$ to the set D, then the resulting set is of size $k+2$ whose deletion from G' makes the resulting graph a forest containing trees of height at most 5.

We now claim that I is a yes instance if and only if I' is a yes instance. Assume first that R is a strong offensive alliance of size at most r in G such that $\{x\} \subseteq R$ and $V_\square \cap R = \emptyset$. We claim $R' = R \cup T$ is an offensive alliance of size at most $r + 4n$ in G' such that $V'_\square \cap R' = \emptyset$. Clearly, $N(R') = \{t^\square, x^\square\} \cup N(R)$. If v is an element of $N(R)$, we know that $d_R(v) \geq d_{R^c}(v) + 2$ in G. Therefore in graph G', we get $d_{R'}(v) \geq d_{R'^c}(v) + 1$ for each $v \in N(R)$ due to the vertex $x^\square$. If $v \in \{x^\square, t^\square\}$, then the neighbours of v that are also in R' consist of the $4n$ elements of T and the elements of R; thus $d_{R'}(v) \geq 4n + 1$. The neighbours of $x^\square$ that are outside R' consist of the n elements of $V_x^\square$ and the elements of $V(G) \setminus (F \cup R)$; thus we get $d_{R'^c}(x^\square) < 2n$. The neighbours of $t^\square$ that are outside R' consist of $4n$ elements of $V_t^\square$

only; thus $d_{R'^c}(t^\square) = 4n$. Therefore $d_{R'}(v) \geq 4n + 1 \geq d_{R'^c}(v) + 1$. This shows that I' is a yes instance.

To prove the reverse direction of the equivalence, suppose R' is an offensive alliance of size at most $r' = r + 4n$ in G' such that $R' \cap V'_\square = \emptyset$. We claim that $T \cup \{x\} \subseteq R'$. Since R' is non empty, it must contain a vertex from the set $T \cup V(G) \setminus F$. Then $x^\square \in N(R')$ and it satisfies the condition $d_{R'}(x^\square) \geq d_{R'^c}(x^\square) + 1$. Due to n forbidden vertices in the set $V_x^\square$, node $x^\square$ must have at least $n + 1$ neighbours in R'. This implies that R' contains at least one vertex from T. Then $t^\square \in N(R')$ and it satisfies the condition $d_{R'}(t^\square) \geq d_{R'^c}(t^\square) + 1$. Since $|V_t^\square| = 4n$, the condition $d_{R'}(t^\square) \geq d_{R'^c}(t^\square) + 1$ forces the set $\{x\} \cup T$ to be inside the solution. Consider $R = R' \cap V(G)$. Clearly $|R| \leq r$, $x \in R$, $R \cap V_\square = \emptyset$ and we show that R is a strong offensive alliance in G. For each $v \in N(R') \cap V(G) = N(R)$, we have $N_{R'}(v) \geq N_{R'^c}(v) + 1$ in G'. Notice that we do not have $x^\square$ in G which is adjacent to all vertices in $N(R)$. Thus for each $v \in N(R)$, we get $N_R(v) \geq N_{R^c}(v) + 2$ in G. Therefore R is a strong offensive alliance of size at most r in G such that $x \in R$ and $R \cap V_\square = \emptyset$. This shows that I is a yes instance. $\square$

We have the following corollaries from Lemma 7.2.5.

Corollary 7.2.6. EXACT OFFENSIVE ALLIANCE$^{\text{F}}$ is W[1]-hard when parameterized by the size of a vertex deletion set into trees of height at most 5.

We are now ready to show our main hardness result for OFFENSIVE ALLIANCE using a reduction from OFFENSIVE ALLIANCE$^{\text{F}}$.

Theorem 7.2.7. OFFENSIVE ALLIANCE is W[1]-hard when parameterized by the size of a vertex deletion set into trees of height at most 7.

Proof. We give a parameterized reduction from OFFENSIVE ALLIANCE$^{\text{F}}$ which is W[1]-hard when parameterized by the size of a vertex deletion set into trees of height at most 5. See Lemma 7.2.5. Let $I = (G, r, V_\square)$ be an instance of OFFENSIVE ALLIANCE$^{\text{F}}$. Let $n = |V(G)|$. We construct an instance $I' = (G', r')$ of OFFENSIVE ALLIANCE the following way. We set $r' = r$. Recall that each degree one forbidden vertex is adjacent to another forbidden vertex and each forbidden vertex of degree greater than one is adjacent to a degree one forbidden vertex. Let u be a degree one forbidden vertex in G and u is adjacent to another forbidden vertex v. For each degree one forbidden vertex $u \in V_\square$, we introduce a tree T_u rooted at u

of height 2 as shown in Figure 7.4. The forbidden vertex v has additional neighbours from the original graph G which are not shown in the figure. We define G' as follows:

$$V(G') = V(G) \cup \bigcup_{u \in V_\square \ \& \ d_G(u)=1} V(T_u)$$

and

$$E(G') = E(G) \cup \bigcup_{u \in V_\square \ \& \ d_G(u)=1} E(T_u).$$

Clearly G' can be constructed in polynomial time. Let D be a vertex deletion set in G into

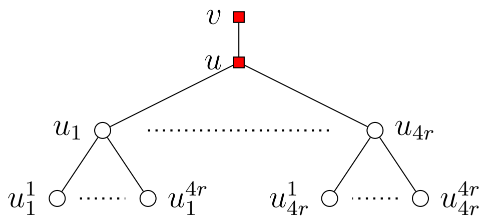


Figure 7.4: Our tree gadget T_u for each degree one forbidden vertex $u \in V_\square$

trees of height at most 5. Each component of $G' - D$ is a tree with height at most 7. It remains to show the equivalence between I and I'. It is easy to verify that if R is an offensive alliance of size at most r in G such that $R \cap V_\square = \emptyset$, then it is also an offensive alliance of size at most $r' = r$ in G'.

To prove the reverse direction of the equivalence, suppose that G' has an offensive alliance R' of size at most $r' = r$. We claim that no vertex from the set $V_\square \bigcup\limits_{u \in V_\square} V(T_u)$ is part of R'. It is to be noted that if any vertex from the set $V_\square \bigcup\limits_{u \in V_\square} V(T_u)$ is in R' then the size of R' exceeds $2r$. This implies that $R = R' \cap G$ is an offensive alliance such that $R \cap V_\square = \emptyset$ and $|R| \leq r$. This shows that I is a yes instance. □

We have the following consequences.

Corollary 7.2.8. Exact Offensive Alliance is W[1]-hard when parameterized by the size of a vertex deletion set into trees of height at most 7.

Clearly trees of height at most seven are trivially acyclic. Moreover, it is easy to verify that such trees have pathwidth [93] and treedepth [110] at most seven, which implies:

Theorem 7.2.9. Offensive Alliance and Exact Offensive Alliance are W[1]-hard when parameterized by any of the following parameters:

- the feedback vertex set number,
- the treewidth and pathwidth of the input graph,
- the treedepth of the input graph.

7.3 FPT Lower Bound Parameterized by Solution Size

We know that Offensive Alliance admits an FPT algorithm when parameterized by the solution size [50]. The algorithm in [50] uses branching technique and solves the problem in $\mathcal{O}^*(2^{\mathcal{O}(k\log k)})$ time. We prove that this running time is essentially optimal assuming ETH. Hardness for Offensive Alliance follows from a reduction from $k \times k$ (Permutation) Hitting Set with Thin Sets. In $k \times k$ Hitting Set, we restrict the universe to a $k \times k$ table. We define the problem as follows:

$k \times k$ Hitting Set
Input: A family $\mathcal{F}$ of sets $F_1, F_2, \ldots, F_m \subseteq [k] \times [k]$.
Question: Is there a set $X \subseteq [k] \times [k]$ containing exactly one element from each row such that $X \cap F_i \neq \emptyset$ for any $1 \leq i \leq m$?

In the $k \times k$ (Permutation) Hitting Set problem, we are given a family $\mathcal{F}$ of subsets of $[k] \times [k]$, and we would like to find a set X, consisting of one element from each row and induces a permutation of $[k]$, such that $X \cap F \neq \emptyset$ for each $F \in \mathcal{F}$. In the *thin set* variant we assume that each $F \in \mathcal{F}$ contains at most one element from each row. In the proof, we will use the fact that $k \times k$ (Permutation) Hitting Set with Thin Sets cannot be solved in time $2^{o(k\log k)}$, unless ETH fails [101].

Theorem 7.3.1. *Unless ETH fails,* Offensive Alliance *cannot be solved in time* $\mathcal{O}^*(2^{o(k\log k)})$, *where* k *is the solution size.*

Proof. We provide a polynomial-time algorithm that takes an instance $(\mathcal{F}, k)$ of $k \times k$ (Permutation) Hitting Set with Thin Sets, and outputs an equivalent instance (G, r) of

Offensive Alliance with $r = 5k$. We construct G the following way.

1. For every $F \in \mathcal{F}$, we introduce a vertex v_F into G. Let $V_{\mathcal{F}} = \{v_F \mid F \in \mathcal{F}\}$. We also introduce a set of k^2 vertices $W = \{w_{i,j} \ : \ i \in [k], j \in [k]\}$. Make v_F adjacent to w_{ij} if $(i,j) \in F$.

2. We introduce a clique $D^{\triangle}$ of size $4k$ into G. For every $d \in D^{\triangle}$, we add a set of $10k$ vertices and make them adjacent to d. For every $F \in \mathcal{F}$, we make v_F adjacent to every vertex of $D^{\triangle}$.

3. We introduce another clique $D^{\square}$ of size $12k+1$ into G. Let $d_F = d_W(v_F)$, the number of neighbours of v_F in W. As we are dealing with thin sets, we have $d_F \le k$ for all $F \in \mathcal{F}$. For every $F \in \mathcal{F}$, we make v_F adjacent to any $4k - d_F + 1$ vertices of $D^{\square}$.

4. For every row $i \in [k]$, create a vertex r_i into G and make r_i adjacent to w_{ij} for all $j \in [k]$. Let $R = \{r_1, \ldots, r_k\}$. For every $r \in R$, make r adjacent to every vertex of $D^{\triangle}$ and any $3k+1$ vertices of $D^{\square}$.

5. For every column $j \in [k]$, create a vertex c_j into G and make c_j adjacent to w_{ij} for all $i \in [k]$. Let $C = \{c_1, \ldots, c_k\}$. For every $c \in C$, make c adjacent to every vertex of $D^{\triangle}$ and any $3k+1$ vertices of $D^{\square}$.

This completes the construction of G. Set $r = 5k$. We now formally argue that instances $(\mathcal{F}, k)$ and (G, r) are equivalent. Assume first that X is a solution to the instance $(\mathcal{F}, k)$. For each $i \in [k]$, let j_i be the unique index such that $(i, j_i) \in X$. We claim that the set

$$S = D^{\triangle} \cup \{w_{1j_1}, w_{2j_2}, \ldots, w_{kj_k}\}$$

is an offensive alliance of size exactly $5k$ in G. We see that $N(S) = R \cup C \cup \{v_F \ : \ F \in \mathcal{F}\}$. Let v be an arbitrary element of $N(S)$. We need to prove that $d_S(v) \ge d_{S^c}(v) + 1$ for all $v \in N(S)$. If v is an element of R or C, the neighbours of v in S are the elements of $D^{\triangle}$ and one element from W. Thus we have $d_S(v) = 4k+1$. The neighbours of v in S^c are $3k+1$ elements of $D^{\square}$ and $k-1$ elements of W; therefore we have $d_{S^c}(v) = 4k$. If v is an element of $\{v_F \ : \ F \in \mathcal{F}\}$, the neighbours of v in S are the elements of $D^{\triangle}$ and at least one element from W as X is a hitting set; thus we have $d_S(v) \ge 4k+1$. The neighbours of v in S^c are $4k - d + 1$ elements of $D^{\square}$ and at most $d-1$ elements from W; thus we have $d_{S^c}(v) \le (4k - d + 1) + (d - 1) = 4k$. This shows that S is indeed an offensive alliance.

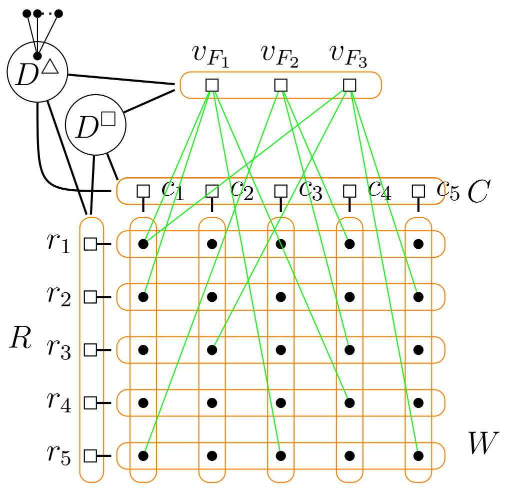


Figure 7.5: Example of the reduction in Theorem 7.3.1 applied to an instance ($\mathcal{F} = \{F_1, F_2, F_3\}, 5$) of 5 × 5 Permutation Hitting Set with Thin Sets that has three sets $F_1 = \{(1,1),(2,1),(4,4),(5,3)\}, F_2 = \{(1,4),(3,4),(5,1)\}, F_3 = \{(1,1),(2,5),(3,2),(5,5)\}$ with each set containing at most one element from each row.

In the reverse direction, let S be an offensive alliance of size at most $5k$ in G. First we show that it can be assumed that $N(S) \cap D^{\square} = \emptyset$. Suppose, for the sake of contradiction, that $v \in N(S) \cap D^{\square}$. Then v must satisfy the condition $d_S(v) \geq d_{S^c}(v) + 1$. As v has degree at least $12k$, in order to satisfy the condition $d_S(v) \geq d_{S^c}(v)+1$, the size of S must be at least $6k$, a contradiction to the assumption that the size of S is at most $5k$. Next we show that it can be assumed that $S \cap D^{\square} = \emptyset$. Suppose that S contains an element of $D^{\square}$. As $D^{\square}$ is a clique and $D^{\square} \cap N(S) = \emptyset$, if S contains one element of $D^{\square}$ then $D^{\square} \subseteq S$. This is not possible as $D^{\square}$ has $12k+1$ elements and S has at most $5k$ elements. Therefore, we may assume that $(S \cup N(S)) \cap D^{\square} = \emptyset$. This in turn implies that that $(R \cup C \cup V_{\mathcal{F}}) \cap S = \emptyset$. As the offensive alliance S is non-empty, we have $S \cap D^{\triangle} \neq \emptyset$ or $S \cap W \neq \emptyset$. Note that in either case, we get $N(S) \cap (R \cup C \cup V_{\mathcal{F}}) \neq \emptyset$. Let u be an arbitrary element of $N(S) \cap (R \cup C \cup V_{\mathcal{F}})$. If $S \cap D^{\triangle} = \emptyset$ then clearly we have $d_S(u) < d_{S^c}(u) + 1$ which is not possible. Therefore $S \cap D^{\triangle} \neq \emptyset$. We observe that if S contains one element of $D^{\triangle}$ then it contains all elements of $D^{\triangle}$, that is, $D^{\triangle} \subseteq S$. As $D^{\triangle} \subseteq S$ and $(R \cup C \cup V_{\mathcal{F}}) \cap S = \emptyset$, we get $(R \cup C \cup V_{\mathcal{F}}) \subseteq N(S)$. As S is an offensive alliance, every element u of $N(S)$ has to satisfy the condition $d_S(u) \geq d_{S^c}(u) + 1$. Consider an arbitrary vertex r_i of R. If $S \cap \{w_{ij} : j \in [k]\} = \emptyset$ then $d_S(r_i) = 4k$ and $d_{S^c}(r_i) = 4k$ which is not possible as r_i does not satisfy the condition $d_S(r_i) \geq d_{S^c}(r_i) + 1$. This implies that, for each $i \in [k]$, S contains at least one element from $\{w_{ij} : j \in [k]\}$ but since $|S| \leq 5k$ and $D^{\triangle} \subseteq S$, S contains exactly one element from $\{w_{ij} : j \in [k]\}$.

Using the same argument for an arbitrary vertex $c_j \in C$, we get that S contains exactly one element from $\{w_{ij} \ : \ i \in [k]\}$. We claim that $X = \{(i,j) \mid w_{ij} \in S\}$ is a permutation hitting set of size k. Let us assume that there exists a set $F \in \mathcal{F}$ which is not hit by X. In that case, we have $d_S(v_F) = 4k$ and $d_{S^c}(v_F) = (4k - d_F) + d_F = 4k$. This means that $d_S(v_F) < d_{S^c}(v_F) + 1$ which is a contradiction. Therefore, X is a hitting set of size k. As X has exactly one element in each row and in each column, X is a permutation hitting set.

An algorithm solving Offensive Alliance in time $2^{o(k \log k)}$ would therefore translate into an algorithm running in time $2^{o(k \log k)}$ for $k \times k$ (Permutation) Hitting Set with Thin Sets and contradicts the ETH. □

Corollary 7.3.2. *Unless ETH fails,* Exact Offensive Alliance *problem cannot be solved in time $\mathcal{O}^*(2^{o(k \log k)})$, where k is the solution size.*

7.4 No polynomial kernel parameterized by solution size and vertex cover

Parameterized by the solution size, the problem is FPT and in this section we show the following kernelization hardness of Offensive Alliance.

Theorem 7.4.1. Offensive Alliance parameterized by the solution size and vertex cover combined does not admit a polynomial compression unless coNP $\subseteq$ NP/poly.

To prove Theorem 7.4.1, we give a polynomial parameter transformation (PPT) from Closest String to Offensive Alliance parameterized by the solution size. In the Closest String problem we are given an alphabet Σ, a set of strings $\mathcal{X} = \{x_1, x_2, \ldots, x_k\}$ over Σ such that $|x_i| = n$ and an integer d. The objective is to check whether there exists a string x over Σ such that $d_H(x, x_i) \leq d, \ i \in \{1, \ldots, k\}$, where $d_H(x,y)$ denotes the number of places strings x and y differ at. Let x be a string over alphabet Σ. We denote the letter on the p-th position of x as $x[p]$. Thus $x = x[1]x[2]\ldots x[n]$ for a string of length n. We say string x and y *differ* on the p-th position if $x[p] \neq y[p]$. The following theorem is known:

Theorem 7.4.2. *[8]* Closest String *parameterized by the distance d and the length of the strings n, does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly.*

They also observe that the kernelization lower bound for Closest String works for any fixed alphabet Σ of size at least two. Therefore, without loss of generality, we assume that $\Sigma = \{A_1, A_2\}$.

7.4.1 Proof of Theorem 7.4.1

We give a PPT from the Closest String problem. Given an instance $(\mathcal{X}, d)$ of the Closest String problem, we construct an instance (G, r) of Offensive Alliance the following way.

1. For every $x \in \mathcal{X}$, we introduce a vertex v_x into G. Let $V_{\mathcal{X}} = \{v_x \mid x \in \mathcal{X}\}$. We also introduce a set of $2n$ vertices $W = \{w_{i,j} \ : \ i \in [n], j \in [2]\}$. Make v_x adjacent to w_{i1} if the letter on the ith position of x is A_1; make v_x adjacent to w_{i2} if the letter on the ith position of x is A_2.
2. We introduce a clique $D^{\triangle}$ of size $3n + 2d + 1$ into G. For every $d \in D^{\triangle}$, we add a set V_d of $12n$ vertices and make them adjacent to d. For every $v_x \in V_{\mathcal{X}}$, we make v_x adjacent to every vertex of $D^{\triangle}$.
3. We introduce another clique $D^{\square}$ of size $12n + 1$ into G. For every $v_x \in V_{\mathcal{F}}$, we make v_x adjacent to any $4n$ vertices of $D^{\square}$.
4. For every row $i \in [n]$, create a vertex r_i into G and make r_i adjacent to w_{i1} and w_{i2}. Let $R = \{r_1, \ldots, r_n\}$. For every $r \in R$, make r adjacent to any three vertices of $D^{\triangle}$ and any two vertices of $D^{\square}$.

This completes the construction of G. Note that the set $R \cup W \cup D^{\square} \cup D^{\triangle}$ forms a vertex cover of G of size $18n + 2d + 2$. We set $r = 4n + 2d + 1$. It is easy to see that the above construction takes polynomial time. We now formally argue that instances $(\mathcal{X}, d)$ and (G, r) are equivalent. Assume first that y is a solution to the instance $(\mathcal{X}, d)$, that is, $d_H(x_i, y) \leq d$ for all $i \in \{1, 2, \ldots, k\}$. We claim that

$$S = D^{\triangle} \cup \Big\{ w_{ij} \mid y[i] = A_j \text{ for } i = 1, 2, \ldots, n \Big\}$$

is an offensive alliance of size at most r. We see that $N(S) = R \cup V_{\mathcal{X}}$. We need to prove that $d_S(v) \geq d_{S^c}(v) + 1$ for every $v \in R \cup V_{\mathcal{X}}$. If r is an element of R, the neighbours of

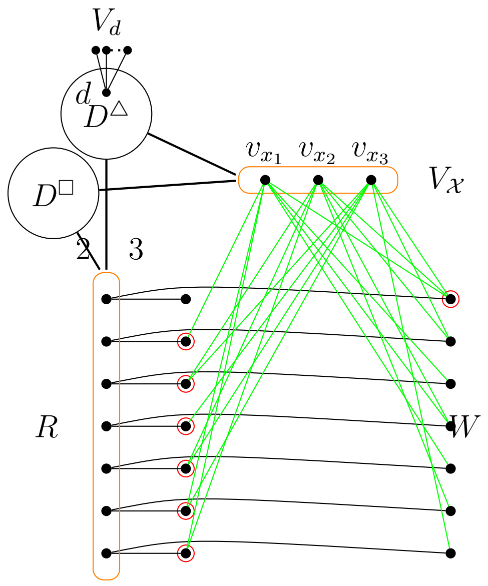


Figure 7.6: Example of reduction of Theorem 7.4.1 applied to an instance (χ, d) of CLOSEST STRING where χ contains three strings $x_1 = 1011100, x_2 = 1101010, x_3 = 1110001$ and $d = 3$. A solution string $y = 1000000$ is shown in red circles.

r in S are three elements of $D^{\triangle}$ and one element from W. Thus we have $d_S(r) = 4$. The neighbours of r in S^c are two elements of $D^{\square}$ and one elements of W; therefore we have $d_{S^c}(r) = 3$ and r satisfies the required condition. If v_x is an element of $V_{\mathcal{X}}$, the neighbours of v_x in S are $3n + 2d + 1$ elements of $D^{\triangle}$ and $n - d_H(x, y)$ element from W. Thus we have $d_S(v_x) \geq (3n + 2d + 1) + (n - d) = 4n + d + 1$. The neighbours of v_x in S^c are $4n$ elements of $D^{\square}$ and $d_H(x, y)$ element from W; hence $d_{S^c}(v_x) = 4n + d_H(x, y) \leq 4n + d$. Therefore v_x satisfies the required condition.

In the reverse direction, let S be an offensive alliance of size at most $4n + 2d + 1$ in G. First we show that it can be assumed that $N(S) \cap D^{\square} = \emptyset$. Suppose, for the sake of contradiction, that $v \in N(S) \cap D^{\square}$. Then v must satisfy the condition $d_S(v) \geq d_{S^c}(v) + 1$. As v has degree at least $12k$, in order to satisfy the condition $d_S(v) \geq d_{S^c}(v) + 1$, the size of S must be at least $6k$, a contradiction to the assumption that the size of S is at most $4n+2d+1$. Next we show that it can be assumed that $S \cap D^{\square} = \emptyset$. Suppose that S contains an element of $D^{\square}$. As $D^{\square}$ is a clique and $D^{\square} \cap N(S) = \emptyset$, if S contains one element of $D^{\square}$ then $D^{\square} \subseteq S$. This is not possible as $D^{\square}$ has $12n+1$ elements and S has at most $4n+2d+1$ elements. Therefore, we may assume that $(S \cup N(S)) \cap D^{\square} = \emptyset$. This in turn implies that S does not contain any element from $V_{\mathcal{X}} \cup R$. As the offensive alliance S is non-empty, we have $S \cap D^{\triangle} \neq \emptyset$ or $S \cap W \neq \emptyset$. Note that in either case, we get $N(S) \cap (R \cup V_{\mathcal{X}}) \neq \emptyset$. Let u be an arbitrary element of $N(S) \cap (R \cup V_{\mathcal{X}})$. If $S \cap D^{\triangle} = \emptyset$ then clearly we have $d_S(u) < d_{S^c}(u) + 1$ which is not possible. Therefore $S \cap D^{\triangle} \neq \emptyset$. We observe that if S contains one element of $D^{\triangle}$ then it contains all elements of $D^{\triangle}$, that is, $D^{\triangle} \subseteq S$. As $D^{\triangle} \subseteq S$ and $(R \cup V_{\mathcal{X}}) \cap S = \emptyset$, we get $(R \cup V_{\mathcal{X}}) \subseteq N(S)$. As S is an offensive alliance, every element u of $N(S)$ has to satisfy the condition $d_S(u) \geq d_{S^c}(u) + 1$. Consider an arbitrary vertex r_i of R. If $S \cap \{w_{i1}, w_{i2}\} = \emptyset$ then $d_S(r_i) = 3$ and $d_{S^c}(r_i) = 4$ which is not possible as r_i does not satisfy the condition $d_S(r_i) \geq d_{S^c}(r_i) + 1$. This implies that, for each $i \in [n]$, S contains at least one element from $\{w_{i1}, w_{i2}\}$. Since $|S| \leq 4n + 2d + 1$ and $D^{\triangle} \subseteq S$, S contains exactly one element from $\{w_{i1}, w_{i2}\}$ for each i. Define a string $y = y[1]y[2]\dots y[n]$, where $y[i] = 1$ if $w_{i1} \in S$ and $y[i] = 2$ if $w_{i2} \in S$. We claim y is a central string. Assume, for the sake of contradiction, that there exists a string x_i such that $d_H(x_i, y) > d$. In this case, we see that $d_S(v_{x_i}) < (3n + 2d + 1) + n - d \leq 4n + d$ and $d_{S^c}(v_{x_i}) > 4n + d$. Therefore, $d_S(v_{x_i}) < d_{S^c}(v_{x_i}) + 1$, which is a contradiction. $\square$

7.5 Faster FPT algorithms parameterized by vertex cover number

We know that both Offensive Alliance and Strong Offensive Alliance admit FPT algorithms [92] when parameterized by the vertex cover number of the input graph. The algorithms in [92] use Integer Linear Programming, and thus their dependency on the parameter may be gigantic. The reason is this. The Offensive Alliance problem is mapped to an ILP with at most $2^{\texttt{vc(G)}}$ many variables where $\texttt{vc(G)}$ is the vertex cover number of the input graph. It is proved in [47] that p-Variable Integer Linear Programming Optimization (p-Opt-ILP) can be solved using $O(p^{2.5p+o(p)} \cdot L \cdot log(MN))$ arithmetic operations and space polynomial in L. Thus the algorithm in [92] requires $\mathcal{O}^*((2^{\texttt{vc(G)}})^{\mathcal{O}(2^{\texttt{vc(G)}})})$ time. The natural question would be whether they admit $\mathcal{O}^*(\texttt{vc(G)}^{\mathcal{O}(\texttt{vc(G)})})$ time algorithm. We answer this question with the following theorem.

Theorem 7.5.1. Offensive Alliance can be solved in time $\mathcal{O}^*(\texttt{vc(G)}^{\mathcal{O}(\texttt{vc(G)})})$ where $\texttt{vc(G)}$ is the vertex cover number of the input graph G.

Proof. Let C be a vertex cover of G of size $\texttt{vc(G)}$. Note that C forms an offensive alliance.

This is because $N(C)$ is an independent set and every vertex of $N(C)$ has no neighbours in C^c and has at least one neighbour in C. Therefore we have $d_C(v) \geq d_{C^c}(v) + 1$ for all $v \in N(C)$, and hence C is an offensive alliance. This implies that the size of minimum offensive alliance is at most $\texttt{vc(G)}$. In [50], it was proved using branching technique that Offensive Alliance problem parameterized by solution size admits a $\mathcal{O}(n^2 k(2k)^{k-1})$. This implies that, we have an algorithm with running time $\mathcal{O}^*(\texttt{vc(G)})^{\texttt{vc(G)}}$. □

The arguments in the proof of Theorem 7.5.1 is also applicable to Strong Offensive Alliance as long as the input graph G has minimum degree at least two. As a direct consequence of Theorem 7.5.1, we have the following corollary.

Corollary 7.5.2. Strong Offensive Alliance can be solved in time $\mathcal{O}^*(\texttt{vc(G)}^{\mathcal{O}(\texttt{vc(G)})})$ where $\texttt{vc(G)}$ is the vertex cover number of the input graph G with $\delta(G) \geq 2$.

7.6 FPT algorithm parameterized by vertex integrity

In this section, we present an FPT algorithm for Offensive Alliance parameterized by vertex integrity. Lets begin by recalling the definition of vertex integrity.

Definition 7.6.1. The *vertex integrity* of a graph G, denoted $\texttt{vi}(G)$, is the minimum integer k satisfying that there is $X \subseteq V(G)$ such that $|X| + |V(C)| \leq k$ for each component C of $G - X$. We call such X a $\texttt{vi}(k)$-set of G.

Theorem 7.6.1. Offensive Alliance *is fixed-parameter tractable when parameterized by the vertex integrity.*

Proof. Let $G = (V, E)$ be a graph and $X \subseteq V$ be a $\texttt{vi}(k)$-set of G. Then $\mathcal{C} = G - X$ is a collection of disjoint components, that is $\mathcal{C} = \{C_1, C_2, \ldots\}$ such that $|X| + |C_i| \leq k$ for all i. We know $\mathcal{C}$ can be partitioned into equivalent classes $\mathcal{C}_1, \mathcal{C}_2, \ldots$. Let C_l be a representative of the equivalence class $\mathcal{C}_l$ and let $v \in C_l$. Note that v has neighbours only in $X \cup C_l$, that is, $N(v) \subseteq X \cup C_l$. We next guess $P = S \cap X$ and $Q = X \cap N(S)$, where S is a smallest offensive alliance. There are at most $3^{|X|} \leq 3^k$ candidates for (P, Q) as each member of X has three options: either in P, Q or $X \setminus (P \cup Q)$. We reduce the problem of finding the rest of S to an integer linear programming (ILP). We say $A_l \subseteq C_l$ is a *valid selection* from C_l if and only if each $u \in N(P \cup A_l) \cap C_l$ satisfies

$$|N_{P\cup A_l}(u)| \geq \frac{d(u)+1}{2}$$

$$N(A_l) \cap (X \setminus (P \cup Q)) = \emptyset.$$

Every component in $\mathcal{C}_l$ contributes some vertices in the solution S. We introduce a variable $y(A_l)$ which denotes the number of components in the equivalence class $\mathcal{C}_l$ that contribute A_l (up to isomorphism) to S. In the following, we present ILP formulation of offensive alliance. The objective is to minimize

$$|P| + \sum_{l} \sum_{A_l \subseteq C_l} y(A_l)|A_l|.$$

For each equivalence class $\mathcal{C}_l$, we have the following constraint $\sum\limits_{A_l \subseteq C_l} y(A_l) = |\mathcal{C}_l|$ where $|\mathcal{C}_l|$ denotes the number of components in $\mathcal{C}_l$. Next we add constraints to make sure that each vertex $u \in Q$ satisfies $|N_S(u)| \geq |N_{S^c}(u)| + 1$, that is, $N_S(u) \geq \frac{d(u)+1}{2}$. Note that

$$|N_S(u)| = |N_P(u)| + \sum_{l} \sum_{A_l \subseteq C_l} |N_{A_l}(u)| \times y(A_l).$$

Therefore for each vertex $u \in Q$, we have the constraint:

$$|N_P(u)| + \sum_{l} \sum_{A_l \subseteq C_l} |N_{A_l}(u)| \times y(A_l) \geq \frac{d(u)+1}{2}.$$

In the following, we present ILP formulation of offensive alliance problem, where P and Q are given:

$$\text{Minimize } |P| + \sum_{l} \sum_{A_l \subseteq C_l} y(A_l)|A_l|$$

Subject to

$$\sum_{A_l \subseteq C_l} y(A_l) = |\mathcal{C}_l| \qquad \forall\, l$$

$$N(A_l) \cap (X \setminus (P \cup Q)) = \emptyset \qquad \forall\, l,\ \forall\, A_l \subseteq C_l$$

$$|N_{P \cup A_l}(u)| \geq \frac{d(u)+1}{2} \qquad \forall\, l,\ \forall\, A_l \subseteq C_l,\ \forall\, u \in N(P \cup A_l) \cap C_l,$$

$$|N_P(u)| + \sum_{l} \sum_{A_l \subseteq C_l} |N_{A_l}(u)| \times y(A_l) \geq \frac{d(u)+1}{2} \qquad \forall u \in Q$$

In the formulation for Offensive Alliance, we have at most $2^{O(k^2)}$ variables. The value of objective function is bounded by n and the value of any variable $y(A_l)$ in the integer linear programming is bounded by n. Each constraint can be represented using at most $O(2^{O(k^2)} \log n)$ bits. Lemma 2.3.2 implies that we can solve the problem with the guess (P, Q) in FPT time. There are at most 3^k choices for (P, Q) and the ILP formula for a given guess can be solved in FPT time. Thus Theorem 7.6.1 holds.

7.7 Classical lower bounds under ETH

The brute-force approach to find a minimum size offensive alliance is to evaluate all $2^n - 1$ possible non-empty subsets and check for each one whether it is an offensive alliance or not. Checking if a specific set of vertices is an offensive alliance takes $O(n^2)$ time. So the running time of this algorithm is $O^*(2^n)$. We use O^* notation to hide polynomial factors (in the input size) in the running time. In this section, we give an algorithmic lower bound for the Offensive Alliance problem using exponential time hypothesis. In computational complexity theory, the exponential time hypothesis (ETH) is an unproven computational hardness assumption that was formulated by Impagliazzo and Paturi (1999). The hypothesis states that 3-SAT cannot be solved in subexponential time in the worst case. While Offensive Alliance can be solved in time $\mathcal{O}^*(2^n)$ for general graphs on n vertices, we now prove that the existence of algorithms with running time $2^{o(n)}$ is unlikely. In order to prove that a too fast algorithm for Offensive Alliance contradicts ETH, we give a reduction from Vertex Cover in graphs of maximum degree 3 and argue that a too fast algorithm for Offensive Alliance would solve Vertex Cover in graphs of maximum degree 3 in time $2^{o(n)}$. Johnson and Szegedy [85] proved that, assuming ETH, there is no algorithm with running time $2^{o(n)}$ to compute a minimum vertex cover in graphs of maximum degree 3.

7.7.1 Offensive Alliance on bipartite graphs

Theorem 7.7.1. Unless ETH fails, Offensive Alliance does not admit a $2^{o(n)}$ algorithm, even when restricted to bipartite graphs.

Proof. We give a linear reduction from Vertex Cover in graphs of maximum degree 3 to Offensive Alliance, that is, a polynomial-time algorithm that takes an instance of Vertex Cover and outputs an equivalent instance of Offensive Alliance whose size is bounded by $O(n)$. Let (G, k) be an instance of Vertex Cover, where $G = (V, E)$ has maximum degree 3. We construct an equivalent instance (G', k') of Offensive Alliance the following way. See Figure 7.7 for an illustration. Take two distinct copies V_0, V_1 of

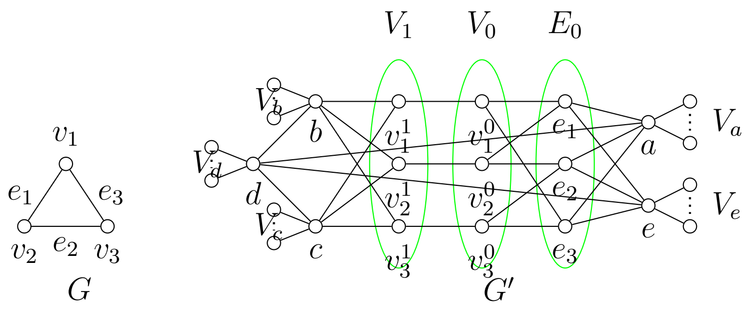


Figure 7.7: An illustration of the reduction from Vertex Cover to Offensive Alliance in Theorem 7.7.1.

$V = \{v_1, v_2, \ldots, v_n\}$, and let v^i be the copy of $v \in V$ in V_i. We introduce the vertex set E_0 into G', where $E_0 = \{e_1, \ldots, e_m\}$, the edge set of G. We make v_i^0 adjacent to e_j if and only if v_i is an endpoint of e_j in G. We make v_i^0 adjacent to v_i^1 for all i. Next, introduce five new vertices a, b, c, d, e. For each $x \in \{a, b, c, d, e\}$, introduce a set V_x of $4k'$ vertices and make x adjacent to every vertex of V_x. Moreover, we make vertex a and e adjacent to every vertex of

E_0 and make b and c adjacent to every vertex of V_1. We also make d adjacent to every vertex of $\{a, b, c, e\}$. Note that G' is a bipartite graph with bipartition $\{d\} \cup V_1 \cup E_0 \bigcup_{x \in \{a,b,c\}} V_x$ and $\{a, b, c, e\} \cup V_d \cup V_0$. We set $k' = k + 5$. Clearly, the size of G' is bounded by $O(n)$.

We claim that (G, k) is a yes-instance of Vertex Cover if and only if (G', k') is a yes-instance of Offensive Alliance. Suppose G has a vertex cover S of size at most k. We show that $D = \{v^0 \in V_0 \mid v \in S\} \cup \{a, b, c, d, e\}$ is an offensive alliance of size at most k' in G'. We see $N(D) = E_0 \cup V_1 \bigcup_{x \in \{a,b,c,d\}} V_x$. It is clear that for each $v \in V_1 \bigcup_{x \in \{a,b,c,d\}} V_x$, we have $d_D(v) \geq d_{D^c}(v) + 1$. Each $v \in E_0$ has at least three neighbours in D, more precisely, a, e and at least one neighbour in V_0. This implies that for each $v \in E_0$, we have $d_D(v) \geq d_{D^c}(v) + 1$.

Conversely, assume that G' admits an offensive alliance D of size at most $k' = k + 5$. Note that the vertices a, b, c, d and e cannot be part of the set $N(D)$ as each of them has degree $4k'$; otherwise the size of D will exceed k'. We now show that $\{a, b, c, d, e\} \subseteq D$. By definition, offensive alliance cannot be empty; therefore it must contain a vertex from $V(G')$.
Case 1: Suppose D contains an arbitrary vertex of $\bigcup_{x \in \{a,b,c,d,e\}} V_x$. Without loss of generality, assume that D contains an arbitrary vertex of V_a. Since $a \notin N(D)$, it implies that $a \in D$. Since $a \in D$, we get b, c, d and e also lie in D as otherwise $\{b, c, d, e\} \subseteq N(D)$. Therefore, if any vertex from the set $\{a, b, c, d, e\}$ is in D, it implies that the whole set is in D.
Case 2: Suppose D contains an arbitrary vertex of V_1. It implies that b, c are in D and therefore $\{a, b, c, d, e\} \subseteq D$. Suppose D contains an arbitrary vertex of E_0. It implies that $\{a, e\} \subseteq D$ and therefore $\{a, b, c, d, e\} \subseteq D$. Suppose D contains v^0 from V_0. This implies that $v^1 \in V_1$ is in $N(D)$. If both b and c are outside D then we have $d_D(v^1) < d_{D^c}(v^1) + 1$, which is a contradiction. This implies that either b or c is in D and therefore $\{a, b, c, d, e\} \subseteq D$.

Now since $\{a, b, c, d, e\} \subseteq D$, a vertex e in E_0 will be either in $N(D)$ or D. If $e \in D$, then we pick an arbitrary neighbour of e in V_0 and put it in D and remove e from D. Therefore, staring with an arbitrary offensive alliance D, we can transform it into another offensive alliance such that $D \cap E_0 = \emptyset$, that is, $E_0 \subseteq N(D)$. As each $e \in E_0$ has to satisfy the condition $d_S(e) \geq d_{S^c}(e) + 1$, we must have a set $S \subseteq V_0$ of size at most k in D such that every vertex in E_0 has at least one neighbour in S. This implies that S is a vertex cover of size at most k in G. $\square$

7.7.2 Strong Offensive Alliance on apex graphs

Recall that a *planar graph* is a graph that can be embedded in the plane, that is, it can be drawn on the plane in such a way that its edges intersect only at their endpoints. In other words, it can be drawn in such a way that no edges cross each other. An *apex graph* is a graph that can be made planar by the removal of a single vertex. The deleted vertex is called an *apex* of the graph.

Theorem 7.7.2. Strong Offensive Alliance *admits a* $\mathcal{O}^*(2^{\mathcal{O}(\sqrt{n}\log n)})$ *algorithm on apex graphs.*

Proof. Note that the treewidth of any apex graph with n vertices is bounded by $\mathcal{O}(\sqrt{n})$. In [70], a polynomial time algorithm is given to solve offensive alliance problem on bounded treewidth graphs with running time $\mathcal{O}^*(2^{\omega}n^{\mathcal{O}(\omega)})$ where ω denotes the treewidth of the input graph. This algorithm can be used to obtain an algorithm with running time $\mathcal{O}^*(2^{\mathcal{O}(\sqrt{n}\log n)})$ for apex graphs. □

While Strong Offensive Alliance can be solved in time $\mathcal{O}^*(2^{\mathcal{O}(\sqrt{n}\log n)})$ for apex graphs on n vertices, we now prove that the existence of algorithms with running time $2^{o(\sqrt{n})}$ is unlikely.

Theorem 7.7.3. Unless ETH fails, the Strong Offensive Alliance problem does not admit a $2^{o(\sqrt{n})}$ algorithm even when restricted to apex graphs.

Proof. We give a linear reduction from Planar Dominating Set to Strong Offensive Alliance, that is, a polynomial-time algorithm that takes an instance of Planar Dominating Set on n vertices and $m = \mathcal{O}(n)$ edges, and outputs an equivalent instance of Strong Offensive Alliance whose size is bounded by $\mathcal{O}(n)$. Let (G, k) be an instance of Planar Dominating Set. Without loss of generality, we assume that G is connected. We construct an equivalent instance (G', k') of Strong Offensive Alliance in the following way. See Figure 7.9 for an illustration.

To construct graph G', we start with graph G. For every edge $e \in E(G)$, we add one parallel edge e' with the same endpoints. We subdivide each edge $e \in E(G)$ and the subdivision vertex for e is denoted by v_e. For each $e \in E(G)$, we introduce a new vertex h_e

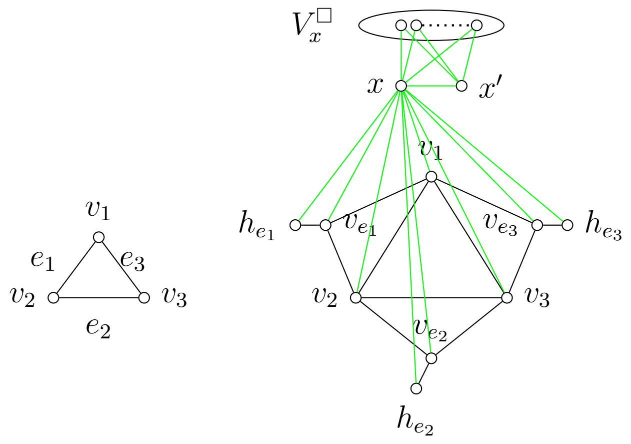


Figure 7.8: The reduction from Planar Dominating Set to Strong Offensive Alliance in Theorem 7.7.3

and make it adjacent to v_e. We introduce two new vertices x and x'. Finally, we add a set $V_x^{\square}$ of $4(m+k+2)$ vertices and make x and x' adjacent to every vertex of $V_x^{\square}$. Lastly, we make x adjacent to every vertex of $V(G) \cup \bigcup\limits_{e\in E(G)} \{v_e, h_e\}$. This completes the construction of G'. Set $k' = m+k+2$ We observe that $G'-x$ is planar. Therefore, G' is an apex graph.

Formally, we claim that G has a dominating set of size at most k if and only if G' has a strong offensive alliance of size at most k'. Suppose G admits a dominating set D of size at most k. We claim that $S = D \cup \{x, x'\} \cup \bigcup\limits_{e\in E(G)} \{v_e\}$ is a strong offensive alliance of size at most k'. It is easy to see that $|S| \leq k'$. We note that $N(S) = \bigcup\limits_{e\in E(G)} \{h_e\} \cup (V(G) \setminus D) \cup V_x^{\square}$. Let v be an arbitrary element of $N(S)$. If v is an element of $\bigcup\limits_{e\in E(G)} \{h_e\}$ or $V_x^{\square}$, then it has two neighbours in S and no neighbours in S^c, so we have $N_S(v) \geq N_{S^c}(v) + 2$. Suppose v is an element of $V(G) \setminus D$. If v has degree d in G then $d_{G'}(v) = 2d+1$. Since D is a dominating set, $v \in V(G) \setminus D$ has at least one neighbour in D. Thus v has at least $d+2$ neighbours in S and at most $d-1$ neighbours outside S. This implies that $N_S(v) \geq N_{S^c}(v) + 2$. Therefore S is a strong offensive alliance.
In the reverse direction, suppose now that S is a strong offensive alliance of size at most k'. We first show that $\{x, x'\} \subseteq S$. As both x and x' have degree more than $2k'$, we claim that x and x' cannot be in $N(S)$. Suppose, for the sake of contradiction, that $x \in N(S)$. Then x must satisfy the condition $N_S(x) \geq N_{S^c}(x) + 2$. This implies that the size of S is greater than k', a contradiction to the assumption that S is a strong offensive alliance of size at

most k'. As S is a non-empty set it contains at least one vertex of G'. Case 1: S contains x; then we are done. Case 2: S contains an element of $V(G') \setminus \{x\}$. As x is adjacent to every other vertex of G', $x \in N(S)$. Since we know x cannot be in $N(S)$, so we must have x in S. Once x is in S, the vertices of $V_x^\square$ are in $N(S)$. The vertices v of $V_x^\square$ satisfy the condition $d_S(v) \geq d_{S^c}(v) + 2$, only if $x' \in S$.

Next we claim that for each $e \in E(G)$ if v_e is not in S then h_e must be in S. Suppose, for the sake of contradiction, that h_e is not in S. Then $h_e \in N(S)$ as h_e is adjacent to x and $x \in S$. Note that h_e has one neighbour x in S and one neighbour v_e in S^c. Therefore we get $N_S(h_e) < N_{S^c}(h_e) + 2$, a contradiction to the assumption that S is a strong offensive alliance. This proves the claim.

Therefore for each $e \in E(G)$, either $v_e \in S$, $h_e \in S$ or both v_e, h_e are in S. Starting from an arbitrary strong offensive alliance S, we can always construct another strong offensive alliance S' such that $\bigcup\limits_{e \in E(G)} \{v_e\} \subseteq S'$ and $|S'| \leq |S|$. We construct S' from S as follows. For each $e \in E(G)$ do the following: if S contains both v_e and h_e, remove h_e; if S contains only h_e, replace it by v_e; if S contains v_e, do not make any changes. Clearly $|S'| \leq |S|$. We claim that S' is also a strong offensive alliance. Note that $N(S') = \Big(N(S) \setminus \bigcup\limits_{e \in E(G)} \{v_e\}\Big) \cup \bigcup\limits_{e \in E(G)} \{h_e\}$. It is easy to see that, for each $e \in E(G)$, we have $N_S(h_e) \geq N_{S^c}(h_e) + 2$. For any other vertex $v \in N(S')$, we have $N_{S'}(v) \geq N_S(v)$. This proves the claim that S' is a strong defensive alliance. Therefore we obtain a strong offensive alliance S' of size at most k' such that $\{x, x'\} \cup \bigcup\limits_{e \in E(G)} v_e \subseteq S'$.

Now we claim that $S' \cap V(G)$ is a dominating set of size at most k in G. Note that $|S' \cap V(G)| \leq k$ as S' contains $\{x, x'\} \cup \bigcup\limits_{e \in E(G)} v_e$ and $|S'| \leq k + m + 2$. Let us consider any vertex $u \in V(G)$. If $d_G(u) = d$ then $d_{G'}(u) = 2d + 1$. If $u \in S' \cap V(G)$ then it is dominated by itself. Therefore, let us assume that $u \in V(G) \setminus S'$. Suppose u has no neighbours in $S' \cap V(G)$. Then it would imply that $d_{S'}(u) = d + 1$ and $d_{S'^c}(u) = d$. In this case, we see that $d_{S'}(u) < d_{S'^c}(u) + 2$ which is a contradiction. This implies that u has a neighbour in $S' \cap V(G)$. That is, every vertex of G is dominated by the set $S' \cap V(G)$. $\square$

7.8 NP-completeness results

In this section, we prove that the Offensive Alliance problem is NP-complete, even when restricted to split, chordal and circle graphs.

7.8.1 Split and Chordal Graphs

A graph G is called *chordal* if it does not contain any chordless cycle of length at least four. Split graphs are a subclass of chordal graphs, where the vertex set can be partitioned into an independent set and a clique. We now prove the following theorem.

Theorem 7.8.1. *The* Offensive Alliance *problem is NP-complete, even when restricted to split or chordal graphs.*

Proof. It is easy to see that the problem is in NP. To show that the problem is NP-hard we give a polynomial reduction from Vertex Cover in graphs of maximum degree 3. Let (G, k) be an instance of Vertex Cover, where G has maximum degree 3. We construct an equivalent instance (G', k') of Offensive Alliance the following way. See Figure 7.9 for an illustration. The vertex set of G' is defined as follows:

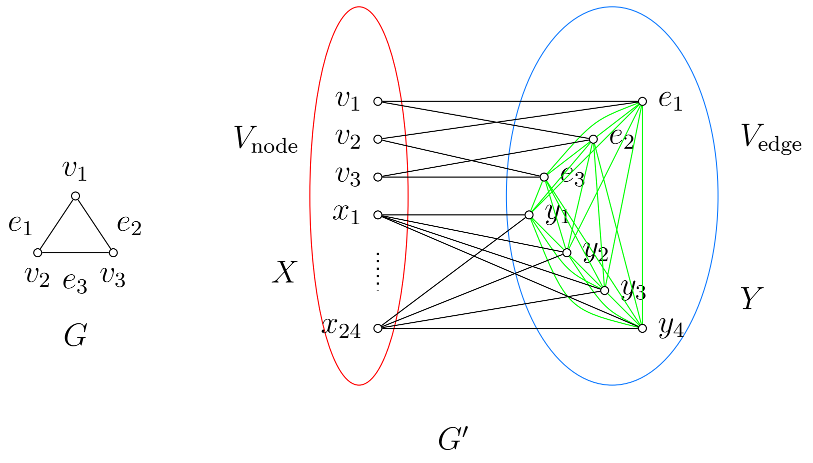


Figure 7.9: An illustration of the reduction from Vertex Cover to Offensive Alliance in Theorem 7.8.1.

1. For every $v \in V(G)$, we introduce v into G'. Set $V_{\text{node}} = V(G)$.

2. For each $e \in E(G)$, we introduce a vertex e into G'. Let $V_{\text{edge}} = \{e_i \mid e_i \in E(G)\}$.

3. We introduce a set $Y = \{y_1, y_2, \ldots, y_{m+1}\}$ of $m + 1$ new vertices into G'.

4. Moreover, introduce a set $X = \{x_1, \ldots, x_{4(n+m)}\}$ of $4(n + m)$ new vertices into G'.

We now create the edge set of G'.

1. For every $v_i \in V_{\text{node}}$ and $e_j \in V_{\text{edge}}$, make v_i adjacent to e_j if and only if v_i is an endpoint of e_j in G.
2. Make the the set $V_{\text{edge}} \cup Y$ a clique in G'.
3. Finally, we make every vertex of X adjacent to every vertex of Y.

Note that $V_{\text{node}} \cup X$ forms an independent set where as the vertices in $V_{\text{edge}} \cup Y$ form a clique. Therefore, G is a split graph. We set $k' = k + m + 1$.

Formally, we claim that G has a vertex cover of size at most k if and only if G' has an offensive alliance of size at most k'. Assume first that G admits a vertex cover S of size at most k. Consider $D = S \cup Y$. Clearly, $|D| \leq k'$. We claim that D is an offensive alliance in G'. Note that $N(D) = V_{\text{edge}} \cup X$. For each $x \in X$, we have $d_D(x) \geq d_{D^c}(x) + 1$ as all its neighbours are inside D. Each $e \in V_{\text{edge}}$ has at least $m + 2$ neighbours in D and at most $m + 1$ neighbours, including itself, outside D. This implies that D is an offensive alliance of size at most k' in G'.

For the reverse direction, let D be an offensive alliance of size at most k' in G'. We first show that $Y \subseteq D$. It is easy to note that $Y \cap N(D) = \emptyset$ as otherwise each $v \in Y \cap N(D)$ has to satisfy the condition $d_D(v) \geq d_{D^c}(v) + 1$ which requires more than k' vertices in D. Since D is a non-empty offensive alliance, it must contain a vertex from the set $V_{\text{node}} \cup V_{\text{edge}} \cup X \cup Y$.

Case 1: Suppose D contains a vertex v from $V_{\text{edge}} \cup X \cup Y$. Then v has at least one neighbour in Y. As $Y \cap N(D) = \emptyset$, therefore we get $Y \subseteq D$.

Case 2: Suppose D contains a vertex v from V_{node}. Let $e \in V_{\text{edge}}$ be a neighbour of v in G'. Then e could be either in D or in $N(D)$. If e is in D, then Case 1 implies that $Y \subseteq D$. Suppose e is in $N(D)$. Then e has to satisfy the condition $d_D(e) \geq d_{D^c}(e) + 1$. Note that the neighbours of e in G' are the vertices of $Y \cup V_{\text{edge}} \setminus \{e\}$ and two endpoints of e in V_{node}, thus $d_{G'}(e) = 2m + 2$. In order to satisfy the condition $d_D(e) \geq d_{D^c}(e) + 1$, vertex e requires at least one vertex from Y to be inside D. Again Case 1 implies that $Y \subseteq D$.

Since the size of D is at most $m+k+1$ and $Y \subseteq D$, it can contain at most k vertices besides the vertices in Y. Given an offensive alliance D, we can construct another offensive alliance D' such that $|D'| \leq |D|$ and $D' \cap (V_{\text{edge}} \cup X) = \emptyset$, in the following way. For each $e \in V_{\text{edge}} \cap D$, we replace e by an arbitrary neighbour of e in V_{node}. If a neighbour of e is already present in D then just remove e and do not add any new vertex. We also remove all the vertices of X from D. The modified D is our D'. Next we argue that D' is an offensive alliance. Since $Y \subseteq D'$ and $D' \cap (V_e \cup X) = \emptyset$, we have $N(D') = V_{\text{edge}} \cup X$. Let v be an arbitrary element of $N(D')$.
Case 1: If $v \in X$ then all its neighbours in G' are in D'. So v trivially satisfies the condition $d_{D'}(v) \geq d_{D'^c}(v) + 1$.
Case 2: Suppose $v \in N(D) \cap V_{\text{edge}}$. We know $V_{\text{edge}} \subseteq D \cup N(D)$. We observe that for the vertices in $N(D) \cap V_{\text{edge}}$, we only increase their number of neighbours in D'. Therefore, every vertex v of $N(D) \cap V_{\text{edge}}$ satisfy the condition $d_{D'}(v) \geq d_{D'^c}(v) + 1$.
Case 3: Suppose $v \in D \cap V_{\text{edge}}$. Every vertex v of $V_{\text{edge}} \cap D$ has at least one neighbour from V_{node} inside D' by the construction of D'. Clearly, each vertex of $V_{\text{edge}} \cap D$ has at least $m+2$ neighbours inside D' and at most $m+1$ (including itself) outside D'.
This shows that D' is an offensive alliance. Note that $V_{\text{edge}} \subseteq N(D')$. Therefore, every $e \in V_{\text{edge}}$ satisfies the condition $d_{D'}(e) \geq d_{D'^c}(e) + 1$, which requires at least one neighbour of e from V_{node} to be inside D'. Therefore $D' \cap V_{\text{node}}$ forms a vertex cover of size at most k in G. This proves that (G, k) is a yes instance. □

7.8.2 Circle graphs

A *circle graph* is an undirected graph whose vertices can be associated with chords of a circle such that two vertices are adjacent if and only if the corresponding chords cross each other. Here, we prove that the OFFENSIVE ALLIANCE problem is NP-complete even when restricted to circle graphs, via a reduction from DOMINATING SET. It is known that the DOMINATING SET problem on circle graphs is NP-hard [88].

Theorem 7.8.2. The OFFENSIVE ALLIANCE problem on circle graphs is NP-complete.

Proof. It is easy to see that the problem is in NP. To show that the problem is NP-hard we give a polynomial reduction from DOMINATING SET on circle graphs. Let (G, k) be an

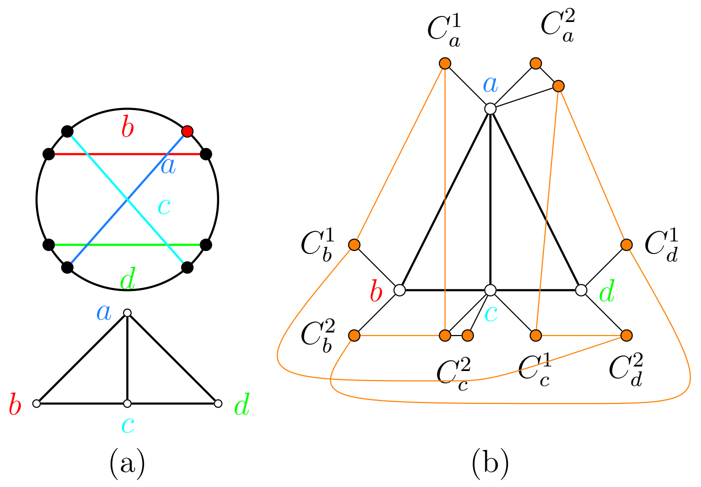


Figure 7.10: (a) Graph G and its circle representation. (b) The graph G' produced by the reduction algorithm. Note that every orange vertex is adjacent to a set of $2r$ vertices, which are not shown here.

instance of DOMINATING SET, where G is a circle graph. Suppose we are also given the circle representation C of G. Without loss of generality, we assume that there are no one degree vertices in G. We construct an instance of (G', r) of OFFENSIVE ALLIANCE as follows (see Figure 7.10). Set $r = 2m + k$, where m is the number of edges in G. For every $v \in V(G)$, we introduce two cliques C_v^1 and C_v^2 where C_v^1 has $\lfloor \frac{d(v)}{2} \rfloor$ nodes and C_v^2 has $\lceil \frac{d(v)}{2} \rceil$ nodes; make v adjacent to every vertex of $C_v^1 \cup C_v^2$; for every $x \in C_v^1 \cup C_v^2$, introduce a set $V_{v,x}^{\square}$ of $2r$ new vertices and make x adjacent to every vertex of $V_{v,x}^{\square}$. The vertices of $V_{v,x}^{\square}$ are not shown in Figure 7.10b. We start at an arbitrary vertex of the circle representation C of G and then traverse the circle in a clockwise direction. We record the sequence in which the chords are visited. For example, in Figure 7.10a, if we start at the red vertex on the circle, then the sequence in which the chords are visited, is a, b, d, c, a, d, b, c. Note that every vertex appears twice in the sequence as every chord is visited twice while traversing the circle. Thus we get a sequence S of length $2n$ where n is the number of chords in C. We use the sequence to connect $2n$ newly added cliques. For every consecutive pair (u, v) in the sequence S, put an edge between a vertex of C_u^1 (resp. C_u^2) and a vertex of C_v^1 (resp. C_v^2) if both u, v appear for the first time (resp. second time) in the sequence; put an edge between a vertex of C_u^1 and a vertex of C_v^2 if u appears for the first time and v appears for the second time in the sequence. These edges are shown in orange in Figure 7.10b. This completes the construction of graph G'. Now we show that G' is indeed a circle graph.

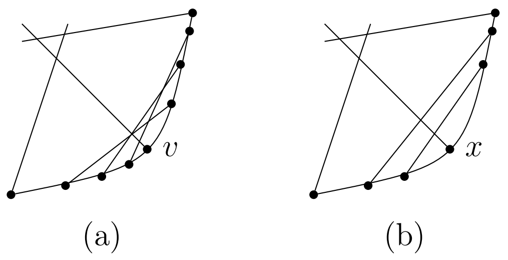


Figure 7.11: (a) The circle representation for the first operation. Let $d(v) = 6$. For v, we introduce C_v^1 in G', and make v adjacent with every vertex of C_v^1. The circle representation of v, C_v^1 and their adjacency are shown here. (b) The circle representation for the second operation with $r = 1$.

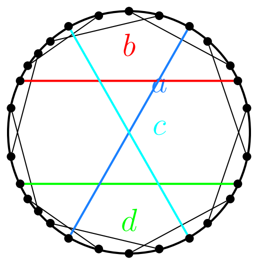


Figure 7.12: A circle representation of the graph G' in Figure 7.10. We do not shown the parallel chords correspond to $2r$ vertices adjacent to every vertex in each clique.

In the reduction algorithm, we have three operations: (i) For every $v \in V(G)$, we introduce two cliques C_v^1 and C_v^2 and make v adjacent to every vertex of C_v^1 and C_v^2. This operation can be incorporated in the circle representation by introducing $\lfloor \frac{d(v)}{2} \rfloor$ intersecting chords at one end of the chord corresponds to v and $\lceil \frac{d(v)}{2} \rceil$ intersecting chords at the other end of the chord corresponds to v. See Figure 7.11a for an illustration. (ii) For every vertex x in clique, we introduce a set of $2r$ new vertices and make x adjacent to each of them. This operation can be easily incorporated in the circle representation by introducing $2r$ parallel chords intersecting the chord corresponds to x. See Figure 7.11b. (iii) For every consecutive pair (u, v) in the sequence S, we put an edge between a vertex of C_u^1 and a vertex of C_v^1. This is incorporated in the circle representation by making the last chord (in clockwise direction) of C_u^1 intersect with the first chord (in clockwise direction) of C_v^1. This is demonstrated in Figure 7.12.

Formally, we claim that G has a dominating set of size at most k if and only if G' has an offensive alliance of size at most r. Assume first that G admits a dominating set S of size at most k. Consider

$$D = \bigcup_{v \in V(G)} V(C_v^1) \cup V(C_v^2) \cup S.$$

Clearly $|D| \leq r$. We claim that D is an offensive alliance in G'. Clearly

$$N(D) = (V(G) \setminus S) \bigcup_{v \in V(G)} \bigcup_{x \in C_v^1 \cup C_v^2} V_{v,x}^{\square}.$$

Each $u \in \bigcup\limits_{v \in V(G)} \bigcup\limits_{x \in C_v^1 \cup C_v^2} V_{v,x}^{\square}$ satisfies $d_D(u) \geq d_{D^c}(u) + 1$. For $u \in V(G) \setminus S$, if $d_G(u) = d$ then in G' we have $d_D(u) \geq d+1$ and $d_{D^c}(u) \leq d-1$. Thus D is an offensive alliance of size at most r in G'.

Conversely, suppose G' admits an offensive alliance D of size at most r. First, we claim that

$$\bigcup_{v \in V(G)} V(C_v^1) \cup V(C_v^2) \subseteq D.$$

It is to be noted that any offensive alliance D of size at most r cannot contain a vertex of degree more than $2r$ in its neighbourhood $N(D)$. As every vertex of $\bigcup\limits_{v \in V(G)} V(C_v^1) \cup V(C_v^2)$ has degree more than $2r$, we have that $N(D) \cap \bigcup\limits_{v \in V(G)} V(C_v^1) \cup V(C_v^2) = \emptyset$. Since D is a non-empty offensive alliance, it must contain a vertex from G'. Suppose D contains a vertex x from $V(C_v^1)$. As x is adjacent to all other vertices in C_v^1, the remaining elements of $V(C_v^1)$ are in $N(D)$. We know $N(D)$ cannot contain any element of $V(C_v^1)$, therefore we have $V(C_v^1) \subseteq D$. Similarly, as $N(D)$ cannot contain any element of $\bigcup\limits_{v \in V(G)} V(C_v^1) \cup V(C_v^2)$, it implies that $\bigcup\limits_{v \in V(G)} V(C_v^1) \cup V(C_v^2) \subseteq D$. This proves the claim.
Note that the total number of vertices in $\bigcup\limits_{v \in V(G)} V(C_v^1) \cup V(C_v^2)$ is $\sum\limits_{v \in V} d(v) = 2m$ and the size of D is at most $2m + k$. Therefore, besides the vertices in $\bigcup\limits_{v \in V(G)} V(C_v^1) \cup V(C_v^2)$, D can include at most k vertices from $V(G)$. The remaining vertices of $V(G)$ are in $N(D)$. Each $v \in V(G) \cap N(D)$ needs exactly one neighbour from $V(G) \cap D$, in addition to its neighbours in $C_v^1 \cup C_v^2$, in order to satisfy the condition $d_D(v) \geq d_{D^c}(v) + 1$. Therefore, $D \cap V(G)$ is a dominating set of size at most k in G. $\square$

7.9 Closing Remarks and Future Directions

In this work we proved that the Offensive Alliance problem is NP-complete even when restricted to bipartite, chordal, split and circle graphs. We proved that the Offensive Alliance problem is W[1]-hard parameterized by a wide range of fairly restrictive structural parameters such as the feedback vertex set number, treewidth, pathwidth, and treedepth of the input graph thus not FPT (unless FPT = W[1]). We thereby resolved an open question stated by Bernhard Bliem and Stefan Woltran (2018) concerning the complexity of Offensive Alliance parameterized by treewidth. This is especially interesting because most "subset problems" that are FPT when parameterized by solution size turned out to be FPT for the parameter treewidth [33], and moreover Offensive Alliance is easy on trees. On the positive side we proved that it can be solved in time $\mathcal{O}^*(\texttt{vc(G)}^{\mathcal{O}(\texttt{vc(G)})})$ where $\texttt{vc(G)}$ is the vertex cover number of the input graph, and the problem admits an FPT algorithm when parameterized by vertex integrity of input graph. We gave lower bound based on ETH for the time needed to solve the Offensive Alliance problem; we proved that it cannot be solved in time $2^{o(n)}$ even when restricted to bipartite graphs, unless ETH fails. We list some natural questions that arise from the results of this study:

- W[t]-membership of Defensive Alliance problem on graphs of bounded treewidth.
- Does Offensive Alliance parameterized by vertex cover number admit a single exponential algorithm or can one show a lower bound with matching time complexity?
- Does Offensive Alliance admit polynomial-time algorithms on some special classes of intersection graph family such as interval graphs, circular arc graphs, unit disk graphs, etc?
- Determine parameterized complexity of Offensive Alliance problem when parameterized by other structural parameters such as twin cover, cluster vertex deletion number and modular-width.

Chapter 8

$\mathcal{F}$-Free Edge Deletion

8.1 Introduction

Given a graph $G = (V, E)$ and a set $\mathcal{F}$ of forbidden subgraphs, we study the $\mathcal{F}$-Free Edge Deletion problem, where the goal is to remove a minimum number of edges such that the resulting graph does not contain any $F \in \mathcal{F}$ as a (not necessarily induced) subgraph. Enright and Meeks (Algorithmica, 2018) gave an algorithm to solve $\mathcal{F}$-Free Edge Deletion whose running time on an n-vertex graph G of treewidth $\mathsf{tw}(G)$ is bounded by $2^{O(|\mathcal{F}|\mathsf{tw}(G)^r)}n$, if every graph in $\mathcal{F}$ has at most r vertices. We complement this result by showing that $\mathcal{F}$-Free Edge Deletion is W[1]-hard when parameterized by $\mathsf{tw}(G) + |\mathcal{F}|$. We also show that $\mathcal{F}$-Free Edge Deletion is W[2]-hard when parameterized by the combined parameters solution size, the feedback vertex set number and pathwidth of the input graph. This chapter is based on the paper [62].

8.2 Hardness of $\mathcal{F}$-Free Edge Deletion parameterized by $\mathsf{tw}(G) + |\mathcal{F}|$

Enright and Meeks [44] gave an algorithm to solve $\mathcal{F}$-Free Edge Deletion whose running time on an n-vertex graph G of treewidth $\mathsf{tw}(G)$ is bounded by $2^{O(|\mathcal{F}|\mathsf{tw}(G)^r)}n$, if every graph in $\mathcal{F}$ has at most r vertices. In this section, we complement this result by showing that $\mathcal{F}$-Free Edge Deletion is W[1]-hard when parameterized by $\mathsf{tw}(G) + |\mathcal{F}|$. To show W[1]-hardness of $\mathcal{F}$-Free Edge Deletion, we reduce from the following problem, which is known to be W[1]-hard parameterized by the treewidth of the graph [123]:

Minimum Maximum Outdegree
Input: An undirected graph $G = (V, E)$, an edge weighting $w : E(G) \to Z^+$ given in unary and a positive integer r.
Question: Is there an orientation of the edges of G such that, for each $v \in V(G)$, the sum of the weights of outgoing edges from v is at most r?

The *weighted degree* $d_w(u; G)$ of a vertex $u \in V$ is defined as $\sum_{v \in N_G(u)} w(u, v)$. The *weighted maximum degree* $\Delta_w(G)$ of G is defined as $\max_{u \in V} d_w(u; G)$. In this section, we prove the following theorem:

Theorem 8.2.1. *The $\mathcal{F}$-Free Edge Deletion problem is W[1]-hard when parameterized by $\mathsf{tw}(G) + |\mathcal{F}|$.*

Proof. Let $I = (G = (V, E, w), r)$ be an instance of the Minimum Maximum Outdegree problem.
Construction: We construct an instance $I' = (G', k, \mathcal{F})$ of $\mathcal{F}$-Free Edge Deletion the following way (see Figure 8.1).

1. For each edge $(u, v) \in E(G)$, we introduce the following sets of new vertices $V_{uv} = \{u_1^v, \ldots, u_{w(u,v)}^v\}$, $V'_{uv} = \{u_1'^v, \ldots, u_{w(u,v)}'^v\}$, $V_{vu} = \{v_1^u, \ldots, v_{w(u,v)}^u\}$ and $V'_{vu} = \{v_1'^u, \ldots, v_{w(u,v)}'^u\}$. We make u (resp. v) adjacent to all the vertices in $V_{uv} \cup V'_{uv}$ (resp. $V_{vu} \cup V'_{vu}$). Let $E_{u,u^v} = \Big\{(u, x) \mid x \in V_{uv}\Big\}$, $E'_{u,u'^v} = \Big\{(u, x) \mid x \in V'_{uv}\Big\}$, $E_{v,v^u} = \Big\{(v, x) \mid x \in V_{vu}\Big\}$ and $E'_{v,v'^u} = \Big\{(v, x) \mid x \in V'_{vu}\Big\}$.
2. For every vertex $u \in V(G)$, we also add a set $V_u^{\square}$ of $\Delta_w(G) - d_w(u; G)$ many vertices and make them adjacent to u.
3. Arrange the vertices of G in a linear order. We define two sets of pairs of vertices:

$$C_1 = \Big\{\{(u_i'^v, v_i'^u)\} \mid (u, v) \in E(G), 1 \le i \le w(u, v)\Big\}$$
$$\bigcup \Big\{(u_i'^v, v_{i+1}'^u), (u_{w(u,v)}'^v, v_1'^u) \mid (u, v) \in E(G), 1 \le i \le w(u, v) - 1\Big\},$$

$$C_2 = \Big\{ \{(u_i^v, v_i^u)\} \mid (u,v) \in E(G), 1 \le i \le w(u,v) \Big\}$$
$$\bigcup \Big\{ (u_i^v, v_{i+1}^u), (u_{w(u,v)}^v, v_1^u) \mid (u,v) \in E(G), 1 \le i \le w(u,v) - 1 \Big\}.$$

In the definition of C_1 and C_2, for each $(u,v) \in E(G)$, consider u appears before v in the linear ordering.

4. Let $\omega = \sum_{e \in E(G)} w(e)$ and $N = n + 3\omega + 1$. For every pair of vertices $(u'^v, v'^u) \in C_1$, we add a "blue" path $P_{u'^v, v'^u}$ of length $4N - 2$ joining u'^v and v'^u, whose internal vertices are new. Similarly, for every pair of vertices $(u^v, v^u) \in C_2$, we add a "red" path P_{u^v, v^u} of length N joining u^v and v^u, whose internal vertices are new.

5. We set $k = \omega$ and $\mathcal{F} = \{S_{\Delta_w(G)+r+1}, C_{5N+2}\}$ where C_{5N+2} is the cycle graph of order $5N + 2$ and $S_{\Delta_w(G)+r+1}$ is the star graph of order $\Delta_w(G) + r + 1$.

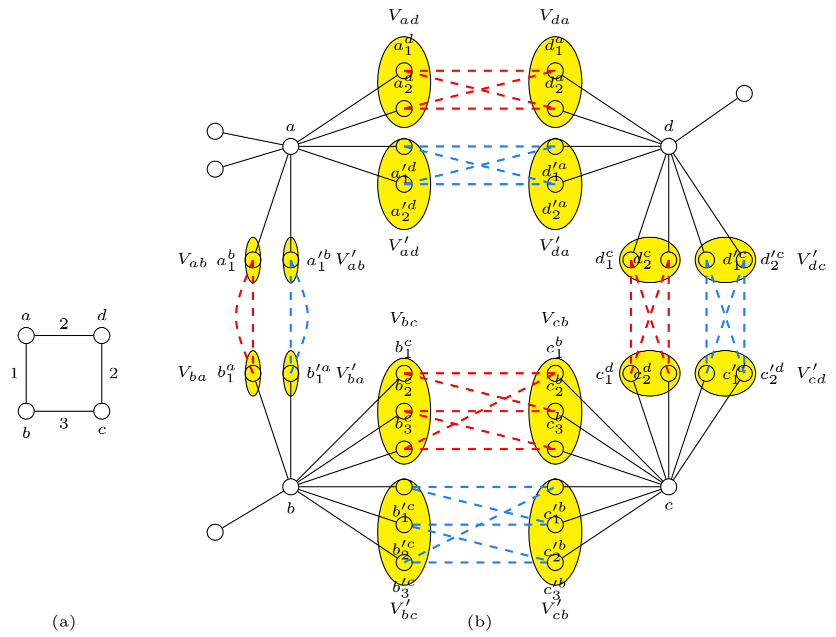


Figure 8.1: The reduction from MINIMUM MAXIMUM OUTDEGREE to $\mathcal{F}$-FREE EDGE DELETION in Theorem 8.2.1. (a) An undirected graph G with edge weights and $r = 3$. (b) The graph G' produced by the reduction algorithm. The red dashed lines represent red paths of length N, and the blue dashed lines represent blue paths of length $4N - 2$.

Clearly I' can be computed in polynomial time. In the I' instance, we have $|\mathcal{F}| = 2$. We now show that the treewidth of G' depends only on the treewidth of G. We do so by modifying an optimal tree decomposition τ of G as follows:

- For every edge (u, v) of G, there is a node t in τ whose bag B contains u and v. We observe that the gadget replacing the edge $(u, v) \in E(G)$ is the union of two vertex-disjoint cycles after removing the vertices u and v. As any cycle has treewidth two, we can construct a tree decomposition τ_{uv} of this gadget of width four. Now, we make the node t adjacent to an arbitrary node of τ_{uv}.
- For each $u \in V(G)$, we take an arbitrary node in τ whose bag B contains u; add to this node a chain of nodes $N_1, N_2, \ldots, N_{|V_u^\square|}$ such that the bag of N_i is $B \cup \{x_i\}$ where $x_i \in V_u^\square$.

This implies that the treewidth of G' is at most treewidth of G plus four.

Correctness: Now we show that our reduction is correct. That is, we prove that $I = (G = (V, E, w), r)$ is a yes instance of MINIMUM MAXIMUM OUTDEGREE if and only if $I' = (G', k, \mathcal{F})$ is a yes instance of $\mathcal{F}$-FREE EDGE DELETION. Let D be the directed graph obtained by an orientation of the edges of G such that for each vertex the sum of the weights of outgoing edges is at most r. We claim that the set of edges

$$E' = \bigcup_{(u,v)\in E(D)} E_{v,v^u} = \bigcup_{(u,v)\in E(D)} \big\{(v, x) \mid x \in V_{vu}\big\}$$

is a solution of I'. Note that (u, v) or (v, u) is a directed edge of D. Clearly, we have $|E'| = \omega$. We need to show that $\widetilde{G'} = G' \setminus E'$ does not contain any forbidden graph $F \in \{S_{\Delta_w(G)+r+1}, C_{5N+2}\}$ as a subgraph. We prove the following two claims:

Claim 8.2.1. *$\widetilde{G'}$ does not contain $S_{\Delta_w(G)+r+1}$ as a subgraph.*

Proof. We show that every vertex in $\widetilde{G'}$ has degree at most $\Delta_w(G) + r$. It is clear from the construction that if $v \in V(G')\setminus V(G)$ then $d_{\widetilde{G'}}(v) \leq 3$. Let w^v_{out} and w^v_{in} denote the sum of the weights of outgoing and incoming edges of vertex v in D, respectively. Note that $d_w(v; G) = w^v_{\text{out}} + w^v_{\text{in}}$. If $v \in V(G)$, then $N_{G'}(v) = V_v^\square \bigcup_{u\in N_G(v)} V_{vu} \cup V'_{vu}$. Hence

$d_{G'}(v) = \Delta_w(G) - d_w(v;G) + 2d_w(v;G)$ as $|V_v^{\square}| = \Delta_w(G) - d_w(v;G)$ and $|\bigcup\limits_{u \in N_G(v)} V_{vu} \cup V'_{vu}| = \sum\limits_{v \in N_G(u)} 2w(u,v) = 2d_w(v;G)$. If the direction of the edge between u and v is from u to v in D, then E' includes edges between v and V_{vu}; in other words, in $\widetilde{G'}$ all edges between v and V_{vu} are deleted. Therefore,

$$N_{\widetilde{G'}}(v) = V_v^{\square} \cup \bigcup_{u \in N_G(v)} (V_{vu} \cup V'_{vu}) \setminus \bigcup_{(u,v) \in E(D)} V_{vu}.$$

Thus the degree of v in $\widetilde{G'}$ is given by

$$\begin{aligned} d_{\widetilde{G'}}(v) &= \Delta_w(G) - d_w(v;G) + 2d_w(v;G) - w_{\text{in}}^v \\ &= \Delta_w(G) + d_w(v;G) - w_{\text{in}}^v \\ &= \Delta_w(G) + (w_{\text{in}}^v + w_{\text{out}}^v) - w_{\text{in}}^v \\ &= \Delta_w(G) + w_{\text{out}}^v \end{aligned}$$

This implies that $d_{\widetilde{G'}}(x) \le \Delta_w(G) + r$ as $w_{\text{out}}^x \le r$. Therefore, $\widetilde{G'}$ does not contain $S_{\Delta_w(G)+r+1}$ as a subgraph.

Claim 8.2.2. *$\widetilde{G'}$ does not contain C_{5N+2} as a subgraph.*

Proof. Targeting a contradiction, let us assume that $\widetilde{G'}$ contains C_{5N+2} as a subgraph. We make two cases based on whether the cycle contains some original vertex u from $V(G)$ or not.

Case 1: Suppose the cycle includes at least one original vertex $u \in V(G)$. Let u be adjacent to v in G. Without loss of generality, we assume that the direction of the edge between u and v is from u to v in D. Then the edges in $E_{v,v^u} = \Big\{(v,x) \mid x \in V_{vu}\Big\}$ are not present in $\widetilde{G'}$. Further, we make two subcases based on whether the cycle contains a "blue" edge or a "red" edge. It is easy to note that $\widetilde{G'}$ does not have a cycle that contains both red and blue edges.

Subcase 1.1: Suppose the cycle starts at $u \in V(G)$ and includes at least one blue edge from a blue path $P_{u'^v,v'^u}$. Then the cycle includes all blue edges of the path $P_{u'^v,v'^u}$ of length $4N-2$ and reaches the vertex v'^u. There are many ways to return to u from v'^u, but every path from v'^u to u in $\widetilde{G'}$ includes another blue path, which makes the length of the cycle at least

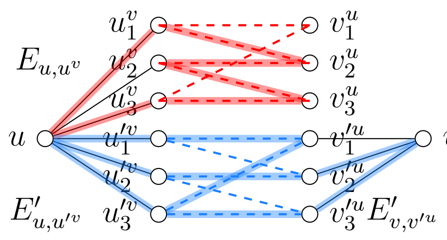


Figure 8.2: The gadget corresponds to an edge (u, v) with weight 3 in $\widetilde{G'}$. The direction of the edge between u and v is from u to v in D. Note that the edges of $E_{v,v^u} = \big\{(v, x) \mid x \in V_{vu}\big\}$ are included in E' and hence not present in $\widetilde{G'}$.

$8N - 2 > 5N + 2$. This implies that a cycle of length $5N + 2$ does not exist in this case. Figure 8.2 shows a “blue" cycle of length $8N - 2$ that contains one original vertex u and a “blue" cycle of length $8N$ that contains two original vertices u and v. Clearly, there are cycles of length larger than $8N$ that contains more than two original vertices.

Subcase 1.2: Suppose the cycle starts at $u \in V(G)$ and includes at least one red edge. In this case, the cycle begins with an edge $e \in E_{u,u^v}$. Next, it must continue with a red edge. Observe that if the cycle includes one red edge from P_{u^v,v^u} then it must include all the red edges of the path P_{u^v,v^u} of length N and reaches v^u. As edges in E_{v,v^u} are not present in $\widetilde{G'}$, it must take another red path of length N starting at v^u, and reaches a vertex in V_{u,u^v}, and finally return to u. This way we can get cycles of length $2Ni+2$ for $i = 1, 2, \ldots, w(u, v)-1$. Figure 8.2 shows a cycle of length $4N+2$. Note that $2Ni+2 \neq 5N+1$ for $i = 1, 2, \ldots, w(u, v)-1$. Thus we showed that a “red" cycle of length $5N + 2$ does not exist.
Case 2: Suppose the cycle does not include any original vertex. Then it also does not include any edge from $\bigcup_{(u,v)\in E(G)} E_{u,u^v} \cup E_{v,v^u} \cup E'_{u,u'^v} \cup E'_{v,v'^u}$. Further, we make two subcases based on whether the cycle contains a blue edge or not.
Subcase 2.1: Suppose the cycle contains a blue edge. Since the original vertices of $V(G)$ are not allowed, the cycle must include $2w(u, v)$ many blue paths of length $4N - 2$ each. In this case, for each $(u, v) \in E(G)$, one can get a “blue" cycle of length $2w(u, v)(4N-2) \neq 5N+2$, where $w(u, v)$ is the weight of (u, v) in G. Figure 8.3 shows a “blue" cycle of length $6(4N-2)$. Therefore, a “blue" cycle of length $5N + 2$ does not exist.

Subcase 2.2: Suppose the cycle does not contain any blue edges. Since the blue edges and

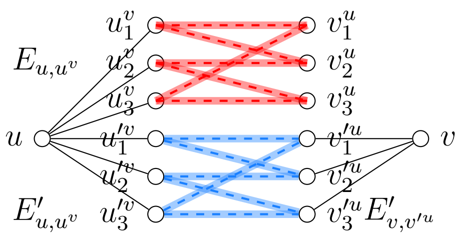


Figure 8.3: The gadget corresponds to an edge (u, v) with weight 3 in $\widetilde{G'}$. Suppose the direction of the edge between u and v is from u to v in D.

the original vertices in $V(G)$ are not allowed, the cycle must contain only red edges. In this case one can get a "red" cycle of length $2w(u,v)N \neq 5N+2$. Therefore, a "red" cycle of length $5N+2$ does not exist. $\square$

For the reverse direction, let $E' \subseteq E(G')$ be a solution for I', that is, $|E'| = \omega$ and $G' \setminus E'$ dose not contain any $F \in \mathcal{F} = \{S_{\Delta_w(G)+r+1}, C_{5N+2}\}$ as a subgraph. We now prove a crucial property of E'.

Claim 8.2.3. *For each $(u,v) \in E(G)$, the set E' contains either $E_{u,u^v}, E_{v,v^u}, E'_{u,u'^v}$ or E'_{v,v'^u}.*

Proof. For each edge $(u,v) \in E(G)$, there are u-v paths of length $4N$ through the blue edges. We call such paths the paths of type `blue`. Similarly, for each edge $(u,v) \in E(G)$, there are u-v paths of length $N+2$ through the red edges. We call such paths the paths of type `red`. We observe that a path of type `red` and a path of type `blue` together form a cycle of length $5N+2$. See Figure 8.4(a). Therefore, to avoid such a cycle, the solution must destroy either all u-v paths of type `red` or all u-v paths of type `blue` for each $(u,v) \in E(G)$. Since the number of edge-disjoint u-v paths of type `red` (resp. `blue`) is $w(u,v)$, the minimum number of edges whose deletion destroys all u-v paths of type `red` (resp. `blue`) is $w(u,v)$. We must add at least $w(u,v)$ many edges to E' for each $(u,v) \in E(G)$. As $|E'| = \omega$, it implies that E' includes exactly $w(u,v)$ many edges for each $(u,v) \in E(G)$.

Case 1: Suppose that the deletion of edges in E' destroys all u-v paths of type `red` and it uses exactly $w(u,v)$ edges to destroy all u-v paths of type `red`. We make two important observations about E':

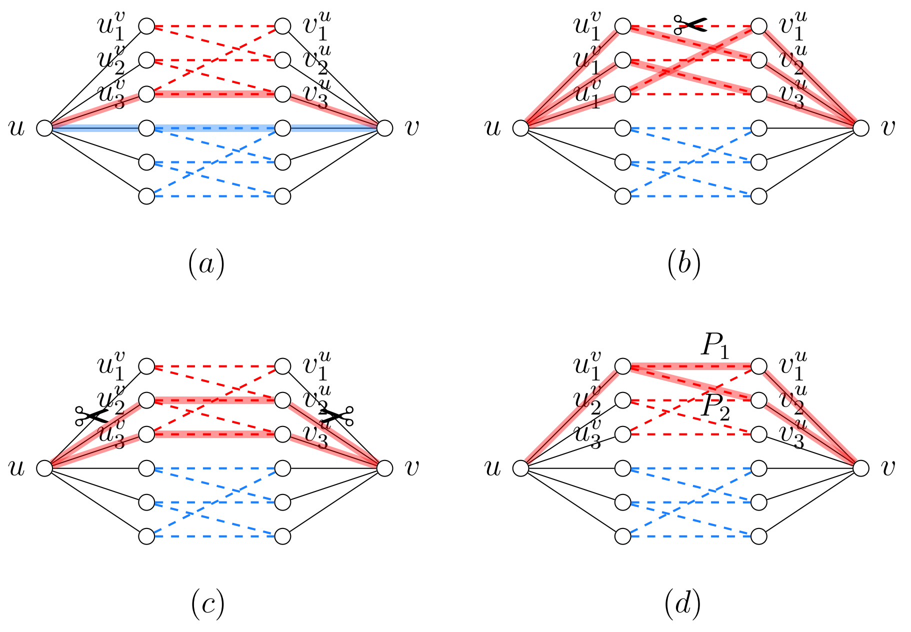


Figure 8.4: The gadget corresponds to an edge (u, v) with weight 3.

1. The solution E' does not include any red edge. The reason is this. If we delete a red edge then, there will be still at least $w(u, v)$ many edge disjoint u-v paths of type `red` left. For example, deletion of a red edge of the path $P_{u_1^v, v_1^u}$ destroys the path $P_{u_1^v, v_1^u}$, but there are still 3 edge disjoint u-v paths of type `red` as shown in Figure 8.4(b). Thus, deletion of three edges (including a red edge) is not enough to destroy all u-v paths of type `red`. In general, deletion of $w(u, v)$ edges (including a red edge) is not enough to destroy all u-v paths of type `red`. Therefore, E' does not include red edges. It implies that a solution must contain edges from $E_{u,u^v} \cup E_{v,v^u}$.

2. The solution E' includes either (u, u_i^v) or (v, v_i^u) for all $1 \leq i \leq w(u, v)$. If we remove both (u, u_i^v) and (v, v_i^u) for some i, we will still have $w(u, v) - 1$ many edge disjoint u-v paths of type `red`. For example, after removing two edges (u, u_1^v) and (v_1^u, v), we still have two edge disjoint u-v paths of type `red` as shown in Figure 8.4(c). Clearly, deletion of 3 edges (including (u, u_1^v) and (v_1^u, v)) are not enough to destroy all u-v paths of type `red`. In general, deletion of $w(u, v)$ edges, including two edges (u, u_i^v) and (v, v_i^u), are not enough to destroy all u-v paths of type `red`.

Without loss of generality, we assume that (u, u_1^v) is not part of the solution, that is, we are not deleting (u, u_1^v) from the graph G'. There are two paths $P_1 = (u, u_1^v, P_{u_1^v, v_1^u}, v_1^u, v)$ and $P_2 = (u, u_1^v, P_{u_1^v, v_2^u}, v_2^u, v)$ that go through edge (u, u_1^v). Figure 8.4(d) shows P_1 and P_2. Since we do not delete (u, u_1^v) from the graph G' and red edges cannot be deleted, we are forced to delete both (v, v_1^u) and (v, v_2^u), in order to destroy P_1 and P_2. That is, edges (v, v_1^u) and (v, v_2^u) are in E'. As (v, v_2^u) is in E', by Observation 2 (u, u_2^v) is not in E'. Again, it forces (v, v_2^u) and (v, v_3^u) to be part of the solution. Applying this argument repeatedly, we see that E' contains E_{v,v^u}. As $|E_{v,v^u}| = w(u, v)$, no edges from E_{u,u^v} can be part of the solution. This shows that E' contains either E_{u,u^v} or E_{v,v^u}.
Case 2: Suppose that the deletion of edges in E' destroys all u-v paths of type `blue`. Using the same arguments, we can prove that E' contains either E'_{u,u'^v} or E'_{v,v'^u}. This concludes the proof of the claim. $\square$

We now define a directed graph D by $V(D) = V(G)$ and

$$E(D) = \Big\{(u, v) \mid E_{v,v^u} \text{ or } E'_{v,v'^u} \subseteq E'\Big\} \bigcup \Big\{(v, u) \mid E_{u,u^v} \text{ or } E'_{u,u'^v} \subseteq E'\Big\}.$$

Suppose there is a vertex x in D for which $w_{\text{out}}^x > r$. In this case, we observe that x is adjacent to more than $\Delta_w(G) + r$ vertices in graph $\widetilde{G'} = G' \setminus E'$. This is a contradiction as vertex x and its neighbours form the star graph $S_{\Delta_w(G)+r+1}$, which is a forbidden graph in I'. $\square$

8.3 Hardness of $\mathcal{F}$-Free Edge Deletion parameterized by $k + \mathsf{fvs}(G) + \mathsf{pw}(G)$

In this section we show that $\mathcal{F}$-Free Edge Deletion is W[2]-hard parameterized by $k + \mathsf{fvs}(G) + \mathsf{pw}(G)$, via a reduction from Hitting Set. In the Hitting Set problem, we are given a universe $U = \{1, 2, \ldots, n\}$, a family $\mathcal{A}$ of sets over U, and a positive integer k. The objective is to decide whether there is a subset $H \subseteq U$ of size at most k such that H contains at least one element from each set in $\mathcal{A}$. It is known that the Hitting Set problem is W[2]-hard when parameterized by the solution size k [31]. We prove the following theorem:

Theorem 8.3.1. The $\mathcal{F}$-FREE EDGE DELETION problem is W[2]-hard when parameterized by $k + \mathsf{fvs}(G) + \mathsf{pw}(G)$.

Proof. Let $(U, \mathcal{A}, k)$ be an instance I of the HITTING SET problem and let $U = \{1, 2, \dots, n\}$. We construct an instance $I' = (G, \mathcal{F}, k')$ of $\mathcal{F}$-FREE EDGE DELETION as follows. We first introduce a *central* vertex v. For every $i \in U$, we attach to this vertex a cycle C_i of length $2i + 2$. See Figure 8.5. Note that $C_1, C_2, \dots, C_n$ have only one vertex v in common. We

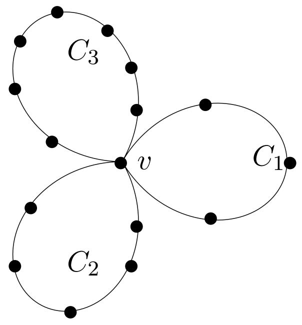


Figure 8.5: The graph G of the $\mathcal{F}$-FREE EDGE DELETION problem instance constructed in the reduction of Theorem 8.3.1 for $n = 3$.

define G as follows

$$V(G) = \bigcup_{i \in U} V(C_i) \text{ and } E(G) = \bigcup_{i \in U} E(C_i).$$

We observe that the graph G contains a unique cycle C_i of length $2i + 2$ for each $i \in U$. Clearly, $\{v\}$ is a feedback vertex set of G. The feedback vertex set number of G is 1. The pathwidth of G is 2. As $G - v$ is disjoint union of paths, $G - v$ has a path decomposition $\mathcal{P}$ of width 1. We add v in each bag of $\mathcal{P}$ to get a path decomposition of G of width 2. Thus its pathwidth is less than or equal to 2. On the other hand, G contains cycle, thus pathwith of G is greater than or equal to 2. Therefore the pathwidth of G is 2. Now, we define a family $\mathcal{F}$ of forbidden subgraphs. For every set $A \in \mathcal{A}$, we add a graph F_A in $\mathcal{F}$, where F_A is defined as follows:

$$V(F_A) = \bigcup_{i \in A} V(C_i) \text{ and } E(F_A) = \bigcup_{i \in A} E(C_i).$$

We take $k' = k$. Next, we show that I is a yes instance if and only if I' is a yes instance. Let H be a solution for the instance I. We see that by deleting one arbitrary edge from every cycle C_i, $i \in H$, we can avoid all the forbidden graphs in $\mathcal{F}$. Therefore, we have a solution $E' \subseteq E(G)$ for the instance I' such that $|E'| \leq k$.

Conversely, suppose $E' \subseteq E(G)$ with $|E'| \leq k$ is a solution for the instance I'. We see that $H = \{i \mid E(C_i) \cap E' \neq \emptyset\}$ is a hitting set for the instance I. Furthermore, $|H| \leq k$ as $|E'| \leq k$. □

Observe that the constructed instance in the previous theorem is an outerplanar bipartite graph. Therefore we get the following result:

Corollary 8.3.2. $\mathcal{F}$-Free Edge Deletion is W[2]-hard when parameterized by $k + \mathsf{fvs}(G) + \mathsf{pw}(G)$, even when restricted to outerplanar bipartite graphs.

8.4 Closing Remarks

The main contributions in this chapter are that $\mathcal{F}$-Free Edge Deletion is W[1]-hard when parameterized by $|\mathcal{F}| + \mathsf{tw}(G)$ and $\mathcal{F}$-Free Edge Deletion is W[2]-hard when parameterized by $k + \mathsf{fvs}(G) + \mathsf{pw}(G)$.

Chapter 9

The $\mathcal{T}_{h+1}$-free edge deletion problem

9.1 Introduction

Given an undirected graph $G = (V, E)$ and two integers k and h, we study $\mathcal{T}_{h+1}$-FREE EDGE DELETION, where the goal is to remove at most k edges such that the resulting graph does not contain any tree on $h + 1$ vertices as a (not necessarily induced) subgraph, that is, we delete at most k edges in order to obtain a graph in which every component contains at most h vertices. This is desirable from the point of view of restricting the spread of a disease in transmission networks. Enright and Meeks (Algorithmica, 2018) gave an algorithm to solve $\mathcal{T}_{h+1}$-FREE EDGE DELETION whose running time on an n-vertex graph G of treewidth $\mathsf{tw}(G)$ is bounded by $O((\mathsf{tw}(G)h)^{2\mathsf{tw}(G)}n)$. However, it remains open whether the problem might belong to FPT when parameterized only by the treewidth $\mathsf{tw}(G)$; they conjectured that treewidth alone is not enough, and that the problem is W[1]-hard with respect to this parameterization. We resolve this conjecture by showing that $\mathcal{T}_{h+1}$-FREE EDGE DELETION is indeed W[1]-hard when parameterized by $\mathsf{tw}(G)$ alone. We resolve two additional open questions posed by Enright and Meeks (Algorithmica, 2018) concerning the complexity of $\mathcal{T}_{h+1}$-FREE EDGE DELETION on planar graphs and $\mathcal{T}_{h+1}$-FREE ARC DELETION. We prove that the $\mathcal{T}_{h+1}$-FREE EDGE DELETION problem is NP-complete even when restricted to planar graphs. We also show that the $\mathcal{T}_{h+1}$-FREE ARC DELETION problem is W[2]-hard when parameterized by the solution size on directed acyclic graphs. This chapter is based on the papers [66, 62].

9.2 $\mathcal{T}_{h+1}$-Free Edge Deletion parameterized by vertex cover number

In this section, we present an FPT algorithm for the $\mathcal{T}_{h+1}$-Free Edge Deletion problem parameterized by the vertex cover number. We prove the following theorem:

Theorem 9.2.1. *$\mathcal{T}_{h+1}$-Free Edge Deletion is FPT when parameterized by the vertex cover number of the input graph.*

To prove this theorem we formulate the problem as an Integer Linear Programming problem and use an algorithm of Fellows et al. [47] that solves parameterized minimization ILPs in FPT time when parameterized by the number of variables. Without loss of generality we assume that the graph has no isolated vertices. Let S be a vertex cover of $G = (V, E)$ of size k. We denote by I the independent set $V \setminus S$. We partition the independent set I into at most 2^k twin classes $I_1, I_2, \ldots, I_{2^k}$, where some of them can also be empty. Two vertices u and v are in the same twin class if $N(u) = N(v)$. Our goal is to minimize the size of $E' \subseteq E(G)$ such that after deleting E' from G, each connected component of the resulting graph has at most h vertices. First, we guess the intersection of S with the connected components in $\widetilde{G} = G \setminus E'$. It is clear that the number of guesses is equal to the number of different partitions of the k-element set S, which is equal to the Bell number B_k. It is known that $B_k = \frac{1}{e}\sum\limits_{i=0}^{\infty} \frac{i^k}{i!}$. See [104]. For every guess, we will reduce our problem to an integer linear programming (ILP) where the number of variables is a function of the vertex cover number k. Since integer linear programming is fixed-parameter tractable when parameterized by the number of variables, we will conclude that our problem is fixed-parameter tractable when parameterized by the vertex cover number. Let us consider a particular partition $P = \{S_1, S_2, \ldots, S_\ell\}$ of S where $\ell \leq k$. For a given partition P of S, we call an edge a *cross edge* if both endpoints of that edge are in S but one endpoint in S_i and the other endpoint in S_j such that $i \neq j$. We denote the number of cross edges of partition P by $\texttt{cr}(P)$.

ILP Formulation: Given a partition $P = \{S_1, S_2, \ldots, S_\ell\}$ of S, let C_i be the component of $\widetilde{G}$ such that $S \cap C_i = S_i$ for $1 \leq i \leq \ell$. Let $C_{\ell+1}$ be the collection of size one components in $\widetilde{G}$ such that $S \cap C_{\ell+1} = \emptyset$. For each I_i and C_j, we associate a variable x_{ij} that indicates $|I_i \cap C_j| = x_{ij}$, that is, x_{ij} denotes the number of vertices in twin class I_i that go to C_j.

Because the vertices in I_i have the same neighbourhood, the variables x_{ij} determine the components uniquely and hence determine the required set of edges $E' \subseteq E(G)$. We add the following constraints to ILP. The vertices of each twin class I_i is distributed among the components $C_1, C_2, \ldots, C_\ell$ and $C_{\ell+1}$. Thus we have the following constraints:

$$\sum_{j=1}^{\ell+1} x_{ij} = |I_i| \text{ for all } 1 \leq i \leq 2^k \tag{9.1}$$

We want each connected component C_j in the resulting graph $\widetilde{G}$ to have at most h vertices. Thus we have the following constraint:

$$\sum_{i=1}^{2^k} x_{ij} + |S_i| \leq h \text{ for all } 1 \leq j \leq \ell \tag{9.2}$$

Note that every vertex in I_i has the same set of neighbours in S. Thus if a vertex $v \in I_i$ goes to C_j then we have to remove all edges between v and $S \setminus S_j$, so that $C_1, C_2, \ldots, C_\ell$ and $C_{\ell+1}$ remains distinct components. Therefore, if x_{ij} vertices of I_i go to C_j, then we need to remove in total $|N_{S \setminus S_j}(v)| \times x_{ij}$ edges, where v is a vertex in I_i. Hence we want to minimize the following objective function:

$$\texttt{cr}(P) + \sum_{i=1}^{2^k} \sum_{j=1}^{\ell+1} |N_{S \setminus S_j}(v_i)| \times x_{ij} \tag{9.3}$$

where $S_{\ell+1} = \emptyset$, $\texttt{cr}(P)$ is the number of cross edges of partition P and v_i is a vertex in the twin class I_i. In the following, we present an ILP formulation of the $\mathcal{T}_{h+1}$-Free Edge Deletion problem, where a partition $P = \{S_1, S_2, \ldots, S_\ell\}$ of S is given:

$$\text{Minimize } \mathtt{cr}(P) + \sum_{i=1}^{2^k} \sum_{j=1}^{\ell+1} |N_{S\setminus S_j}(v_i)| \times x_{ij}$$

Subject to

$$\sum_{j=1}^{\ell+1} x_{ij} = |I_i| \quad \text{for all} \quad 1 \leq i \leq 2^k$$

$$\sum_{i=1}^{2^k} x_{ij} + |S_i| \leq h \quad \text{for all} \quad 1 \leq j \leq \ell$$

In the formulation for $\mathcal{T}_{h+1}$-Free Edge Deletion, we have at most $2^k(k+1)$ variables. The value of objective function is bounded by n^2 and the value of any variable in the integer linear programming is bounded by n. The constraints can be represented using at most $O(2^k \log n)$ bits. Lemma 2.3.2 implies that we can solve the problem with the guess P in FPT time. There are at most B_k choices for P, and the ILP formula for a guess can be solved in FPT time. Thus Theorem 9.2.1 holds.
Now we make two important remarks:

Remark 9.2.1. We argue that the vertex cover number and the solution size are incomparable. Let G be the disjoint union of $\frac{n}{h}$ cliques of size $h \geq 2$ each, and an arbitrary vertex a of the first clique is adjacent to an arbitrary vertex b of the second clique. Clearly E' has to include only (a,b) so that each connected component in $G \setminus E'$ has at most h vertex. Thus the solution size is equal to 1, while the vertex cover number is $\frac{n}{h}(h-1) = n - \frac{n}{h}$. On the other hand consider the graph $G = K_{1,n-1}$. Clearly, the solution size is equal to $n-h$ while the vertex cover number is 1. As the vertex cover number and the solution size are incomparable, the question of whether the $\mathcal{T}_{h+1}$-Free Edge Deletion problem is FPT parameterized by the solution size remains open.

Remark 9.2.2. Usually, algorithms for the vertex cover number can be adapted for twin cover. Here we observe that our algorithm for the vertex cover number cannot be adapted to twin cover. Let X be a twin cover of G of size k. Then $G \setminus X$ is a disjoint collection of cliques. We can partition $G \setminus X$ into at most 2^k twin classes, where some of them can also be empty. Two cliques are in the same twin class if they have the same neighbourhood in X. While designing an FPT algorithm parameterized by the vertex cover number, for each I_i and C_j, we associate a variable x_{ij} that indicates $|I_i \cap C_j| = x_{ij}$, that is, x_{ij} denotes the number of vertices in twin class I_i that go to C_j. Here as each twin class consists of cliques of arbitrary sizes, we need to introduce a variable for each clique to denote the number of vertices of the clique that go to the solution. As the number of cliques are arbitrary, the number of variables would not be bounded by a function of k. Therefore, the question of whether the $\mathcal{T}_{h+1}$-FREE EDGE DELETION problem is FPT parameterized by the twin cover number remains open.

9.3 $\mathcal{T}_{h+1}$-FREE EDGE DELETION parameterized by treewidth

To show W[1]-hardness of $\mathcal{T}_{h+1}$-FREE EDGE DELETION, we reduce from MINIMUM MAXIMUM OUTDEGREE, which is known to be W[1]-hard parameterized by the treewidth of the graph [123]. An *orientation* of an undirected graph is an assignment of a direction to each of its edges. The MINIMUM MAXIMUM OUTDEGREE Problem (MMO) takes as input an undirected, edge-weighted graph $G = (V, E, w)$, where V, E, and w denote the set of vertices of G, the set of edges of G, and an edge-weight function $w : E \to Z^+$, respectively, and asks for an orientation of G that minimizes the resulting maximum weighted outdegree taken over all vertices in the oriented graph. More formally

MINIMUM MAXIMUM OUTDEGREE
Input: An undirected edge-weighted graph $G = (V, E, w)$, where w denote an edge-weight function $w : E \to Z^+$ where the edge weights w are given in unary, and a positive integer r.
Question: Is there an orientation of the edges of G such that, for each $v \in V(G)$, the sum of the weights of outgoing edges from v is at most r?

Theorem 9.3.1. *The $\mathcal{T}_{h+1}$-FREE EDGE DELETION problem is W[1]-hard when parameterized by the treewidth of the input graph.*

Proof. Let $I = (G = (V, E, w), r)$ be an instance of MINIMUM MAXIMUM OUTDEGREE. We construct an instance $I' = (G', k)$ of $\mathcal{T}_{h+1}$-FREE EDGE DELETION the following way. See Figure 9.1. The construction of G' starts with $V(G') := V(G)$ and then add the following new vertices and edges.

1. For each edge $(u, v) \in E(G)$, create a set of $w(u, v)$ vertices $V_{uv} = \{x_1^{uv}, \ldots, x_{w(u,v)}^{uv}\}$. Make u and v adjacent to every vertex of V_{uv}. For every $1 \leq i \leq w(u, v) - 1$, introduce an edge (x_i^{uv}, x_{i+1}^{uv}) and an edge $(x_{w(u,v)}^{uv}, x_1^{uv})$.

2. Let $\omega = \sum_{e \in E(G)} w(e)$ and $h = 4\omega$. For each $u \in V(G)$, we add a set $V_u = \{x_1^u, \ldots, x_{h-r-1}^u\}$ of $h - r - 1$ new vertices and make them adjacent to u.

3. We set $k = \omega$.

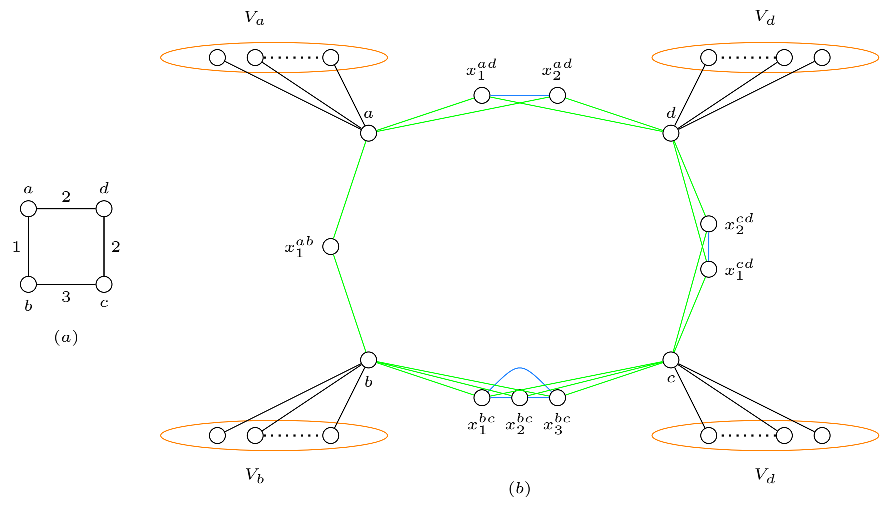


Figure 9.1: The reduction from MINIMUM MAXIMUM OUTDEGREE to T_{h+1}-FREE EDGE DELETION in Theorem 9.3.1. (a) An instance (G, r) of MINIMUM MAXIMUM OUTDEGREE with $r = 3$. The orientation $(a, d), (d, c), (c, b), (b, a)$ satisfies the property that for each $v \in V(G)$, the sum of the weights of outgoing edges from v is at most 3. (b) The graph G' produced by the reduction algorithm.

Clearly I' can be computed in time polynomial in the size of I. We now show that the treewidth of G' is bounded by a function of the treewidth of G. We do so by modifying an optimal tree decomposition T of G as follows:

- For every edge (u, v) of G, we take an arbitrary node in T whose bag X contains both u and v; add to this node a chain of nodes $1, 2, \ldots, w(u, v) - 1$ such that the bag of node i is

$$X \cup \{x_1^{uv}, x_i^{uv}, x_{i+1}^{uv}\}.$$

- For every edge $u \in V(G)$, we take an arbitrary node in T whose bag X contains u. Add to this node a chain of nodes $1, 2, \ldots h-(r+1)$ such that the bag of node i is $X \cup \{x_i^u\}$ where $x_i^u \in V_u$.

It is easy to verify that the result is a valid tree decomposition of G' and its width is at most the treewidth of G plus three. Now we show that our reduction is correct. That is, we prove that $I = (G = (V, E, w), r)$ is a yes instance of Minimum Maximum Outdegree if and only if $I' = (G', k)$ is a yes instance of $\mathcal{T}_{h+1}$-Free Edge Deletion. Let D be the directed graph obtained by an orientation of the edges of G such that for each vertex the sum of the weights of outgoing edges is at most r. We claim that the set of edges

$$E' = \bigcup_{(u,v)\in E(D)} \{(v, x) \mid x \in V_{uv}\} \subseteq E(G')$$

is a solution of I'. In Figure 9.1, the orientation $(a, d), (d, c), (c, b), (b, a)$ satisfies the property that for each $v \in V(G)$, the sum of the weights of outgoing edges from v is at most 3. Therefore $E(D) = \{(a, d), (d, c), (c, b), (b, a)\}$ and $E' = \{(d, x) \mid x \in V_{ad}\} \cup \{(d, x) \mid x \in V_{dc}\} \cup \{(b, x) \mid x \in V_{bc}\} \cup \{(a, x) \mid x \in V_{ab}\}$. Note that the edges of D are directed. Clearly, we have $|E'| = \omega$. We need to show that $\widetilde{G'} = G' \backslash E'$ does not contain any connected components of size $h+1$. Observe that every connected component in $\widetilde{G'}$ contains exactly one vertex from $V(G)$. For each $u \in V(\widetilde{G'}) \cap V(G)$, let C_u be the component of $\widetilde{G'}$ that contains u. Then $C_u = \{u\} \cup V_u \cup \bigcup_{(u,v)\in E(D)} V_{uv}$. For each $u \in V(D)$, let w^u_{out} denote the sum of the weights of outgoing edges of vertex u in D. Note that for every $u \in V(G)$, $\Big|\bigcup_{(u,v)\in E(D)} V_{uv}\Big| = w^u_{\text{out}} \leq r$ and $|V_u| = h-(r+1)$. Therefore we have $|C_u| \leq 1 + h - (r+1) + r = h$.

For the reverse direction, let $E' \subseteq E(G')$ be a solution for I', that is, $|E'| = \omega$ and $G' \setminus E'$ does not contain any connected component of size more than h. We first claim that deletion of E' from G' destroys all paths between any pair of vertices $u, v \in V(G) \cap V(\widetilde{G'})$. For the sake of contradiction, let us assume that there is a path between u and v in $\widetilde{G'} = G' \setminus E'$. Note that u (resp. v) is adjacent to $h-(r+1)$ vertices of V_u (resp. V_v) in G'. If there is a path between u and v in $G' \setminus E'$ then we get a connected component C_{uv} consists of u, v and at least $|V_u|+|V_v|-\omega$ vertices of $V_u \cup V_v$. The reason is this. If E' contains s edges between u and V_u or between v and V_v, then C_{uv} contains $|V_u|+|V_v|-s$ vertices of $V_u \cup V_v$. Thus, we have

$$\begin{aligned}
|C_{uv}| &\geq 2h - 2(r+1) - \omega + 2 \\
&= 8\omega - 2r - \omega \\
&\geq 8\omega - 2\omega - \omega \qquad\qquad \text{as } r < \omega \\
&= 5\omega \\
&\geq 4\omega + 1 \\
&= h + 1
\end{aligned}$$

This contradict the assumption that $G' \setminus E'$ does not contain any connected component of size more than h. This concludes the proof of the claim.
As we delete at most ω edges, a solution E' must contain either $E_{uv} = \big\{(u,x) \mid x \in V_{uv}\big\}$ or $E_{vu} = \big\{(v,x) \mid x \in V_{uv}\big\}$ for every edge $(u,v) \in E(G)$; otherwise there will be a path between u and v. We now define a directed graph D by $V(D) = V(G)$ and

$$E(D) = \Big\{(u,v) \mid E_{vu} \subseteq E'\Big\} \bigcup \Big\{(v,u) \mid E_{uv} \subseteq E'\Big\}.$$

We claim that for each vertex x in D the sum of the weights of outgoing edges is at most r. For the sake of contradiction, suppose there is a vertex x in D for which $w^x_{\text{out}} > r$. In this case, we observe that x is adjacent to $h - (r+1) + w^x_{\text{out}} \geq h - (r+1) + (r+1) = h$ vertices in graph $\widetilde{G'} = G' \setminus E'$. This is a contradiction as vertex x and its h neighbours form a connected component of size at least $h+1$, which is a forbidden graph in $\widetilde{G'}$. □

9.4 $\mathcal{T}_{h+1}$-Free Edge Deletion on planar graphs

Enright and Meeks [44] have discussed the importance of studying $\mathcal{T}_{h+1}$-Free Edge Deletion on planar graphs. We show that $\mathcal{T}_{h+1}$-Free Edge Deletion remains NP-complete even when restricted to planar graphs. To prove this, we give a polynomial time reduction from Multiterminal Cut. The Multiterminal Cut problem can be defined as follows: Given a graph $G = (V,E)$, a set $T = \{t_1, t_2, ..., t_p\}$ of p specified vertices or terminals, and a positive weight $w(e)$ for each edge $e \in E$, find a minimum weight set of edges $E' \subseteq E$ such that the removal of E' from E disconnects each terminal from all the others. Dahlhaus et al. [32] proved the following result:

Theorem 9.4.1. *[32] If p is not fixed, the* Multiterminal Cut *problem for planar graphs* *is NP-hard even if all edge weights are equal to 1.*

Theorem 9.4.2. *The $\mathcal{T}_{h+1}$-*Free Edge Deletion *problem is NP-complete even when restricted to planar graphs.*

Proof. It is easy to see that the problem is in NP. In order to obtain the NP-hardness result for the $\mathcal{T}_{h+1}$-Free Edge Deletion problem, we obtain a polynomial reduction from the Mutliterminal Cut problem on planar graphs with all edge weights equal to 1. Let $I = (G, T = \{t_1, t_2, ..., t_p\}, \ell)$ be an instance of Multiterminal Cut. The objective in Multiterminal Cut is to find a set $E' \subseteq E$ of at most ℓ edges such that the removal of E' from E disconnects each terminal from all the others. We produce an equivalent instance $I' = (G', k, h)$ of $\mathcal{T}_{h+1}$-Free Edge Deletion in the following way. Start with $G = G'$ and then add the following new vertices and edges. For each $t \in T$, we introduce a set V_t of $\frac{h+1}{2} + \ell$ vertices and make them adjacent to t. We take $h = 100n^3$. This completes the construction of G'. We set $k = \ell$. Let us now show that I and I' are equivalent instances.

Assume first that there exists a set $E' \subseteq E(G)$ of at most ℓ edges such that the removal of E' from E disconnects each terminal from all the others. We claim that the same set $E' \subseteq E(G')$ is a solution of I'. That is, we show that

$$\widetilde{G'} = G' \setminus E'$$

does not contain any connected component of size $h + 1$. For each $t \in T$, let C_t be the component of $\widetilde{G'}$ that contains t. Note that t is adjacent to every vertex in V_t and some vertices in $V(G)$. Therefore the size of C_t is at most $n + \frac{100n^3+1}{2} + \ell$. This is true because there is no path between t and any vertex in $V_{t'} \cup \{t'\}$ for all $t' \in T$, $t \neq t'$. Thus, we have

$$\begin{aligned} |C_t| &\leq n + \frac{100n^3 + 1}{2} + \ell \\ &\leq n + \frac{100n^3 + 1}{2} + |E(G)| \\ &\leq n + \frac{100n^3 + 1}{2} + \binom{n}{2} \\ &\leq 100n^3 \\ &= h \end{aligned}$$

Hence the size of each component in $\widetilde{G'}$ is at most h.

Conversely, suppose that there exists a set $E' \subseteq E(G')$ of k edges such that $\widetilde{G'} = G' \setminus E'$ does not contain any connected component of size $h+1$. We claim that there is no path between t_i and t_j in $\widetilde{G'}$ for all $1 \leq i, j \leq k$ and $i \neq j$. For the sake of contradiction, assume that there is a path between terminals t_i and t_j in $\widetilde{G'}$. Note that t_i and t_j are each adjacent to $\frac{h+1}{2} + \ell$ many pendent vertices in G', and we have deleted at most $k = \ell$ many edges. Therefore the connected component containing t_i and t_j contains at least $h+1$ vertices, which is a contradiction. This concludes the proof of the claim.
We now claim $S = E' \cap E(G)$ is a solution of I. That is, we claim that $G \setminus S$ disconnects each terminal from all the others. Note that $|S| \leq |E'| \leq \ell$. For the sake of contradiction, assume that there is a path between two terminals t_i and t_j in $G \setminus S$. Then there is also a path between t_i and t_j in $G' \setminus E'$. Note that if there exists a path between two terminals t_i and t_j in $G' \setminus E'$ then clearly we get a connected components of size $h+1$, a contradiction. Therefore $G \setminus S$ disconnects each terminal from all the other terminals, and hence I is a yes-instance. □

9.5 $\mathcal{T}_{h+1}$-Free Arc Deletion parameterized by solution size

Enright and Meeks [44] explained the importance of studying $\mathcal{T}_{h+1}$-Free Arc Deletion. A directed acyclic graph (DAG) is a directed graph with no directed cycles. One natural problem mentioned in [44] is to consider whether there exists an efficient algorithm to solve this problem on directed acyclic graphs. In this section, we show that the problem is W[2]-hard parameterized by the solution size k, even when restricted to directed acyclic graphs (DAG). We prove this result via a reduction from Hitting Set. In the Hitting Set problem we are given as input a family $\mathcal{F}$ over a universe U, together with an integer k, and the objective is to determine whether there is a set $B \subseteq U$ of size at most k such that B has nonempty intersection with all sets in $\mathcal{F}$. It is proved in [31] (Theorem 13.28) that Hitting Set problem is W[2]-hard parameterized by the solution size.

Theorem 9.5.1. *The $\mathcal{T}_{h+1}$-Free Arc Deletion problem is W[2]-hard parameterized by*

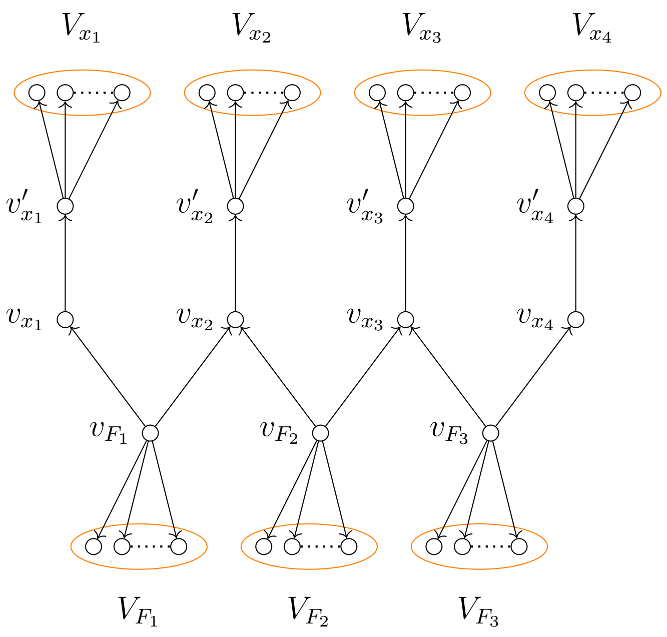


Figure 9.2: The graph in the proof of Theorem 9.5.1 constructed from HITTING SET instance $U = \{x_1, x_2, x_3, x_4\}$, $F = \{\{x_1, x_2\}, \{x_2, x_3\}, \{x_3, x_4\}\}$ and $k = 2$.

the solution size k, even when restricted to directed acyclic graphs.

Proof. Let $I = (U, \mathcal{F}, k)$ be an instance of HITTING SET where $U = \{x_1, x_2, \ldots, x_n\}$. We create an instance $I' = (G', k', h)$ of $\mathcal{T}_{h+1}$-FREE ARC DELETION the following way. For every $x \in U$, create two vertices v_x and v'_x and add a directed edge (v_x, v'_x). For every $F \in \mathcal{F}$, create one vertex v_F. Next, we add a directed edge (v_F, v_x) if and only if $x \in F$. For each $x \in U$, we add a set V_x of $\frac{h}{n}$ many new vertices and add a directed edge from v'_x to every vertex of V_x. We specify the value of h at the end of the construction. For each vertex $F \in \mathcal{F}$, we add a set V_F of $(h+1) - \sum\limits_{x \in F} |V_x|$ new vertices and add a directed edge from v_F to every vertex of V_F. Finally, we set $k' = k$ and $h = n^c$ for some large constant c. This completes the construction of G'. Next, we show that I and I' are equivalent instances.

Let us assume that there exists a subset $S \subseteq U$ such that $|S| \leq k$ and $S \cap F \neq \emptyset$ for all $F \in \mathcal{F}$. We claim that every vertex in $\widetilde{G'} = G' \setminus \bigcup\limits_{x \in S} (v_x, v'_x)$ can reach at most h vertices. Let us assume that there exists a vertex in $\widetilde{G'}$ which can reach more than h vertices. Clearly that vertex must be v_F for some $F \in \mathcal{F}$. Without loss of generality assume that $x_1 \in S \cap F$. As we have removed the edge (v_{x_1}, v'_{x_1}) from G', clearly v_F cannot reach any vertex in V_{x_1}.

Note that in such a case v_F cannot reach more than h vertices as $h = n^c$ for some large constant. In particular, v_F can reach at most $h + 1 - (\sum_{x \in F} |V_x|) + (\sum_{x \in F \setminus \{x_1\}} |V_x|) < h$ vertices.

In the other direction, let us assume that there exists a set $E' \subseteq E(G')$ such that $|E'| \leq k$ and every vertex in $\widetilde{G'} = G' \setminus E'$ can reach at most h vertices. First we show that, given a solution E' we can construct another solution E'' such that $E'' \subseteq \bigcup_{x \in U} (v_x, v'_x)$ and $|E''| \leq |E'|$. To do this, we observe that the only vertices that can possibly reach more than h vertices are v_F. Note that if E' contains an edge of the form (v_F, u) for some $u \in V_F$ then we can replace it by an arbitrary edge (v_x, v'_x) for some $x \in F$. This will allow us to disconnect at least $\frac{h}{n}$ vertices from v_F rather than just 1. Similar observation can be made for edges of type (v_F, v_x) for some $x \in F$ by replacing it with edge (v_x, v'_x). Therefore, we can assume that $E'' \subseteq \bigcup_{x \in U}^{n} (v_x, v'_x)$. Next, we show that if there exists a vertex v_F such that for every $x \in F$ we have $(v_x, v'_x) \notin E''$ then v_F can reach $h + 1$ vertices. Clearly, v_F can reach V_F, $\{v_x \mid x \in F\}$ and also $\{V_x \mid x \in F\}$. Due to construction, this set is of size more than h. This implies that for every $F \in \mathcal{F}$, there exists an edge (v_x, v'_x) for some $x \in F$ which is included in E''. As $|E''| \leq k$, we can define $S = \{x \mid (v_x, v'_x) \in E''\}$. Due to earlier observations, S is a hitting set of size at most k. □

9.6 Closing Remarks and Future Directions

The main contributions is that the $\mathcal{T}_{h+1}$-Free Edge Deletion problem is W[1]-hard when parameterized by the treewidth of the input graph. Thus we resolved a conjecture stated by Enright and Meeks [44] concerning the complexity of $\mathcal{T}_{h+1}$-Free Edge Deletion parameterized by the treewidth of the input graph. We also studied the following important open questions stated in [44]: The $\mathcal{T}_{h+1}$-Free Edge Deletion problem is NP-complete even when restricted to planar graphs; and the $\mathcal{T}_{h+1}$-Free Arc Deletion problem is W[2]-hard parameterized by the solution size k, even when restricted to directed acyclic graphs. However, it remains open whether $\mathcal{T}_{h+1}$-Free Edge Deletion problem is FPT when parameterized by the solution size k. See Figure 9.3 for a schematic representation of the relationship between selected graph parameters [94]. Note that $A \rightarrow B$ means that there exists a function f such that for all graphs, $f(A(G)) \geq B(G)$; therefore the existence of an FPT algorithm parameterized by B implies the existence of an FPT algorithm

$$\begin{tikzcd}
 & & & \mathrm{cw} & & & \\
 & \mathrm{tw} \arrow[urr] & & & & \mathrm{mw} \arrow[ull] & \\
\mathrm{fvs} \arrow[ur] & & \mathrm{pw} \arrow[ul] & & \mathrm{nd} \arrow[ur] & & \mathrm{tc} \arrow[ul] \\
 & & & \mathrm{vc} \arrow[ulll] \arrow[ul] \arrow[ur] \arrow[urrr] & & &
\end{tikzcd}$$

Figure 9.3: Relationship between vertex cover (vc), neighbourhood diversity (nd), twin cover (tc), modular width (mw), feedback vertex set (fvs), pathwidth (pw), treewidth (tw) and clique width (cw). Arrow indicate generalizations, for example, treewidth generalizes both feedback vertex set and pathwidth.

parameterized by A, and conversely, any negative result parameterized by A implies the same negative result parameterized by B. Gaikwad and Maity [61] proved that the $\mathcal{T}_{h+1}$-Free Edge Deletion problem is fixed-parameter tractable when parameterized by the vertex cover number of the input graph. Here we have proved that the $\mathcal{T}_{h+1}$-Free Edge Deletion problem is W[1]-hard when parameterized by the treewidth of the input graph. The parameterized complexity of the $\mathcal{T}_{h+1}$-Free Edge Deletion problem remains open when parameterized by other structural parameters such as feedback vertex set, pathwidth, treedepth, neighbourhood diversity, cluster vertex deletion set, modular width etc.

Chapter 10

MaxMin Separation Problems

In this chapter, we study the parameterized complexity of the MaxMin versions of two fundamental separation problems: Maximum Minimal st-Separator and Maximum Minimal Odd Cycle Transversal (OCT), both parameterized by the solution size. In the Maximum Minimal st-Separator problem, given a graph G, two distinct vertices s and t and a positive integer k, the goal is to determine whether there exists a minimal st-separator in G of size at least k. Similarly, the Maximum Minimal OCT problem seeks to determine if there exists a minimal set of vertices whose deletion results in a bipartite graph, and whose size is at least k. We demonstrate that both problems are fixed-parameter tractable parameterized by k. Our FPT algorithm for Maximum Minimal st-Separator answers the open question by Hanaka, Bodlaender, van der Zanden & Ono [TCS 2019].

In contrast to MaxMin variants of several other deletion problems on graphs, where proving the existence of an FPT algorithm is not very difficult and treewidth-based win-win approaches usually work, the problems considered in this work pose great difficulties even if one does not care about the explicit running times of the FPT algorithm. We showcase several barriers that both these problems pose and show how a clever combination of different combinatorial ideas finally work in overcoming them.

One unique insight from this work is the following. We use the meta-result of Lokshtanov, Ramanujan, Saurabh & Zehavi [ICALP 2018] that enables us to reduce our problems to highly unbreakable graphs. This is interesting, as an explicit use of the recursive understanding and randomized contractions framework of Chitnis, Cygan, Hajiaghayi, Pilipczuk

& Pilipczuk [SICOMP 2016] to reduce to the highly unbreakable graphs setting (which is the result that Lokshtanov et al. tries to abstract out in their meta-theorem) does not seem obvious because certain "extension" variants of our problems are W[1]-hard.

Another interesting feature of these results is that in the MaxMin setting, it is not straight-forward to use an FPT algorithm for the *st*-Separator problem to design an FPT algorithm for the OCT problem (unlike in many other settings). In fact, our FPT algorithms for both the problems are independent of each other. This chapter is based on the paper [59].

10.1 Challenges and Our Approach

Problem 1: MaxMin st-Sep. In this work, for the first time, we use the power of highly unbreakable instances to show fixed-parameter tractability of some MaxMin separation problems. For two positive integers q, k, a graph G is called (q, k)-unbreakable if *no* vertex set of size *at most* k can disconnect two (large) sets of size at least q each. For the purposes of an informal discussion, we say a graph is unbreakable if it is (q, k)-unbreakable for some values of q and k.

Chitnis et al. [28] developed a win-win approach based on this unbreakable structure of the graph. The approach has two parts: recursive understanding and randomized contractions. In the first part, a large enough part of the input graph is detected that is unbreakable and has small boundary to the rest of the graph. In the second part, a family of "partial solutions" are computed for this unbreakable part of the graph depending on how the solution for the whole graph interacts with its boundary. If the unbreakable part is large enough, then there exists an irrelevant edge/vertex that does not participate in the computed partial solutions, which helps in reducing the size of the graph.

Unfortunately, at the first glance it looks impossible to use this approach for solving any of the two problems MaxMin *st*-Sep and MaxMin OCT. The problem is that in the above approach one needs to find partial solutions on unbreakable graphs for a more general "extension-kind" of problem. In particular, the family of partial solutions should be such that if there exists a solution for the whole graph, then one should be able to replace the part of this solution that intersects with the unbreakable part, with one of the computed partial solutions. To do so, for example, it seems necessary to, in some implicit way at least, guess how an optimum solution of the whole graph intersects with the boundary of this unbreakable part and the partial solution should at least try to be "compatible" with this guess. The bottleneck here is that given a vertex v, finding whether the graph G has *any* minimal st-separator *containing* v is NP-hard (see Section 10.3). Thus, the problem of determining, given a subset of vertices X, whether G has a minimal st-separator of size at least k, that contains X, is W[1]-hard (it is in fact para-NP-hard) (see Section 10.3). Therefore, a first glance suggests that computing "partial solutions" may be W[1]-hard in general. One can ask if the hardness holds even when the graph is unbreakable (which is our scenario). It turns out yes, because a result by Lokshtanov et al. [103], shows that the FPT algorithm for unbreakable graphs can be lifted to an FPT algorithm on general graphs for problems definable in Counting Monadic Second Order (CMSO) Logic. Since the extension version is also CMSO definable, there should not be an FPT algorithm for the extension version even on unbreakable graphs.

Despite this issue, we show that the core lies in solving the problem on unbreakable graphs, and in fact the problem on unbreakable graphs is FPT using non-trivial insights. In fact the result of Lokshtanov et al. [103] comes to rescue. In [103] Lokshtanov et al. show that, in order to show fixed-parameter tractability of CMSO definable problems parameterized by the solution size (say k), it is enough to design FPT algorithms for such problems when the input graph is (q, k)-unbreakable, for some large enough q that depends only on k. The highlight of this result, compared to explicitly doing the above-mentioned approach of recursive understanding and randomized contractions is that, this results says that it is enough to solve the *same* problem on unbreakable graphs. This allows us to skip taking the provably hard route of dealing with extension-kind problems.

After overcoming the above issues, we can safely say that the core lies in designing an FPT algorithm for the unbreakable case, which itself is far from obvious. We give a technical overview of this phase in Section 10.2.

Problem 2: MaxMin OCT. The challenges continue for the MaxMin OCT problem. In parameterized complexity, the first FPT algorithm for the classical Odd Cycle Transversal (OCT) problem parameterized by the solution size, introduced the technique of Iterative Compression [115]. Using this, it was shown that OCT reduces to solving at most $3^k \cdot n$ instances of the polynomial-time st-Separator problem. Most algorithms for different variants of the OCT problem, like INDEPENDENT OCT, use the same framework and reduce to essentially solving the variant of st-SEPARATOR, like INDEPENDENT st-SEPARATOR [102].

For the MAXMIN variant, unfortunately this is not the case. In fact our FPT algorithm for MAXMIN OCT is independent of our FPT algorithm for MAXMIN st-SEP. The reason is again that finding a minimal odd cycle transversal set (oct) that extends a given vertex is NP-hard (see Section 10.6). At the core of the Iterative Compression based approach for OCT, a subset of vertices X is guessed to be in the solution, and after deleting X, the problem reduces to finding an st-separator, for some s, t. In particular, the final solution is this set X union a minimum st-separator. For the MAXMIN case, this amounts to finding a minimal st-separator that together with X forms a minimal oct. Since the extension version of both the MAXMIN st-SEPARATOR and MAXMIN OCT are para-NP-hard, this leaves little hope to use the same approach for MAXMIN versions.

Having eliminated this approach, we again go back to the approach via unbreakable graphs. This time again the core lies in designing an FPT algorithm when the graph is highly unbreakable and this algorithm require lot more insights than that of the MAXMIN st-SEP. We give a technical overview of this stage in Section 10.2.

10.1.1 The Sunflower Lemma

Now, we state the Sunflower Lemma. We first define the terminology used in the statement of the next lemma. Given a set system $(U, \mathcal{F})$, where U is a set and $\mathcal{F}$ is a family containing distinct subsets of U, a *sunflower* with k petals and a core Y is a collection of sets $S_1, S_2, \ldots, S_k \in \mathcal{F}$ such that $S_i \cap S_j = Y$ for all $i \neq j$. The sets $S_i \setminus Y$ are called the *petals* of this sunflower. If the sets in $\mathcal{F}$ are distinct and $k \geq 2$, then none of the petals of the sunflower are empty. Note that a family of pairwise disjoint sets is a sunflower (with an empty core).

Theorem 10.1.1 (Sunflower Lemma, [45, 31])**.** *Let $\mathcal{F}$ be a family of distinct sets over a universe U, such that each set in $\mathcal{F}$ has cardinality exactly d. If $|\mathcal{F}| > d!(k-1)^d$, then $\mathcal{F}$ contains a sunflower with k petals and such a sunflower can be computed in time polynomial in $|\mathcal{F}|$, $|U|$, and k.*

10.2 Our Results and Technical Overview

Below we state two main theorems of this work.

Theorem 10.2.1. MAXIMUM MINIMAL st-SEPARATOR *parameterized by k is FPT.*

Theorem 10.2.2. MAXIMUM MINIMAL OCT *parameterized by k is FPT.*

As discussed earlier, to prove both our theorems we first show that both MAXMIN st-SEP and MAXMIN OCT are CMSO definable. We then use Proposition 10.2.3, to reduce to solving the problem on unbreakable graphs.

Theorem 10.2.3 (Theorem 1, [103]). *Let ψ be a CMSO formula. For all $c \in \mathbb{N}$, there exists $s \in \mathbb{N}$ such that if there exists an algorithm that solves CMSO[ψ] on (s, c)-unbreakable structures in time $\mathcal{O}(n^d)$ for some $d > 4$, then there exists an algorithm that solves CMSO[ψ] on general structures in time $\mathcal{O}(n^d)$.*

In particular, to prove Theorems 10.2.1 and 10.2.2, it is enough to prove Theorems 10.2.4 and 10.2.5.

Theorem 10.2.4. *For positive integers $q, k \geq 1$,* MAXIMUM MINIMAL st-SEPARATOR *on (q, k)-unbreakable graphs on n vertices can be solved in time $(k-1)^{2q} \cdot n^{\mathcal{O}(1)}$.*

Theorem 10.2.5. *For any positive integers $q, k \geq 1$* MAXIMUM MINIMAL OCT *on (q, k)-unbreakable graphs is FPT parameterized by $q + k$.*

MAXMIN st-SEP on (q, k)-unbreakable graphs. To prove Theorem 10.2.4, we develop a branching algorithm which exploits the unbreakability of the input graph. The key observation behind the algorithm is that for any two vertex sets S and T such that $s \in S$, $t \in T$ and $G[S]$ and $G[T]$ are connected, every minimal ST-separator is also a minimal st-separator. As we are working on (q, k)-unbreakable graphs, if we manage to find such sets where $|S| > q$ and $|T| > q$, then every minimal ST-separator has size at least k. In this case, we can construct a minimal ST-separator greedily in polynomial time, which can then be returned as a solution. Therefore, the goal of our branching algorithm is to construct such sets S and T.

The algorithm begins with $S = \{s\}$ and $T = \{t\}$. We use a reduction rule which ensures that at any point $N(S)$ and $N(T)$ (the neighborhoods of S and T) are minimal ST-separators in G. Clearly, we can assume that $|N(S)| < k$ and $|N(T)| < k$; otherwise, we have already found a solution. A crucial observation is that if there exists a minimal ST-separator Z of size at least k, then there exists a vertex $u \in N(S) \setminus N(T)$ (resp. $u \in N(T) \setminus N(S)$) such that $u \notin Z$. This implies that such a vertex u remains reachable from S even after removing the solution Z. This observation allows us to grow the set S to $S \cup \{u\}$ (resp. T to $T \cup \{u\}$). Since $|N(S)| < k$ (resp. $|N(T)| < k$), one can branch on all possible vertices $u \in N(S)$ (resp. $u \in N(T)$) and in each branch grow S (resp. T). Based on whether $\min(|S|, |T|)$ is $|S|$ or $|T|$, we branch on the vertices in $N(S) \setminus N(T)$ or $N(T) \setminus N(S)$, respectively, to increase the size of these sets. Once both S and T have sizes of at least q, we can greedily construct and output a minimal ST-separator (which is also a minimal st-separator) of size at least k.

MaxMin OCT on (q, k)-unbreakable graphs. The algorithm for this case uses two key lemmas, both of which provide sufficient conditions for a yes instance. Here the input is a $(q, 2k)$-unbreakable graph.

Our first key lemma (Lemma 10.7.4) says that if there exists an induced odd cycle of length at least $2q + 2$ in G, then there always exist a minimal oct in G of size at least k. The proof of this result is based on a branching algorithm, which works similarly to the branching algorithm for the MaxMin st-Separator problem on (q, k)-unbreakable graphs, by carefully selecting the sets S and T at the beginning of the algorithm. Note that we cannot use this lemma, on its own as a a stopping criterion, because one does not know how to find a long induced odd length cycle efficiently. Nevertheless, as we see later it becomes useful together with our second sufficient condition.

Our second key lemma (Lemma 10.7.5), which states a second sufficient condition, says that if there exists a large enough family of distinct (and not necessarily disjoint) induced odd cycles in G of lengths at most $2q+1$, then there always exists a minimal oct in G of size at least k. The proof of this result relies on the Sunflower Lemma [45, 31], along with the observation that the subgraph induced by the core of the sunflower must be bipartite. Such a large sunflower then serves as a certificate that any oct which is disjoint from the core of the sunflower is large. Also since the core is bipartite, there exists an oct that is disjoint from it.

Another simple, yet important observation is that, if a vertex is not part of any induced odd cycle, then deleting this vertex from the graph does not affect the solution. Again, the problem here is that determining whether a vertex passes through an induced odd cycle is NP-hard (see Section 10.6), therefore we cannot use it as a reduction rule.

Finally, using these two sufficient conditions and the observation above, we provide an FPT algorithm that, given a vertex x, either outputs an induced odd cycle containing x if it exists, or concludes correctly that one of the two scenarios mentioned above occurs. In the later situation we are done. Also if the induced odd cycle containing x, which is returned, is long, then also we are done because of the first sufficient condition. If the algorithm outputs, that there is no induced odd cycle through x, then we can safely delete x from the graph and reduce its size. Because of deleting such vertices, the new graph may no longer be unbreakable, in which case we start solving the problem on the reduced graph completely from scratch.

By running this algorithm for each $x \in V(G)$, we reduce the graph to one where each vertex is contained in some small induced odd cycle. If such a graph contains a sufficiently large number of vertices, it guarantees the existence of a large family of distinct small induced odd cycles in G, each of length at most $2q+2$, in which case we can return a yes instance because of the second sufficient condition. This result finally allows us to bound the number of vertices in the graph by $(qk)^{\mathcal{O}(q)}$. We can then solve the problem using a brute-force algorithm on this graph.

10.3 Para-NP-hardness for extending a vertex to a minimal st-separator

We begin by providing a characterization for vertices that can be a part of a minimal st-separator.

Lemma 10.3.1. *Let G be a graph containing vertices s and t. A vertex $v \in V(G)$ is in some minimal st-separator Z if and only if there is an induced path between s and t containing v.*

Proof. For the first part, assume there exists a minimal st-separator Z that contains v. By the minimality of Z, there must be a path P between s and t in the graph $G - (Z \setminus \{v\})$, which passes through v. In other words, path P is such that no vertex in Z other than v appears on it. However, since there is no induced path between s and t through v, two vertices, say a and b, on P must be adjacent, with a lying between s and v, and b lying between v and t. This creates a new path between s and t that does not include any vertex from Z, contradicting the assumption that Z is an st-separator.

For the second part, if there exists an induced path between s and t passing through v, we can construct a minimal st-separator Z that includes v. According to Definition 10.4.1, let S be the set of all vertices on the induced path from s to a, and T the set of all vertices on the induced path from b to t, where a is a predecessor of v and b a successor of v. These sets, S and T, serve as a *certificate* for the st-separator minimality for v. By Lemma 10.4.2, we can then construct a minimal st-separator that contains v. □

Lemma 10.3.2 ([80]). *Given a graph G and any three arbitrary vertices s, t and v, determining if there exist an induced path between s and t through v is NP-hard.*

Lemma 10.3.1 and Lemma 10.3.2 together shows that given a vertex v, it is NP-hard to determine if there is a minimal st-separator containing v.

10.4 Maximum Minimal st-Separator parameterized by the solution size

The goal of this section is to prove Theorem 10.2.1.

Let (G, k) be an instance of MaxMin st-separator. The goal is to reduce the task to designing an algorithm for (q, k)-unbreakable graphs. For this we first show that the problem can be expressed in CMSO (in fact in MSO). Since MaxMin st-Separator is a maximization problem, the size of the solution can potentially be as large as $\mathcal{O}(n)$. To formulate a CMSO sentence that is bounded by a function of k, we focus on a k-sized subset of the solution and encode the minimality of each of its vertices in a way that allows for its extension to a "full-blown" minimal solution.

Lemma 10.4.1. *Maximum Minimal st-Separator is CMSO-definable with a formula of length $\mathcal{O}(k)$.*

Proof. The instance (G, k) is a yes instance of MaxMin st-Sep if there exists $Z \subseteq V(G)$ of size at least k such that $G - Z$ has no st-path (equivalently s and t are in different connected components of $G - Z$), and for each $v \in Z$, $G - (Z \setminus \{v\})$ has an st-path (that is s and t are in the same connected component of $G - (Z \setminus \{v\})$).

Alternately, suppose $Z \subseteq V(G)$ (of arbitrarily large size) such that $G - Z$ has no st-path, and Z contains k distinct vertices $v_1, \dots, v_k$ such that for each $i \in [k]$, $G - (Z \setminus \{v_i\})$ contains an st-path. Then Z may not be a minimal st-separator but it always contains a minimal st-separator of size at least k. In fact, (G, k) is a yes instance if and only if such a set Z exists. These properties of Z can be incorporated as a CMSO formula ψ as follows, where $\mathbf{conn}(U)$ is a CMSO sentence that checks whether a vertex set U induces a connected graph. The CMSO description of CMSO can be, for example, found in [31].

$$\begin{aligned}\psi =& \exists Z \subseteq V(G) \Bigg(\exists v_1, v_2, \dots, v_k \in Z \Bigg(\bigwedge_{1 \le i < j \le k} v_i \neq v_j \Bigg) \\ & \wedge \exists U \subseteq V(G) \setminus Z \Bigg((s \in U) \wedge (t \in U) \wedge \mathbf{conn}(U) \Bigg) \\ & \bigwedge_{i=1}^{k} \neg \exists U \in V(G) \setminus (Z \setminus \{v_i\}) \Bigg((s \in U) \wedge (t \in U) \wedge \mathbf{conn}(U) \Bigg) \Bigg)\end{aligned}$$

It is clear that the size of the above formula ψ depends linearly on k. □

From Lemma 10.4.1 and Proposition 10.2.3, to prove Theorem 10.2.1, it is enough to prove Theorem 10.2.4.

Theorem (Theorem 10.2.4)**.** *For positive integers $q, k > 1$,* Maximum Minimal *st*-separator *on (q, k)-unbreakable graphs on n vertices can be solved in time $(k-1)^{2q}.n^{\mathcal{O}(1)}$.*

We prove Theorem 10.2.4 in Section 10.5. As mentioned earlier, given a vertex set V', it may not always be possible to extend it to a minimal st-separator. Below, we give a definition for a *certificate* for the st-separator minimality of a set $V' \subseteq V(G)$, which guarantees the existence of a minimal st-separator that contains (extends) the set V'.

Definition 10.4.1. *Let G be a graph, $s, t \in V(G)$ and $V' \subseteq V(G)$. We say that two sets of vertices S and T serve as a certificate for the st-separator minimality for V' if the following* *conditions hold: $s \in S$ and $t \in T$, $S \cap T = \emptyset$, $G[S]$ and $G[T]$ are connected subgraphs, $E_G(S,T) = \emptyset$, and for every $v \in V'$, the subgraph $G[S \cup T \cup \{v\}]$ is connected. Note that $V' \cap (S \cup T) = \emptyset$.*

Lemma 10.4.2. *Let G be a graph, and let $s, t \in V(G)$. If there exists a certificate for the st-separator minimality of $V' \subseteq V(G)$, then there exists a minimal st-separator in G that includes all the vertices of V'.*

Proof. Let S and T serve as a certificate for the st-separator minimality of V'. We will construct a set $V' \subseteq Z \subseteq V(G) \setminus (S \cup T)$ which is a minimal st-separator in G. The set Z is constructed iteratively. Initialize $Z := \emptyset$ and $G' := G[S \cup T]$. Fix an arbitrary ordering of the vertices in $V(G) \setminus (S \cup T)$.

For each vertex v in the prescribed order:

- if $G[S \cup T \cup \{v\}]$ is connected, then update $Z := Z \cup \{v\}$;
- if $G[S \cup T \cup \{v\}]$ is not connected, then update $G' := G[V(G') \cup \{v\}]$, update S to be the vertices reachable from old S in G', and update T to be the vertices reachable from old T in G'.

The process continues until all vertices in $V(G) \setminus V(G')$ have been processed. The final set Z is then returned as a minimal st-separator of G. Note that if the initial sets S and T served as a *certificate* for the minimality of the st-separator V', then $V' \subseteq Z$, as the subgraph $G[S \cup T \cup \{v\}]$ will always be connected for each $v \in V'$ during every stage of the above process. $\square$

10.5 Maximum Minimal st-Separator on (q, k)-unbreakable graphs

In this section, we prove Theorem 10.2.4. To prove Theorem 10.2.4 we design a branching algorithm that maintains a tuple (G, S, T, k, q) where G is a (q, k)-unbreakable graph, $S, T \subseteq V(G)$ and q, k are positive integers. Additionally the sets S, T satisfy the following properties.

1. $s \in S$ and $t \in T$,
2. $S \cap T = \emptyset$,
3. both $G[S]$ and $G[T]$ are connected and
4. $E(S, T) = \emptyset$.

An instance (G, S, T, k, q) satisfying the above properties is called a *valid* instance. Given a valid instance, we design a branching algorithm that outputs a minimal ST-separator of G, which is disjoint from $S \cup T$, and has size at least k, if it exists. The algorithm initializes $S := \{s\}$ and $T := \{t\}$. Note that the above-mentioned properties of the sets S, T ensure that at each stage of the algorithm, every minimal ST-separator is also a minimal st-separator.

The algorithm has one reduction rule (Reduction Rule 3), four stopping criteria (Reduction Rules 1, 2, 4 and 5) and one branching rule (Branching Rule 1). The branching rule is applied when neither the reduction rule nor the three stopping criterion can be applied. The overall idea is the following. Observe that $N(S) \cap N(T)$ is a part of any minimal ST-separator. Therefore, if $|N(S) \cap N(T)| \geq k$, then we can correctly report that G has a minimal ST-separator of size at least k. Reduction Rule 3 ensures that $N(S)$ (resp. $N(T)$) is a *minimal* ST-separator. Therefore when Reduction Rule 3 is no longer applicable, if $|N(S)| \geq k$ (resp. $|N(T)| \geq k$), then we can correctly report that G has a minimal ST-separator of size at least k. In fact, $|N(S)|$ (resp. $|N(T)|$) is a minimal ST-separator of size at least k. Otherwise, we have that both $|N(S)| < k$ and $|N(T)| < k$. In this case, we use the branching rule. Say, without loss of generality that $|S| \leq |T|$. Since $N(S)$ is a minimal ST-separator, but its size is strictly less than k and every minimal ST-separator contains $N(S) \cap N(T)$, there exists a vertex in $N(S) \setminus N(T)$ that does not belong to the solution (if there exists a solution of size at least k). In this case we branch on the vertices of $N(S) \setminus N(T)$. If we guess that a vertex $v \in N(S)$ does not belong to the solution, since $v \in N(S)$, v remains reachable from S after removing the solution. In this case, we update $S := S \cup \{v\}$. Therefore, each application of the branching rule increases the size of the smaller of the two sets S or T. When both $|S| \geq q$ and $|T| \geq q$, from the (q, k)-unbreakability of G, we know that *every* ST-separator of G has size at least k. And hence there is a minimal st-separator of size at least k.

Below we formalize the above arguments. Given $I = (G, S, T, k, q)$, we define its measure

$\mu(I) = q - \min(|S|, |T|)$. We state the reduction rules and a branching rule below. We apply the reduction rules in order exhaustively before applying the branching rule.

Lemma 10.5.1. *Let $I = (G, S, T, k, q)$ where G is (q, k)-unbreakable. If $\mu(I) \leq 0$, then every minimal ST-separator, which is disjoint from $S \cup T$, is of size at least k.*

Proof. If $\mu(I) \leq 0$, then $q \leq \min(|S|, |T|)$. For the sake of contradiction, let us assume that there exists a minimal ST-separator Z in G, which is disjoint from $S \cup T$, and has size strictly less than k. Note that $G \setminus Z$ contains two connected components of size at least q each: one containing S, say C_S, and the other containing T. Consider the separation $(C_S \cup Z, V(G) \setminus C_S)$ of G. This is a witnessing separation that G is (q, k)-breakable, which is a contradiction. $\square$

The safeness of Reduction Rule 1 is immediate from Lemma 10.5.1.

Reduction Rule Max Min *st*-sep 1. *If $\mu(I) \leq 0$, then report a yes instance.*

Reduction Rule Max Min *st*-sep 2. If $|N(S) \cap N(T)| \geq k$, then report a yes instance.

Lemma 10.5.2. *Reduction Rule 2 is safe.*

Proof. The above reduction rule is safe because the sets S and T serve as a *certificate* for the st-separator minimality of the vertex set $N(S) \cap N(T)$. From Lemma 10.4.2, there exists a minimal st-separator in G that contains $N(S) \cap N(T)$. Since $|N(S) \cap N(T)| \geq k$, we conclude that G contains a minimal st-separator of size at least k. $\square$

Lemma 10.5.3. *If there exists $v \in N(S)$ (resp. $v \in N(T)$), such that every path from v to any vertex of T (resp. S) intersects $N(S) \setminus \{v\}$ (resp. $N(T) \setminus \{v\}$), or there is no path from v to any vertex of T (resp. S), then there is no minimal ST-separator which is disjoint from $S \cup T$ and that contains v.*

Proof. When v has no path to any vertex of T, then such a vertex cannot lie on any ST-path and hence, is not a part of any minimal ST-separator.

Suppose now that $v \in N(S)$ and every path from v to any vertex of T intersects $N(S) \setminus \{v\}$. The other case when $v \in N(T)$ is symmetric. For the sake of contradiction, say there exists a minimal ST-separator Z such that $v \in Z$ and $Z \cap (S \cup T) = \emptyset$. This implies that in the graph $G - Z$, there is no path from any vertex in S to any vertex in T, but in the graph $G - (Z \setminus \{v\})$, such a path, say P, exists. Let P be a path from s' to t' in $G - (Z \setminus \{v\})$, where $s' \in S$ and $t' \in T$.

Since v cannot reach any vertex of T (in particular t') in G, without traversing another vertex, say u, in $N(S)$, consider the u to t' subpath of P (which does not contain v). Since $u \in N(S)$, let $s'' \in N(u) \cap S$. Since $Z \cap (S \cup T) = \emptyset$, there exists a path P' from s'' to t' in $G - (Z \setminus \{v\})$ that does not contain v (take the edge (s'', u), followed by the u to t' subpath of P). This contradicts that Z is an ST-separator. □

The following reduction rule ensures that both $N(S)$ and $N(T)$ are minimal ST-separators.

Reduction Rule Max Min *st*-sep 3. If there exists $v \in N(S)$ (resp. $v \in N(T)$), such that every path from v to any vertex of T (resp. S) intersects $N(S)$ (resp. $N(T)$), or there is no path from v to any vertex of T (resp. S), then update $S := S \cup \{v\}$ (resp. $T := T \cup \{v\}$).

Lemma 10.5.4. *Reduction Rule 3 is safe.*

Proof. From Lemma 10.5.3, no minimal ST-separator, that is disjoint from $S \cup T$, contains v. Since $v \in N(S)$ (resp. $v \in N(T)$), for any minimal ST-separator Z, v is reachable from S in $G - Z$, since $Z \cap S = \emptyset$ (resp. $Z \cap T = \emptyset$). Thus, Z is also a minimal separator between $S \cup \{v\}$ and T. □

Lemma 10.5.5. *When Reduction Rule 3 is no longer applicable, $N(S)$ (resp. $N(T)$) is a minimal ST-separator.*

Proof. First note that $N(S)$ (resp. $N(T)$) is an ST-separator in G which is disjoint from $S \cup T$, since $N(S) \cap T = \emptyset$ because $E(S, T) = \emptyset$.

For the sake of contradiction, say $N(S)$ is not a minimal ST-separator in G. In particular, there exists $v \in N(S)$ such that $N(S) \setminus \{v\}$ is also an ST-separator. Since Reduction Rule 3 is no longer applicable, there exists a path from v to a vertex of T, say t', which has no other vertex of $N(S)$. Such a path together with an edge from v to a vertex of S, gives an ST-path, which intersects $N(S)$ only at v. □

From Lemma 10.5.5, the safeness of Reduction Rule 4 is immediate.

Reduction Rule Max Min *st*-sep 4. If $|N(S)| \geq k$ or $|N(T)| \geq k$ then report a yes instance.

Reduction Rule Max Min *st*-sep 5. *If $N(S) \setminus N(T) = \emptyset$ or $N(T) \setminus N(S) = \emptyset$, then report a no instance.*

Lemma 10.5.6. *Reduction Rule 5 is safe.*

Proof. Suppose $N(S) \setminus N(T) = \emptyset$. The other case is symmetric. Then any ST-path uses only the vertices of $S \cup T \cup (N(S) \cap N(T))$. In this case there is a unique ST-separator in G which is disjoint from $S \cup T$. This separator is $N(S) \cap N(T)$. Since Reduction Rule 2 is no longer applicable, $|N(S) \cap N(T)| < k$. Thus G has no ST-separator of size at least k. □

Branching Rule 1. *If $|S| \leq |T|$ (resp. $|T| < |S|$) and $N(S) \setminus N(T) \neq \emptyset$ (resp. $N(T) \setminus N(S) \neq \emptyset$), then for each vertex $x \in N(S) \setminus N(T)$ (resp. $x \in N(T) \setminus N(S)$), we recursively solve the instance $(G, S \cup \{x\}, T, k, q)$ (resp. $(G, S, T \cup \{x\}, k, q)$).*

First observe that the new instances created in this branching rule are all valid, that is, the sets $S \cup \{x\}$, T (respectively S, $T \cup \{x\}$) satisfy the desired properties: the two sets $S \cup \{x\}$ and T (resp. S and $T \cup \{x\}$) are disjoint, $G[S \cup \{x\}]$ (resp. $G[T \cup \{x\}]$) is connected and $E_G(S \cup \{x\}, T) = \emptyset$ (resp. $E_G(S, T \cup \{x\})$) because $x \in N(S) \setminus N(T)$.

Lemma 10.5.7. *Branching Rule 1 is exhaustive, that is, G has a minimal ST-separator of size at least k which is disjoint from $S \cup T$ if and only if there exists $x \in N(S) \setminus N(T)$ (resp. $x \in N(T) \setminus N(S)$) such that G has a minimal separator of size at least k between $S \cup \{x\}$ and T (resp. S and $T \cup \{x\}$), which is disjoint from $S \cup T \cup \{x\}$.*

Proof. Assume that $|S| \leq |T|$. The other case is symmetric. Since Reduction Rule 4 is no longer applicable, $|N(S)| < k$. Since Reduction Rule 3 is no longer applicable and because of Lemma 10.5.5, $N(S)$ is a minimal ST-separator. Also because $N(S) \cap N(T)$ is contained in every minimal ST-separator in G (from the proof of Lemma 10.5.2), for any minimal ST-separator in G of size at least k, say Z, which is disjoint from $S \cup T$, there exists a vertex $x \in N(S) \setminus N(T)$ such that $x \notin Z$. Since $x \in N(S)$, x remains reachable from S in $G - Z$. In particular, Z is also a minimal separator between $S \cup \{x\}$ and T.

In the other direction say the instance $(G, S \cup \{x\}, T, k, q)$ reports a minimal separator, say Z, between $S \cup \{x\}$ and T of size at least k. Since $S \cup \{x\}$ and T satisfy the desired properties of a valid instance, Z is also a minimal ST-separator in G. □

Proof of Theorem 10.2.4. Initialize an instance $I = (G, S, T, k, q)$ where $S = \{s\}$ and $T = \{t\}$. Note that $\mu(I) = q - 1$. Apply Reduction Rules 1-5 exhaustively in-order. Without loss of generality, say $|S| \leq |T|$, the other case is analogous. Since none of the reduction rules are applicable, $1 \leq |N(S) \setminus N(T)| < k$. Now apply Branching Rule 1. After every application of the Branching Rule 1, $\mu(I')$, where I' is the new instance, strictly decreases if $|S| \neq |T|$. If $|S| = |T|$, then after every two applications of the Branching Rule 1, the measure μ decreases by 1. From Reduction Rule 1, if μ of an instance is at most 0, then we stop and report a yes instance. The correctness of this algorithm follows from the safeness of the Reduction Rules 1-5 and Branching Rule 1. We now argue about the running time.

Note that all reduction rules can be applied in polynomial time. Also all reduction rules, except Reduction Rule 3, is applied only once throughout the algorithm. Reduction Rule 3 is applied at most $2q$ times (or until both S and T grow to a size of q each). Thus only polynomial time is spent on all applications of all reduction rules. The branching rule branches in at most $k - 1$ instances and has depth bounded by $2q$. Therefore the overall running time is $(k-1)^{2q} \cdot n^{\mathcal{O}(1)}$. □

10.6 NP-hardness of Maximum Minimal OCT

Lemma 10.6.1. Maximum Minimal OCT *is NP-hard.*

Proof. It was shown in [81] that Maximum Weight Minimal st-Separator is NP-hard on bipartite graphs, even when all vertex weights are identical. This implies that Maximum Minimal st-Separator is NP-hard on bipartite graphs. We provide a polynomial-time reduction from the Maximum Minimal st-Separator to the Maximum Minimal OCT. Given an instance $I = (G, s, t, k)$ of the Maximum Minimal st-Separator, we construct an instance $I' = (G', k' = k)$ of the Maximum Minimal OCT as follows. We assume that $k > 1$. We consider two cases based on the bipartition of G:

Case 1: If s and t are on the same side of the bipartition, we add an edge between s and t, and subdivide it by adding two vertices u and v, creating a new graph G'.

Case 2: If s and t are on opposite sides of the bipartition, we add a subdivided edge between s and t with one new vertex u', resulting in G'.

In both cases, the newly added path between s and t in G' is denoted by P'.

We now prove that the instances I and I' are equivalent.

In the forward direction, let Z be a minimal st-separator in G of size at least k. We claim that Z is a minimal oct in G'. Since G is bipartite, any odd cycle in G' must involve s and t. Removing Z from G separates s and t, meaning $G' - Z$ contains only the newly added path P', ensuring no odd cycles in G'. Thus, Z is an oct in G'.

Next, we show that Z is minimal. For every $z \in Z$, the graph $G - (Z \setminus \{z\})$ contains a path between s and t. Lets call it P. In Case 1, this path is even-length, and in Case 2, it is odd-length. In both cases, the graph $G' - (Z \setminus \{z\})$ contains two vertex-disjoint paths P nad P' between s and t of different parities, forming an odd cycle. Therefore, Z is a minimal oct in G'.

In the backward direction, let Z' be a minimal oct in G' of size at least k. Since $k > 1$, Z' does not include the newly added vertices u, v (in Case 1) or u' (in Case 2). We claim that Z' is a minimal st-separator in G.

If Z' were not an st-separator in G, then there would exist a path between s and t in $G - Z'$, implying the presence of an odd cycle in $G' - Z'$ involving the path P'. Since Z' is a minimal oct in G', for every $z \in Z'$, the graph $G' - (Z' \setminus \{z\})$ contains an odd cycle. Therefore, $G' - (Z' \setminus \{z\})$ must contain a path P which is a vertex-disjoint st-path from P'. This implies that $G - (Z' \setminus \{z\})$ contains an st-path, proving that Z' is a minimal st-separator in G. Thus, I and I' are equivalent, completing the proof. □

Lemma 10.6.2. *Given a graph G and a vertex $v \in V(G)$, determining whether there is an induced odd cycle containing a given vertex is NP-complete.*

Proof. We know that given two vertices a and b, determining if a graph contains an induced path of odd length between a and b is NP-complete [15]. Construct a graph G' by adding a new vertex x to G and making it adjacent to vertices a and b. It is clear that there exists an induced odd cycle through x in G' if and only if there is an induced path of odd length between a and b in G. This finishes the proof. □

Note that the Lemma 10.6.2 also shows that given a vertex $x \in V(G)$, it is NP-hard to determine if there exists a minimal odd cycle transversal containing x.

10.7 MAXIMUM MINIMAL OCT parameterized by solution size

In this section, we prove Theorem 10.2.2.

Throughout this section, we call a set of vertices of G whose deletion results in a bipartite graph, an *oct* of G. To prove Theorem 10.2.2, we first show that the problem is CMSO definable (Lemma 10.7.1). Using Proposition 10.2.3, one can reduce to solving this problem on $(q, 2k)$-unbreakable graphs. On $(q, 2k)$-unbreakable graphs, we then list and prove two sufficient conditions (Lemmas 10.7.5 and 10.7.4) which always imply a yes instance (in fact the first one implies a yes instance even when the input graph is not $(q, 2k)$-unbreakable). We also make an observation about irrelevant vertices that can be deleted without changing the solution of the instance (Observation 10.7.1). Even though checking whether any one of these sufficient conditions hold or finding these irrelevant vertices, may not be efficient, nonetheless we design an FPT algorithm that correctly concludes that at least one of the sufficient conditions is met, or outputs an irrelevant vertex, whenever the number of vertices in the graph is strictly more than a number with is a function of q and k (Theorem 10.7.3). In the case when an irrelevant vertex is outputted, deleting them reduces the size of the graph but the resulting graph may not be $(q, 2k)$-unbreakable. In this case, we start from the beginning and solve the problem from scratch (on general graphs). If none of the above hold, then the number of vertices in the graph is bounded, and we can solve the problem using brute-force.

Lemma 10.7.1. *MAXIMUM MINIMAL OCT is CMSO-definable by a formula of length $\mathcal{O}(k)$.*

Proof of Lemma 10.7.1. Let (G, k) be an instance of MAXIMUM MINIMAL OCT. Then

(G,k) is a yes instance if there exists $Z \subseteq V(G)$ of size at least k such that $G - Z$ is bipartite and for each $v \in Z$, $G - (Z \setminus \{v\})$ is not bipartite.

Alternately, let $Z \subseteq V(G)$ such that Z contains k distinct vertices $v_1, \ldots, v_k$, such that $G - Z$ is bipartite and for each $i \in [k]$, $G - (Z \setminus \{v\})$ is not bipartite. Observe that if such a set Z exists, it may not be a minimal oct of G, but it definitely contains a minimal oct of size at least k. In fact, (G,k) is a yes instance if and only if such a set Z exists. We phrase this description of Z as the CMSO formula ψ as defined below.

$$\begin{aligned}\varphi \equiv \exists Z \subseteq V(G) \Bigg(& \exists v_1, v_2, \ldots, v_k \in Z \Bigg(\bigwedge_{1 \le i < j \le k} v_i \neq v_j \Bigg) \\ & \wedge \textbf{bipartite}(V(G) \setminus Z) \\ & \wedge \Bigg(\bigwedge_{i=1}^{k} \neg \textbf{bipartite}(V(G) \setminus (Z \setminus \{v_i\}) \Bigg) \Bigg)\end{aligned}$$

where $\textbf{bipartite}(W)$ is a CMSO sentence given below, which checks whether the graph induced by the vertices in W is bipartite.

$$\begin{aligned}\textbf{bipartite}(W) \equiv & \exists X \subseteq W, \exists Y \subseteq W \\ & \Bigg((X \cap Y = \emptyset) \wedge (X \cup Y = W) \\ & \wedge \forall u, v \in W \; (E(u,v) \implies (u \in X \iff v \in Y)) \Bigg).\end{aligned}$$

It is clear that the size of the above formula φ depends linearly on k. □

Because of Lemma 10.7.1 we can invoke Proposition 10.2.3. We invoke Proposition 10.2.3 with $c = 2k$. Let q be the s from this proposition that corresponds to this choice of c. We conclude that to prove Theorem 10.2.2 it is enough to prove Theorem 10.2.5.

Theorem (Theorem 10.2.5)**.** *For any positive integers $q, k > 1$* Maximum Minimal OCT *on (q,k)-unbreakable graphs is FPT parameterized by $q+k$.*

We prove Theorem 10.2.5 in Section 10.7.1. Next we define a certificate for minimality of oct for a vertex set V'. The existence of such certificates guarantees the existence of a minimal oct which contains V'.

Definition 10.7.1. *Given a graph G and a set of vertices $V' \subseteq V(G)$, we say that an induced subgraph G' of G is a certificate for the oct-minimality of V', if G' is bipartite and for every $v \in V'$, $G[V(G') \cup \{v\}]$ contains an odd cycle.*

Lemma 10.7.2. *Given a graph G and a set of vertices V', if there is a certificate for the oct-minimality of V' then there exists a minimal oct of G that contains all the vertices in V'.*

Proof. Let G' be a certificate for minimality of V'. First observe that $V' \cap V(G') = \emptyset$ because G' is bipartite, but $G[V(G') \cup \{v\}]$, for any $v \in V'$, contains an odd cycle. We now construct a minimal oct of G, say Z, iteratively as follows. Initialize $Z = \emptyset$. Fix an arbitrary ordering of the vertices in $V(G) \setminus V(G')$ (note that $V' \subseteq V(G) \setminus V(G')$). Traverse the vertices of $V(G) \setminus V(G')$ in this order. For any vertex $v \in V(G) \setminus V(G')$ in the chosen order:

- if $G[V(G') \cup \{v\}]$ is bipartite, update $G' := G[V(G') \cup \{v\}]$, that is add v to G', otherwise
- $G[V(G') \cup \{v\}]$ contains an odd cycle, in which case add v to the set Z.

When all the vertices in $V(G) \setminus V(G')$ have been processed as stated above, then the set Z is a minimal oct of G. Moreover, one can observe that if we start with G' which a certificate for minimality of V' then $V' \subseteq Z$. $\square$

10.7.1 Maximum Minimal OCT on $(q, 2k)$-unbreakable graphs

The goal of this section is to prove Theorem 10.2.5. Let (G,k) be an instance of MaxMin OCT. A vertex $v \in V(G)$ is called *irrelevant* if (G,k) is equivalent to $(G-v,k)$. To prove Theorem 10.2.5 it is enough to prove Theorem 10.7.3.

Theorem 10.7.3. *For positive integers $q,k \geq 1$, given as input a graph G which is $(q,2k)$-unbreakable on at least $(2q+2)^2(2q+2)!(k-1)^{2q+2}+1$ vertices, there exists an algorithm* *that runs in time $(qk)^{\mathcal{O}(q)} \cdot n^{\mathcal{O}(1)}$, and either returns a minimal oct of G of size at least k or, outputs an irrelevant vertex v.*

To see the proof of Theorem 10.2.5 assuming Theorem 10.7.3, observe that if the number of vertices is at most $(2q+2)^2(2q+2)!(k-1)^{2q+2}$, then the problem can be solved using brute-force. Otherwise, the algorithm of Theorem 10.7.3 either reports a yes instance, or finds an irrelevant vertex v, in which case, delete v from the graph and solve the problem on $G - v$ (which is not necessarily $(q, 2k)$-unbreakable). The rest of the section is devoted to the proof of Theorem 10.7.3.

Irrelevant vertices.

Observation 10.7.1. *If v does not participate in any induced odd cycle, then v is irrelevant.*

Proof. For the forward direction, let Z be a minimal oct of G of size at least k. Then $G - Z$ is bipartite. Additionally, for any $z \in Z$, the graph $G - (Z \setminus \{z\})$ contains an odd cycle through z, implying the existence of an induced odd cycle C_z containing z. As the cycle is an induced odd cycle, we know that the vertex v does not lie on this cycle. We will show that Z is also a minimal oct in $G - v$. Clearly, $G - (Z \cup \{v\})$ is a bipartite graph. Therefore, Z is an oct of $G - v$. For any vertex $z \in Z$, there is an induced odd cycle C_z in $G - (Z \cup \{v\})$. Therefore, Z is also a minimal odd cycle traversal of $G - v$. □

Note that we cannot explicitly design a (polynomial-time) reduction rule based on the above observation, because determining whether there is an induced odd cycle containing a given vertex is NP-complete (see Section 10.6).

Sufficient condition 1 [Long induced odd cycle in G].

Lemma 10.7.4. *For any positive integers q, k, if G is $(q, 2k)$-unbreakable and there exists an induced odd cycle in G of length at least $2q+2$, then G has a minimal oct of size at least k.*

Proof. Let C be an induced odd cycle of length at least $2q+2$ in G. Let $x, y \in V(C)$ be arbitrarily chosen vertices such that $C \setminus \{x, y\}$ contains exactly two paths S and T each of length at least q each. Moreover since C is an odd cycle one of these two paths is odd and the other is even. Without loss of generality, we can assume that S is a path of even length and T is a path of odd length. Let $Z_x := \{x\}$ and $Z_y := \{y\}$.

Below we define a procedure that iteratively grows the sets in (S, T, Z_x, Z_y) while maintaining the following invaraints.

- The sets S, T and $Z_x \cup Z_y$ are pairwise disjoint.
- $G[S]$ is connected, bipartite and $|S| \geq q$.
- $G[T]$ is connected, bipartite and $|T| \geq q$.
- $G[S \cup T \cup \{y\}]$ is a certificate for the oct-minimality for Z_x (and in particular, $G[S \cup T \cup \{y\}]$ is bipartite).
- $G[S \cup T \cup \{x\}]$ is a certificate for the oct-minimality for Z_y (and in particular, $G[S \cup T \cup \{x\}]$ is bipartite).
- $N(x) \cap S \neq \emptyset$, $N(x) \cap T \neq \emptyset$, $N(y) \cap S \neq \emptyset$ and $N(y) \cap T \neq \emptyset$.

Observe that the starting sets (S, T, Z_x, Z_y) defined earlier satisfy these invariant. Since the iterative procedure grows these sets, the last property always hold. Also $G[S \cup T \cup \{x\} \cup \{y\}]$ contains the odd cycle C. The idea is to grow these sets until Z_x or Z_y has size at least k. If this happens then we can use Lemma 10.7.2, to conclude that G has a minimal oct of size at least k.

Since G is $(q, 2k)$-unbreakable and $|S|, |T| \geq q$, we have that every ST-separator in G has size at least $2k + 1$. In particular, $|N(S) \cup N(T)| \geq 2k + 1$. Thus, if we guarantee that $(N(S) \cup N(T)) \subseteq (Z_x \cup Z_y)$, then $|Z_x \cup Z_y| \geq 2k + 1$. Hence either $|Z_x| \geq k$ or $|Z_y| \geq k$.

Towards this we grow the sets in (S, T, Z_x, Z_y) as follows. Let us call the vertices in $S \cup T \cup Z_x \cup Z_y$ as marked.

Claim 10.7.1. *Let $v \in N(S) \cap N(T)$ be an unmarked vertex. Either $G[S \cup T \cup \{y, v\}]$ or $G[S \cup T \cup \{x, v\}]$ contains an odd cycle.*

Proof. Since both $G[S]$ and $G[T]$ are connected bipartite graphs, there exists a unique bipartition, say $S = A_S \cup B_S$ and $T = A_T \cup B_T$ of S and T respectively. Recall that x has neighbours in both the sets S and T. Since the graph $G[S \cup \{x\}]$ is bipartite, without loss of generality, assume that $(N(x) \cap S) \subseteq B_S$. Since $G[T \cup \{x\}]$ is also bipartite, without loss of generality assume that $N(x) \cap T \subseteq B_T$. Since the graph $G[S \cup T \cup \{x\}]$ is connected and bipartite, we know that there is an unique bipartition of $G[S \cup T \cup \{x\}]$ which is $G[S \cup T \cup \{x\}] = ((A_S \cup A_T \cup \{x\}) \cup (B_S \cup B_T))$.

Now, let us focus on the connected graph $G[S \cup T \cup \{y\}]$. Because $G[S \cup \{y\}]$ is bipartite, without loss of generality, we can assume that $(N(y) \cap S) \subseteq B_S$.

We now show that $(N(y) \cap T) \subseteq A_T$. For the sake of contradiction, assume that this is not the case. This implies that there are two possibilities. If y has neighbours in both the sets A_T and B_T then it leads to a contradiction as $G[T \cup \{y\}]$ is bipartite. If the neighbours of y are contained in the set B_T then it leads to a contradiction as it would imply that $G[S \cup T \cup \{x, y\}]$ is bipartite.

Therefore, we get an unique bipartition $G[S \cup T \cup \{y\}] = ((A_S \cup B_T \cup \{y\}) \cup (B_S \cup A_T))$ of $G[S \cup T \cup \{y\}]$. As the vertex v has neighbours in both the sets S and T, there are four possibilities.

- If v has a neighbour in A_S and A_T then $G[S \cup T \cup \{y, v\}$ contains an odd cycle.
- If v has a neighbour in A_S and B_T then $G[S \cup T \cup \{x, v\}$ contains an odd cycle.
- If v has a neighbour in A_T and B_S then $G[S \cup T \cup \{x, v\}$ contains an odd cycle.
- If v has a neighbour in A_T and B_T then $G[S \cup T \cup \{y, v\}$ contains an odd cycle.

This finishes the proof of the claim. $\triangleright$

Case 1: $v \in N(S) \cap N(T)$**.** From Lemma 10.7.1, either $G[S \cup T \cup \{x, v\}]$ or $G[S \cup T \cup \{y, v\}]$ or both contain an odd cycle. If $G[S \cup T \cup \{x, v\}]$ contain an odd cycle, then update $Z_y := Z_y \cup \{v\}$. If $G[S \cup T \cup \{y, v\}]$ contain an odd cycle, then update $Z_x := Z_x \cup \{v\}$. If both are true then update $Z_x := Z_x \cup \{v\}$ and $Z_y := Z_y \cup \{v\}$. The sets S, T remain the same. Observe that in either case, the updated sets satisfy all the invariants.

Case 2: $v \in N(S) \setminus N(T)$**.** In this case, if $G[S \cup \{v\}]$ is bipartite, then update $S := S \cup \{v\}$. The other sets T, Z_x, Z_y remain the same. Note that the updated sets (in particular S) maintains the invariant. Otherwise $G[S \cup \{v\}]$ contain an odd cycle. In this case update $Z_x := Z_x \cup \{v\}$ and $Z_y := Z_y \cup \{v\}$.

Case 3: $v \in N(T) \setminus N(S)$**.** This case is symmetric to Case 2.

We repeat the above process until we have sets (S, T, Z_x, Z_y) such that all vertices in $N(S) \cup N(T)$ are marked. In particular, in this case $(N(S) \cup N(T)) \subseteq (Z_x \cup Z_y)$. As argued earlier, this implies either $|Z_x| \geq k$ or $|Z_y| \geq k$. In both cases, we report that G has a minimal oct of size at least k from Lemma 10.7.2. □

Sufficient condition 2 [Large family of short induced odd cycles].

Lemma 10.7.5. *Let (G, k) be an instance of* Maximum Minimal OCT. *Let d be any positive integer. If $\mathcal{F}$ is a family containing distinct induced odd cycles of G of length at most d and $|\mathcal{F}| > d(d!)(k-1)^d$ then G has a minimal oct of size at least k.*

Proof. From the Sunflower Lemma (Theorem 10.1.1), we can conclude that there exist at least k induced odd cycles $\{F_1, F_2, \ldots, F_k\} \subseteq \mathcal{F}$ such that $V(F_i) \cap V(F_j) = Y$ for all $i, j \in [k]$. As the cycles in $\mathcal{F}$ are induced odd cycles, the graph induced by the set of vertices in Y must be bipartite. Note that Y could possibly be empty. We claim that G has a minimal odd cycle transversal of size at least k. Such a minimal odd cycle transversal can be obtained by the greedy algorithm described in the proof of Lemma 10.7.2. We start with the induced subgraph $G[Y]$. Clearly, any minimal odd cycle transversal obtained by this algorithm will contain at least one vertex from $V(F_i) \setminus Y$ for each $i \in [k]$. Since the sets $V(F_i) \setminus Y$ for each $i \in [k]$ are disjoint, a minimal odd cycle transversal must have a size of at least k. □

The combination lemma.

Lemma 10.7.6. *Given a graph G, a vertex $x \in V(G)$ and positive integers d, k, there is an algorithm that runs in $(kd)^{\mathcal{O}(d)} \cdot n^{\mathcal{O}(1)}$ time, and correctly outputs one of the following:*

1. *an induced odd cycle containing x,*

2. an induced odd cycle of length at least d,

3. a family $\mathcal{F}$ of distinct induced odd cycles, each of length at most $d-1$, such that $|\mathcal{F}| \geq d \cdot d! \cdot (k-1)^d$,

4. a determination that there is no induced odd cycle containing x in G.

Proof. Suppose G contains an induced odd cycle containing x. Let C_x be one such cycle. We design an iterative algorithm that maintains a pair $(G', \mathcal{F})$, where G' is an induced subgraph of G and $\mathcal{F}$ is a family of induced odd cycles of length at most $d-1$ in G, with the following additional properties. The graph G' is guaranteed to contain the cycle C_x, if C_x existed in the first place in G, and every induced odd cycle in G' is distinct from any cycle in the family $\mathcal{F}$.

Initialize $G' := G$ and $\mathcal{F} = \emptyset$. In each iteration, the algorithm finds an arbitrary induced odd cycle of G, say F, in polynomial time, if it exists. The following cases can now arise.

1. The algorithm fails to find the cycle F. That is G' has no induced odd cycle. In this case, report that G has no induced odd cycle passing through x.

   This is correct because if G had such an induced odd cycle passing through x, then C_x exists in G and by the invariants of the algorithm C_x also exists in G', which is a candidate for the cycle F.

2. If F contains x, then return F as the induced odd cycle containing x.

3. If the length of F is at least d, then return F as an induced odd cycle of length at least d in G.

4. Otherwise, F exists but does not contain x and has a length of at most $d-1$.

   By the invariants of the pair $(G', \mathcal{F})$, F is distinct from all the cycles in $\mathcal{F}$. Update $\mathcal{F}$ by adding F to it. Then the updated $\mathcal{F}$ satisfies the required invariants.

   To update G' proceed as follows. Guess the intersection of $V(F)$ with $V(C_x)$ (in G'). Let this be F'. The number of guesses is bounded by $2^{|F|} \leq 2^{d-1}$. Since F does not contain x, F is not equal to C_x. Hence, $F \setminus F'$ is non-empty. Update $G' := G' - (F \setminus F')$. Observe that the updated G' contains C_x if the old G_x contained it. Also every induced odd cycle in the updated G' is distinct from F (that has been newly added to $\mathcal{F}$) because a non-empty subset of $V(F)$ (that is $V(F) \setminus V(F')$) has been deleted in the updated G'.

If the first, second or third condition mentioned above does not apply in each of the first $d \cdot d! \cdot (k-1)^d$ iterations, then at the end of the i-th iteration where $i = d \cdot d! \cdot (k-1)^d$, $|\mathcal{F}| = d \cdot d! \cdot (k-1)^d$. In this case, we return $\mathcal{F}$ as a large family of induced odd cycles of length at most $d-1$ of G.

This finishes the description and correctness of the algorithm. For the running time observe that in each iteration, the algorithm runs in polynomial time to find F and check the first three conditions. The fourth condition in each iteration makes 2^{d-1} guesses and the number of iterations is at most $d \cdot d! \cdot (k-1)^d$. Therefore the overall running time is $(kd)^{\mathcal{O}(d)} \cdot n^{\mathcal{O}(1)}$. □

Proof of Theorem 10.7.3. The algorithm for Theorem 10.7.3 proceeds as follows. Recall that G is a (q,k)-unbreakable graph. For each $x \in V(G)$, run the algorithm of Lemma 10.7.6 on input $(G, x, 2q+2, k)$.

If for some $x \in V(G)$, the algorithm returns an induced odd cycle of length at least $2q+2$ or a family $\mathcal{F}$ of distinct induced odd cycles of length at most $2q+1$ such that $|\mathcal{F}| \geq (2q+2)(2q+2)!(k-1)^{2q+2}$ then we return a yes instance due to Lemma 10.7.4 or 10.7.5, respectively. If the algorithm returns that there is no induced odd cycle containing x in G then we can delete x from G and solve the problem on the reduced graph. This is safe because of Observation 10.7.1. Finally, if the algorithm returns an induced odd cycle containing x of length at least $2q+2$, then again report that it is a yes instance because of Lemma 10.7.4.

If none of the above conditions hold, then for each $x \in V(G)$, the algorithm of Lemma 10.7.6 returns an induced odd cycle containing x, say C_x, of length at most $2q+1$. Note that the cycles returned for each $x \in V(G)$ in this case may not be distinct.

The following claim shows that if the number of vertices in G is large (and there is an induced odd cycle of small length for each vertex of the graph), then there exists a large family containing *distinct* induced odd cycles of small length in G, in which case we can report a yes instance by Lemma 10.7.5.

Claim 10.7.2. *If the number of vertices in G is at least $(2q+2)^2(2q+2)!(k-1)^{2q+2}+1$, then G contains a minimal oct of size at least k.*

Proof. We will construct a $\mathcal{F}$ of distinct induced odd cycles in G of length at most $2q+1$. For every vertex $x \in V(G)$, we take a cycle C_x. Unless the cycle C_x is already in $\mathcal{F}$, we will update $\mathcal{F} = \mathcal{F} \cup \{C_x\}$. As the length of cycles in C_x returned by the above algorithm is bounded by $2q+1$, if the number of vertices in G is more than $(2q+1)(2q+2)(2q+2)!(k-1)^{2q+2}$ then $|\mathcal{F}| \geq (2q+2)(2q+2)!(k-1)^{2q+2}$. Due to Lemma 10.7.5, we return a yes instance. $\triangleright$

This finishes the proof of Theorem 10.7.3.

$\square$

10.8 Closing Remarks and Future Directions

In this work, we established that both Maximum Minimal *st*-Separator and Maximum Minimal Odd Cycle Transversal (OCT) are fixed-parameter tractable parameterized by the solution size. Instead of using treewidth-based win-win approaches, we design FPT algorithms for highly unbreakable graphs for both these problems. While we have demonstrated the FPT nature of these problems, the challenge of developing efficient FPT algorithms remains open. Additionally, the edge-deletion version of the Maximum Minimal OCT can be shown to be FPT using similar techniques, but with much simpler ideas. But designing an faster/explicit FPT algorithm even for this version remains an interesting direction for future research. Finally, the parameterized complexity of the weighted version of Maximum Minimal *st*-Separator and Maximum Minimal Weight OCT remain open as very large weights cause problems while formulating the problem in CMSO, in order to reduce it to the unbreakable case.

Chapter 11

Conclusions and Open Problems

This thesis explores the parameterized complexity of several graph problems, including Harmless Set, Defensive Alliance, Offensive Alliance, Locally and Globally Minimal Defensive Alliances, F-Free Edge Deletion, T_{h+1}-Free Edge Deletion, and MaxMin Separation Problems. These problems are not only of theoretical interest but also have practical relevance in areas like social networks, epidemiology, and optimization.

A key highlight of this thesis is the resolution of several open problems mentioned in the literature. Specifically:

- We resolved an open question posed by Enright and Meeks [44] by proving that the T_{h+1}-Free Edge Deletion problem is W[1]-hard when parameterized by treewidth.
- We addressed an open question raised by Hanaka et al. [81] regarding the fixed-parameter tractability of the Maximum Minimal st-Separator problem, demonstrating that it is FPT when parameterized by solution size.
- We resolved an open problem posed by Bazgan et al. [10] regarding the parameterized complexity of Harmless Set parameterized by treewidth, showing that it is W[1]-hard.
- We answered an open question by Bernhard Bliem and Stefan Woltran [16] concerning the complexity of the Offensive Alliance problem parameterized by treewidth,

proving its W[1]-hardness.

This thesis presents new FPT algorithms, complexity classifications, and kernelization results. For example:

- We provided FPT algorithms for Harmless Set under parameters such as vertex integrity, twin cover and neighborhood diversity.
- We developed kernelization techniques for Locally Minimal Defensive Alliance on specific graph classes and subexponential algorithm on planar graphs.
- We demonstrated the W[1]-hardness of problems like Harmless set, Defensive Alliance and Offensive Alliance under restrictive structural parameters such as treedepth, feedback vertex set, pathwidth etc. showcasing the inherent complexity of these problems.

This thesis not only advances the theoretical understanding of these problems but also lays the groundwork for future research. Potential directions include designing faster algorithms for MaxMin Separation problems, addressing the weighted versions of Harmless Set and considering Maximum Minimal *st*-separator on directed graphs, and exploring complexity of $\mathcal{T}_{h+1}$-Free Edge Deletion parameterized by restrictive structural parameters along with solution size as a parameter.

In summary, this thesis significantly contributes to the landscape of parameterized complexity, offering a deeper understanding of fundamental graph problems, resolving long-standing open questions, and providing new tools and techniques for tackling challenging computational problems.

Bibliography

[1] A. Aazami and K. Stilp. Approximation algorithms and hardness for domination with propagation. *SIAM Journal on Discrete Mathematics*, 23(3):1382–1399, 2009.

[2] H. AbouEisha, S. Hussain, V. V. Lozin, J. Monnot, B. Ries, and V. Zamaraev. Upper domination: Towards a dichotomy through boundary properties. *Algorithmica*, 80(10):2799–2817, 2018.

[3] J. Araújo, M. Bougeret, V. Campos, and I. Sau. Introducing lop-kernels: A framework for kernelization lower bounds. *Algorithmica*, 84(11):3365–3406, nov 2022.

[4] J. Araújo, M. Bougeret, V. A. Campos, and I. Sau. Parameterized complexity of computing maximum minimal blocking and hitting sets. *Algorithmica*, 85(2):444–491, sep 2022.

[5] V. Bafna, P. Berman, and T. Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM Journal on Discrete Mathematics*, 12(3):289–297, 1999.

[6] K. Bagga, L. Beineke, W. Goddard, M. Lipman, and R. Pippert. A survey of integrity. *Discrete Applied Mathematics*, 37-38:13–28, 1992.

[7] C. A. Barefoot, R. Entringer, and H. C. Swart. Vulnerability in graphs—a comparative survey. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 1(38):13–22, Sept. 1987.

[8] M. Basavaraju, F. Panolan, A. Rai, M. S. Ramanujan, and S. Saurabh. On the kernelization complexity of string problems. *Theoretical Computer Science*, 730:21–31, 2018.

[9] C. Bazgan, L. Brankovic, K. Casel, H. Fernau, K. Jansen, K. Klein, M. Lampis, M. Liedloff, J. Monnot, and V. T. Paschos. The many facets of upper domination. *Theoretical Computer Science*, 717:2–25, 2018.

[10] C. Bazgan and M. Chopin. The complexity of finding harmless individuals in social networks. *Discret. Optim.*, 14(C):170–182, Nov. 2014.

[11] C. Bazgan, M. Chopin, A. Nichterlein, and F. Sikora. Parameterized approximability of maximizing the spread of influence in networks. In D.-Z. Du and G. Zhang, editors, *Computing and Combinatorics*, pages 543–554, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[12] C. Bazgan, H. Fernau, and Z. Tuza. Aspects of upper defensive alliances. *Discrete Applied Mathematics*, 266:111 – 120, 2019.

[13] R. Belmonte, E. J. Kim, M. Lampis, V. Mitsou, and Y. Otachi. Grundy distinguishes treewidth from pathwidth. *SIAM J. Discret. Math.*, 36(3):1761–1787, 2022.

[14] O. Ben-Zwi, D. Hermelin, D. Lokshtanov, and I. Newman. Treewidth governs the complexity of target set selection. *Discrete Optimization*, 8(1):87–96, 2011. Parameterized Complexity of Discrete Optimization.

[15] D. Bienstock. On the complexity of testing for odd holes and induced odd paths. *Discrete Mathematics*, 90(1):85–92, 1991.

[16] B. Bliem and S. Woltran. Defensive alliances in graphs of bounded treewidth. *Discrete Applied Mathematics*, 251:334 – 339, 2018.

[17] H. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11:1–21, 1993.

[18] H. L. Bodlaender, G. Cornelissen, and M. van der Wegen. Problems hard for treewidth but easy for stable gonality. *CoRR*, abs/2202.06838, 2022.

[19] N. Boria, F. Della Croce, and V. T. Paschos. On the max min vertex cover problem. *Discrete Applied Mathematics*, 196:62–71, 2015. Advances in Combinatorial Optimization.

[20] L. Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58(4):171 – 176, 1996.

[21] A. Cami, H. Balakrishnan, N. Deo, and R. Dutton. On the complexity of finding optimal global alliances. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 58:23–31, 2006.

[22] C. C. Centeno, M. C. Dourado, L. D. Penso, D. Rautenbach, and J. L. Szwarcfiter. Irreversible conversion of graphs. *Theoretical Computer Science*, 412(29):3693–3700, 2011.

[23] C.-W. Chang, M.-L. Chia, C.-J. Hsu, D. Kuo, L.-L. Lai, and F.-H. Wang. Global defensive alliances of trees and cartesian product of paths and cycles. *Discrete Applied Mathematics*, 160(4):479 – 487, 2012.

[24] J. Chaudhary, S. Mishra, and B. S. Panda. Minimum maximal acyclic matching in proper interval graphs. In A. Bagchi and R. Muthu, editors, *Algorithms and Discrete Applied Mathematics - 9th International Conference, CALDAM 2023, Gandhinagar, India, February 9-11, 2023, Proceedings*, volume 13947 of *Lecture Notes in Computer Science*, pages 377–388. Springer, 2023.

[25] M. Chellali and T. W. Haynes. Global alliances and independence in trees. *Discussiones Mathematicae Graph Theory*, 27(1):19–27, 2007.

[26] N. Chen. On the approximability of influence in social networks. *SIAM Journal on Discrete Mathematics*, 23(3):1400–1415, 2009.

[27] C.-Y. Chiang, L.-H. Huang, B.-J. Li, J. Wu, and H.-G. Yeh. Some results on the target set selection problem. *Journal of Combinatorial Optimization*, 25(4):702–715, 2013.

[28] R. Chitnis, M. Cygan, M. Hajiaghayi, M. Pilipczuk, and M. Pilipczuk. Designing fpt algorithms for cut problems using randomized contractions. *SIAM Journal on Computing*, 45(4):1171–1229, 2016.

[29] M. Chopin, A. Nichterlein, R. Niedermeier, and M. Weller. Constant thresholds can make target set selection tractable. *Theory of Computing Systems*, 55(1):61–83, 2014.

[30] B. Courcelle and S. Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1):77 – 114, 2000.

[31] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.

[32] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiterminal cuts. *SIAM Journal on Computing*, 23(4):864–894, 1994.

[33] M. Dom, D. Lokshtanov, S. Saurabh, and Y. Villanger. Capacitated domination and covering: A parameterized perspective. In M. Grohe and R. Niedermeier, editors, *Parameterized and Exact Computation*, pages 78–90, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[34] M. Doucha and J. Kratochvíl. Cluster vertex deletion: A parameterization between vertex cover and clique-width. In *Proceedings of the 37th International Conference on Mathematical Foundations of Computer Science*, MFCS'12, page 348–359, Berlin, Heidelberg, 2012. Springer-Verlag.

[35] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness i: Basic results. *SIAM Journal on Computing*, 24(4):873–921, 1995.

[36] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999.

[37] P. G. Drange, M. Dregi, and P. van 't Hof. On the computational complexity of vertex integrity and component order connectivity. *Algorithmica*, 76(4):1181–1202, 2016.

[38] P. G. Drange, I. Muzi, and F. Reidl. Harmless sets in sparse classes. In C. Bazgan and H. Fernau, editors, *Combinatorial Algorithms*, pages 299–312, Cham, 2022. Springer International Publishing.

[39] P. A. Dreyer and F. S. Roberts. Irreversible k-threshold processes: Graph-theoretical threshold models of the spread of disease and of opinion. *Discrete Applied Mathematics*, 157(7):1615–1627, 2009.

[40] G. L. Duarte, H. Eto, T. Hanaka, Y. Kobayashi, Y. Kobayashi, D. Lokshtanov, L. L. C. Pedrosa, R. C. S. Schouery, and U. S. Souza. Computing the largest bond and the maximum connected cut of a graph. *Algorithmica*, 83(5):1421–1458, 2021.

[41] L. Dublois, T. Hanaka, M. Khosravian Ghadikolaei, M. Lampis, and N. Melissinos. (in)approximability of maximum minimal fvs. *Journal of Computer and System Sciences*, 124:26–40, 2022.

[42] L. Dublois, M. Lampis, and V. T. Paschos. Upper dominating set: Tight algorithms for pathwidth and sub-exponential approximation. *Theoretical Computer Science*, 923:271–291, 2022.

[43] R. Enciso. *Alliances in graphs: Parameterized algorithms and on partitioning series -parallel graphs*. PhD thesis, University of Central Florida, USA, 2009.

[44] J. Enright and K. Meeks. Deleting edges to restrict the size of an epidemic: A new application for treewidth. *Algorithmica*, 80(6):1857–1889, 2018.

[45] P. Erdös and R. Rado. Intersection theorems for systems of sets. *Journal of the London Mathematical Society*, 1(1):85–90, 1960.

[46] M. R. Fellows, D. Hermelin, F. Rosamond, and S. Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, 410(1):53–61, 2009.

[47] M. R. Fellows, D. Lokshtanov, N. Misra, F. A. Rosamond, and S. Saurabh. Graph layout problems parameterized by vertex cover. In S.-H. Hong, H. Nagamochi, and T. Fukunaga, editors, *Algorithms and Computation*, pages 294–305, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[48] H. Fernau. Extremal kernelization: A commemorative paper. In L. Brankovic, J. Ryan, and W. F. Smyth, editors, *Combinatorial Algorithms*, pages 24–36, Cham, 2018. Springer International Publishing.

[49] H. Fernau and D. Raible. Alliances in graphs: a complexity-theoretic study. In *Proceeding Volume II of the 33rd International Conference on Current Trends in Theory and Practice of Computer Science*, 2007.

[50] H. Fernau and D. Raible. Alliances in graphs: a complexity-theoretic study. In J. van Leeuwen, G. F. Italiano, W. van der Hoek, C. Meinel, H. Sack, F. Plásil, and M. Bieliková, editors, *SOFSEM 2007: Theory and Practice of Computer Science, 33rd Conference on Current Trends in Theory and Practice of Computer Science, Harrachov, Czech Republic, January 20-26, 2007, Proceedings Volume II*, pages 61–70. Institute of Computer Science AS CR, Prague, 2007.

[51] H. Fernau and J. A. Rodriguez-Velazquez. A survey on alliances and related parameters in graphs. *Electronic Journal of Graph Theory and Applications*, 2(1), 2014.

[52] H. Fernau, J. A. Rodríguez, and J. M. Sigarreta. Offensive r-alliances in graphs. *Discrete Applied Mathematics*, 157(1):177 – 182, 2009.

[53] F. Fomin, D. Lokshtanov, S. Saurabh, and M. Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2019.

[54] F. V. Fomin, P. Golovach, and D. M. Thilikos. Contraction obstructions for treewidth. *Journal of Combinatorial Theory, Series B*, 101(5):302–314, 2011.

[55] F. V. Fomin, D. Lokshtanov, S. Saurabh, and M. Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2019.

[56] G. Fricke, L. Lawson, T. Haynes, M. Hedetniemi, and S. Hedetniemi. A note on defensive alliances in graphs. *Bulletin of the Institute of Combinatorics and its Applications*, 38:37–41, 2003.

[57] T. Fujito. A unified approximation algorithm for node-deletion problems. *Discrete Applied Mathematics*, 86(2):213 – 231, 1998.

[58] F. Furini, I. Ljubić, and M. Sinnl. An effective dynamic programming algorithm for the minimum-cost maximal knapsack packing problem. *European Journal of Operational Research*, 262(2):438–448, 2017.

[59] A. Gaikwad, H. Kumar, S. Maity, S. Saurabh, and R. Sharma. MaxMin Separation Problems: FPT Algorithms for st-Separator and Odd Cycle Transversal. In O. Beyersdorff, M. Pilipczuk, E. Pimentel, and N. K. Thng, editors, *42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025)*, volume 327 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 36:1–36:21, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[60] A. Gaikwad and S. Maity. On structural parameterizations of the offensive alliance problem. In D.-Z. Du, D. Du, C. Wu, and D. Xu, editors, *Combinatorial Optimization and Applications*, pages 579–586, Cham, 2021. Springer International Publishing.

[61] A. Gaikwad and S. Maity. Defensive alliances in graphs. *Theoretical Computer Science*, 928:136–150, 2022.

[62] A. Gaikwad and S. Maity. Further parameterized algorithms for the $\mathcal{F}$-free edge deletion problem. *Theoretical Computer Science*, 933:125–137, 2022.

[63] A. Gaikwad and S. Maity. Globally minimal defensive alliances. *Information Processing Letters*, 177:106253, 2022.

[64] A. Gaikwad and S. Maity. Globally minimal defensive alliances: A parameterized perspective. *CoRR*, abs/2202.02010, 2022.

[65] A. Gaikwad and S. Maity. On the harmless set problem parameterized by treewidth. In P. Mutzel, M. S. Rahman, and Slamin, editors, *WALCOM: Algorithms and Computation*, pages 227–238, Cham, 2022. Springer International Publishing.

[66] A. Gaikwad and S. Maity. Parameterized complexity of the $\mathcal{T}_{h+1}$-free edge deletion problem. In H. Fernau and K. Jansen, editors, *Fundamentals of Computation Theory (FCT 2023)*, volume 14052 of *Lecture Notes in Computer Science*, pages 221–233, Cham, 2023. Springer Nature Switzerland.

[67] A. Gaikwad and S. Maity. Offensive alliances in graphs. *Theoretical Computer Science*, 989:114401, 2024.

[68] A. Gaikwad and S. Maity. On structural parameterizations of the harmless set problem. *Algorithmica*, 86(5):1475–1511, 2024.

[69] A. Gaikwad, S. Maity, and S. Saurabh. Parameterized algorithms for locally minimal defensive alliance. *CoRR*, abs/2208.03491, 2022.

[70] A. Gaikwad, S. Maity, and S. K. Tripathi. Parameterized complexity of defensive and offensive alliances in graphs. In D. Goswami and T. A. Hoang, editors, *Distributed Computing and Internet Technology*, pages 175–187, Cham, 2021. Springer International Publishing.

[71] A. Gaikwad, S. Maity, and S. K. Tripathi. Parameterized complexity of locally minimal defensive alliances. *CoRR*, abs/2105.10742, 2021.

[72] A. Gaikwad, S. Maity, and S. K. Tripathi. Parameterized intractability of defensive alliance problem. In N. Balachandran and R. Inkulu, editors, *Algorithms and Discrete Applied Mathematics*, pages 279–291, Cham, 2022. Springer International Publishing.

[73] R. Ganian. Improving Vertex Cover as a Graph Parameter. *Discrete Mathematics & Theoretical Computer Science*, Vol. 17 no.2, Sept. 2015.

[74] R. Ganian, F. Klute, and S. Ordyniak. On structural parameterizations of the bounded-degree vertex deletion problem. *Algorithmica*, 2020.

[75] E. Ghosh, S. Kolay, M. Kumar, P. Misra, F. Panolan, A. Rai, and M. S. Ramanujan. Faster parameterized algorithms for deletion to split graphs. *Algorithmica*, 71(4):989–1006, 2015.

[76] J. C. Gibbens, J. W. Wilesmith, C. E. Sharpe, L. M. Mansley, E. Michalopoulou, J. B. M. Ryan, and M. Hudson. Descriptive epidemiology of the 2001 foot-and-mouth disease epidemic in great britain: the first five months. *Veterinary Record*, 149(24):729–743, 2001.

[77] T. Gima, T. Hanaka, M. Kiyomi, Y. Kobayashi, and Y. Otachi. Exploring the gap between treedepth and vertex cover through vertex integrity. In T. Calamoneri and F. Corò, editors, *Algorithms and Complexity*, pages 271–285, Cham, 2021. Springer International Publishing.

[78] L. Gourvès, J. Monnot, and A. T. Pagourtzis. The lazy bureaucrat problem with common arrivals and deadlines: Approximation and mechanism design. In L. Gąsieniec and F. Wolter, editors, *Fundamentals of Computation Theory*, pages 171–182, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[79] J. Guo. Problem kernels for np-complete edge deletion problems: Split and related graphs. In T. Tokuyama, editor, *Algorithms and Computation*, pages 915–926, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[80] R. Haas and M. Hoffmann. Chordless paths through three vertices. *Theoretical Computer Science*, 351(3):360–371, 2006. Parameterized and Exact Computation.

[81] T. Hanaka, H. L. Bodlaender, T. C. van der Zanden, and H. Ono. On the maximum weight minimal separator. *Theoretical Computer Science*, 796:294–308, 2019.

[82] K. Hassan Shafique and R. D. Dutton. *Partitioning a graph in alliances and its application to data clustering.* PhD thesis, University of Central Florida, USA, 2004. AAI3163605.

[83] L. H. Jamieson. *Algorithms and Complexity for Alliances and Weighted Alliances of Various Types.* PhD thesis, Clemson University, USA, 2007.

[84] L. H. Jamieson, S. T. Hedetniemi, and A. A. McRae. The algorithmic complexity of alliances in graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 68:137–150, 2009.

[85] D. S. Johnson and M. Szegedy. What are the least tractable instances of max independent set? In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '99, page 927–928, USA, 1999. Society for Industrial and Applied Mathematics.

[86] M. Kamiński, V. V. Lozin, and M. Milanič. Recent developments on graphs of bounded clique-width. *Discrete Applied Mathematics*, 157(12):2747 – 2761, 2009.

[87] R. Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, 12(3):415–440, 1987.

[88] J. Keil. The complexity of domination problems in circle graphs. *Discrete Applied Mathematics*, 42(1):51–63, 1993.

[89] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, page 137–146, New York, NY, USA, 2003. Association for Computing Machinery.

[90] B. Kerr, L. Danon, A. P. Ford, T. House, C. P. Jewell, M. J. Keeling, G. O. Roberts, J. V. Ross, and M. C. Vernon. Networks and the epidemiology of infectious disease. *Interdisciplinary Perspectives on Infectious Diseases*, 2011:284909, 2011.

[91] S. Khot and O. Regev. Vertex cover might be hard to approximate to within 2-epsilon. *Journal of Computer and System Sciences*, 74(3):335–349, 2008.

[92] M. Kiyomi and Y. Otachi. Alliances in graphs of bounded clique-width. *Discrete Applied Mathematics*, 223:91 – 97, 2017.

[93] T. Kloks. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994.

[94] D. Knop, T. Masařík, and T. Toufar. Parameterized complexity of fair vertex evaluation problems. In *MFCS*, 2019.

[95] P. Kristiansen, M. Hedetniemi, and S. Hedetniemi. Alliances in graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 48:157–177, 2004.

[96] M. Lampis. Algorithmic meta-theorems for restrictions of treewidth. *Algorithmica*, 64(1):19–37, 2012.

[97] M. Lampis. Minimum stable cut and treewidth. In N. Bansal, E. Merelli, and J. Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 92:1–92:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[98] M. Lampis, N. Melissinos, and M. Vasilakis. Parameterized Max Min Feedback Vertex Set. In J. Leroux, S. Lombardy, and D. Peleg, editors, *48th International Symposium on Mathematical Foundations of Computer Science (MFCS 2023)*, volume 272 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 62:1–62:15, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[99] H. W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983.

[100] D. Lokshtanov, D. Marx, and S. Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, pages 41–71, 2011.

[101] D. Lokshtanov, D. Marx, and S. Saurabh. Slightly superexponential parameterized problems. *SIAM Journal on Computing*, 47(3):675–702, 2018.

[102] D. Lokshtanov, F. Panolan, S. Saurabh, R. Sharma, and M. Zehavi. Covering small independent sets and separators with applications to parameterized algorithms. *ACM Transactions on Algorithms*, 16(3):32:1–32:31, 2020.

[103] D. Lokshtanov, M. S. Ramanujan, S. Saurabh, and M. Zehavi. Reducing CMSO Model Checking to Highly Connected Graphs. In I. Chatzigiannakis, C. Kaklamanis, D. Marx, and D. Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, volume 107 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 135:1–135:14, Dagstuhl, Germany, 2018. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[104] L. Lovász. *Combinatorial Problems and Exercises*. Amsterdam, Netherlands: North-Holland, 2nd edition, 1993.

[105] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41(5):960–981, Sept. 1994.

[106] L. M. Mansley, P. J. Dunlop, S. M. Whiteside, and R. G. H. Smith. Early dissemination of foot-and-mouth disease virus through sheep marketing in february 2001. *Veterinary Record*, 153(2):43–50, 2003.

[107] R. M. McConnell and J. P. Spinrad. Modular decomposition and transitive orientation. *Discrete Mathematics*, 201(1):189–241, 1999.

[108] J. Monnot, H. Fernau, and D. F. Manlove. Algorithmic aspects of upper edge domination. *Theoretical Computer Science*, 877:46–57, 2021.

[109] A. Natanzon, R. Shamir, and R. Sharan. Complexity classification of some edge modification problems. *Discrete Applied Mathematics*, 113(1):109 – 128, 2001.

[110] J. Nesetril and P. O. de Mendez. *Sparsity: Graphs, Structures, and Algorithms*. Springer Publishing Company, Incorporated, 2014.

[111] A. Nichterlein, R. Niedermeier, J. Uhlmann, and M. Weller. On tractable cases of target set selection. *Social Network Analysis and Mining*, 3(2):233–256, 2013.

[112] D. Peleg. Local majorities, coalitions and monopolies in graphs: a review. *Theoretical Computer Science*, 282(2):231 – 257, 2002.

[113] C. R., M. M., R. I., and S. N. Small alliances in graphs. In *Kučera L., Kučera A. (eds) Mathematical Foundations of Computer Science, MFCS 2007, Lecture Notes in Computer Science*, volume 4708, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[114] T. Reddy and C. Rangan. Variants of spreading messages. *Journal of Graph Algorithms and Applications*, 15(5):683–699, 2011.

[115] B. A. Reed, K. Smith, and A. Vetta. Finding odd cycle transversals. *Operations Research Letters*, 32(4):299–301, 2004.

[116] N. Robertson and P. Seymour. Graph minors. iii. planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49 – 64, 1984.

[117] J. Rodríguez-Velázquez and J. Sigarreta. Global offensive alliances in graphs. *Electronic Notes in Discrete Mathematics*, 25:157 – 164, 2006.

[118] K. H. Shafique. *Partitioning a graph in alliances and its application to data clustering.* PhD thesis, University of Central Florida, 2004.

[119] J. Sigarreta, S. Bermudo, and H. Fernau. On the complement graph and defensive k-alliances. *Discrete Applied Mathematics*, 157(8):1687 – 1695, 2009.

[120] J. Sigarreta and J. Rodríguez. On defensive alliances and line graphs. *Applied Mathematics Letters*, 19(12):1345 – 1350, 2006.

[121] J. Sigarreta and J. Rodríguez. On the global offensive alliance number of a graph. *Discrete Applied Mathematics*, 157(2):219 – 226, 2009.

[122] P. K. Srimani and Z. Xu. Distributed protocols for defensive and offensive alliances in network graphs using self-stabilization. In *2007 International Conference on Computing: Theory and Applications (ICCTA'07)*, pages 27–31, 2007.

[123] S. Szeider. Not so easy problems for tree decomposable graphs. *CoRR*, abs/1107.1177, 2011.

[124] M. Tedder, D. Corneil, M. Habib, and C. Paul. Simpler linear-time modular decomposition via recursive factorizing permutations. In *Automata, Languages and Programming*, pages 634–645, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[125] M. Thorup. All structured programs have small tree width and good register allocation. *Information and Computation*, 142(2):159 – 181, 1998.

[126] T. Watanabe, T. Ae, and A. Nakamura. On the np-hardness of edge-deletion and -contraction problems. *Discrete Applied Mathematics*, 6(1):63 – 78, 1983.

[127] D. B. West. *Introduction to Graph Theory.* Prentice Hall, 2000.

[128] M. Yannakakis. Node-and edge-deletion np-complete problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, pages 253–264, New York, NY, USA, 1978. Association for Computing Machinery.

[129] M. Zehavi. Maximum minimal vertex cover parameterized by vertex cover. *SIAM Journal on Discrete Mathematics*, 31(4):2440–2456, 2017.

[130] Édouard Bonnet, M. Lampis, and V. T. Paschos. Time-approximation trade-offs for inapproximable problems. *Journal of Computer and System Sciences*, 92:171–180, 2018.